

Building Mobile Applications with React

Native from Zero to Hero

Power by

Tamnuwat Valeeprakhon

Digital Academy Thailand

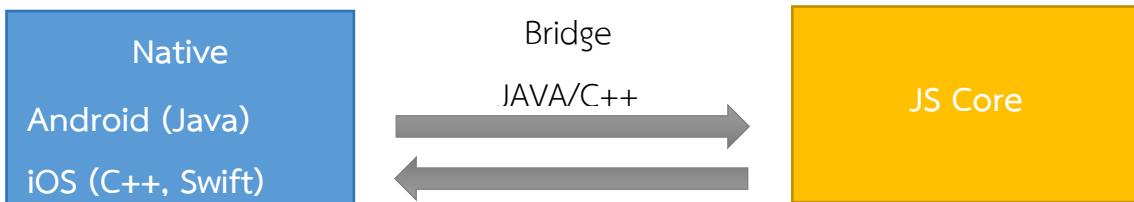
บทที่ 1 การสร้าง React-Native โปรเจค Expo

React – Native คืออะไร

หลายๆ คนคงรู้แล้วว่า react-native ถูกพัฒนาโดย บริษัท Facebook แต่ก็เกิดข้อสงสัยสงสัยว่าเจ้า react-native มันคืออะไรแล้วทำไม่มันถึงสามารถสร้างแอพพลิเคชั่นบนมือถือทั้งในระบบปฏิบัติการ Android และ iOS ได้ ก่อนอื่นเลย ต้องกล่าวว่าย้อนไปถึงอดีตในช่วงที่ Facebook กำลังโด่งดังอย่างพลุแตกและมียอดผู้ใช้งานเพิ่มขึ้นเรื่อยๆ ทำให้เจ้า Facebook ต้องปรับปรุงพัฒนาฟีเจอร์ใหม่ๆ เพื่อตอบโจทย์ผู้ใช้งานที่หลากหลายมากยิ่งขึ้น ในกระบวนการพัฒนานี้เองจำเป็นจะต้องใช้โปรแกรมเมอร์หลายคนทำงานเป็นทีมซึ่งทำให้เกิดความยุ่งยากลำบากมากในการพัฒนาส่วนต่างๆ ของเว็บไซต์ ทำให้ Facebook ตัดสินใจสร้าง framework ตัวหนึ่งที่ชื่อว่า ReactJS ที่เขียนด้วยภาษา Java script มาช่วยในการพัฒนาเว็บไซต์ โดยมีจุดเด่นก็คือสามารถแบ่งส่วนโปรแกรมออกเป็นส่วนย่อย (Component) เพื่อให้โปรแกรมเมอร์แต่ละคนพัฒนาส่วนย่อย ของโครงสร้างได้ จุดเด่นนี้เองทำให้สามารถพัฒนาเว็บไซต์ได้อย่างรวดเร็วขนาดที่ว่า Facebook สามารถอัปเดตฟีเจอร์ใหม่ๆ ได้ทุกวัน ที่นี่ก็ลับมาพูดถึงทางด้าน Mobile Application บ้าง เมื่อมีผู้ใช้งานโทรศัพท์มือถือสมาร์ทโฟนเพิ่มมากขึ้น Facebook เองก็เล็งเห็นช่องทางการตลาดที่เพิ่มมากขึ้น เช่นกัน จึงได้มุ่งเน้นคิดในการเพิ่ม platform การให้บริการจากเดิมที่ มีเพียงเว็บไซต์ก็ได้ขยายการให้บริการบนสมาร์ทโฟนทั้งในระบบ Android และ iOS โดยการพัฒนาแอพพลิเคชั่นบนสอง ระบบปฏิบัติการนี้ มี Facebook ได้ประสบปัญหาที่สำคัญอยู่สองประการคือ จะทำอย่างไรให้แอพพลิเคชั่นทำงานได้อย่างมีประสิทธิภาพลื่นไหลดังเช่นใช้งานบนเว็บไซต์ และจะทำอย่างไรให้การพัฒนาแอพพลิเคชั่นเป็นไปได้อย่างรวดเร็ว ทั้งนี้หลายๆ ท่านคงทราบดีอยู่แล้วว่าการพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟนแต่ละระบบปฏิบัติการ (Native Application) จำเป็นจะต้อง ใช้เครื่องมือและภาษาที่แตกต่างกัน เช่น การพัฒนาแอพพลิเคชั่นที่ทำงานบนระบบปฏิบัติการ Android จำเป็นจะต้องใช้ โปรแกรม Android Studio ที่เขียนด้วยภาษา JAVA หรือ Katlin ส่วนแอพพลิเคชั่นที่ทำงานบนระบบปฏิบัติการ iOS จำเป็นต้องใช้โปรแกรม XCode ที่เขียนด้วยภาษา Swift ใน การพัฒนา ทำให้การพัฒนาแอพพลิเคชั่นเดียวแต่ทำงานใน ระบบปฏิบัติการที่แตกต่างกันเป็นไปได้อย่างอยากลำบาก ด้วยปัญหาที่กล่าวมาข้างต้นทำให้ Facebook เกิดไอเดียใหม่ที่ แปลงโฉมวงการนักพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟนโดยการนำเอา ReactJS ที่ตัวเองมีอยู่แล้วมาต่อยอดให้พัฒนาบนมือ ถือเหมือนดังเช่นการพัฒนาแอพพลิเคชั่นแบบ Native ได้ ด้วยแนวคิดนี้จึงเป็นที่มาของคำว่า React-Native ซึ่งไอเดีย Frameworks ตัวนี้มันมีข้อดีก็คือสามารถแบ่งส่วนโปรแกรมออกเป็นส่วนย่อย (Component) เพื่อให้จ่ายต่อการพัฒนาเป็น ทีมได้ เช่นเดียวกับการพัฒนาเว็บไซต์ และยังสามารถ Compile ให้แอพพลิเคชั่นทำงานได้ทั้งระบบปฏิบัติการ Android และ iOS อย่างมีประสิทธิภาพลื่นไหล หรือแม้แต่ตั้งแต่ใช้งานบนเว็บไซต์

โครงสร้างการทำงานของ React-Native

มาถึงจุดนี้หลายคนคงมีข้อสงสัยเกิดขึ้นว่า React-Native มีกระบวนการทำงานอย่างไร เหตุไนน์จึงง่ายเหมือนกับผู้ที่ต้องการเริ่มเขียนแอพพลิเคชั่นบนมือถือ เมื่อลองสังเกตดูที่โครงสร้างการทำงานของReact-Native จะพบส่วนประกอบด้วย 3 ส่วนสำคัญตามภาพที่ 1 ดังนี้



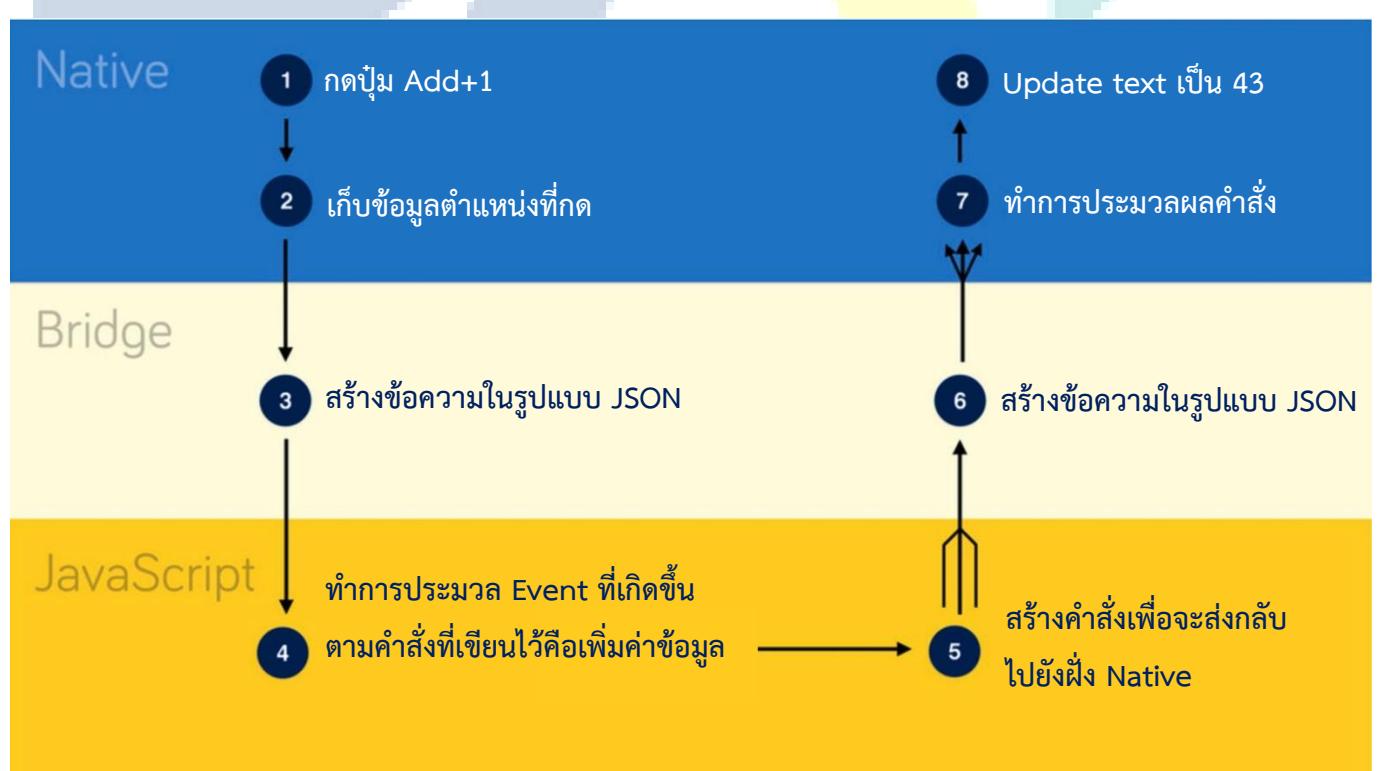
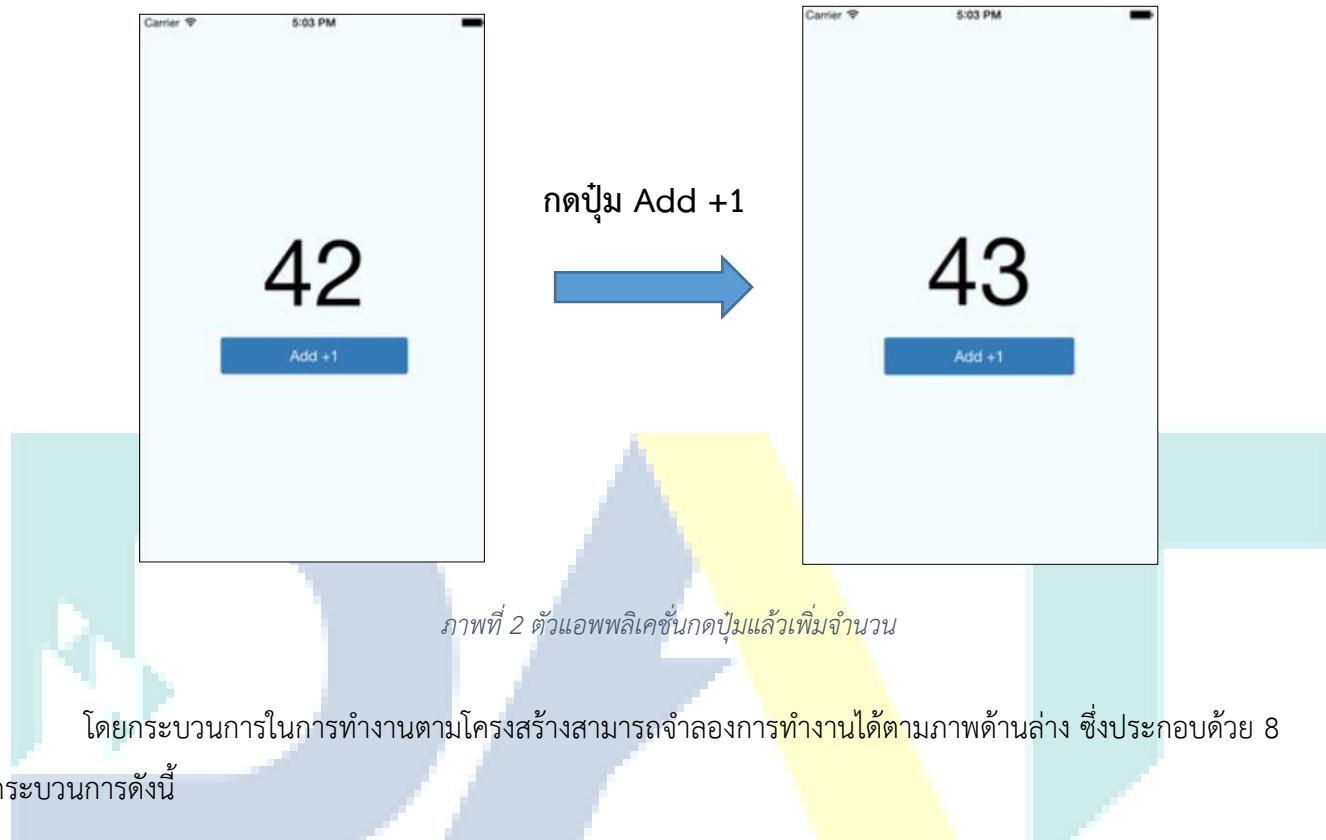
ภาพที่ 1 โครงสร้างของ React Native

1. JavaScript Core ทำหน้าที่รันโค้ดภาษา JavaScript เพื่อแสดงผลบน View ต่างๆไม่ว่าจะเป็น ข้อความ, รูปภาพ, ปุ่มหรือส่วนที่ติดต่อแสดงผลผู้ใช้งาน โดยในส่วนของระบบปฏิบัติการ iOS จะถูกเตรียมไว้ให้อยู่แล้วแต่ในระบบปฏิบัติการ Android จะต้องถูกทาง React Native รวมเข้าไปใน Package ด้วย
2. Native ทำหน้ารับข้อมูลคำสั่งในฝั่ง Native, จัดการ Event, ติดต่อเข้าถึง Resource ต่างๆภายในเครื่อง นอกจากนี้นักพัฒนายังเขียนโค้ดเพิ่มเติมเองในฝั่งนี้ได้
3. Bridge ทำหน้าที่สร้าง JSON ที่มีการบรรจุข้อมูลจำเพาะ(Serialized Messages) ในสื่อสารกันระหว่างฝั่ง JavaScript core และ ฝั่ง Native

Digital Academy Thailand

ตัวอย่างการทำงาน

จากตัวอย่างแอพพลิเคชั่นตามภาพด้านล่างประกอบไปด้วย Button ปุ่มกด Add +1 และ Text แสดงตัวเลข โดยมีหลักการทำงานคือเมื่อมี Event ทำการกดปุ่ม Add +1 สมาร์ทโฟนจะแสดงข้อความเป็นตัวเลขที่ถูกเพิ่มจำนวนขึ้น



ภาพที่ 3 การจำลองการทำงานตามโครงสร้าง

- ฝั่ง Native : มี Event กดปุ่ม Add+1
- ฝั่ง Native : ทำการเก็บข้อมูลตำแหน่งที่ถูกกดและลักษณะของ Event ที่เกิดขึ้นตามคำสั่งด้านล่าง

```
[_bridge enqueueJSCall:@"RCTEventEmitter.receiveTouches"  
args:@[@\"end\", @{@"x":@42, @"y":@106}]];
```

- ฝั่ง Bridge : ทำการสร้างข้อมูลในรูปแบบ JSON แล้วส่งไปยังฝั่ง JavaScript

```
[`EventEmitter`, `receiveTouches`, [`end`, {`x`:42, `y`:106}]]
```

- ฝั่ง JavaScript : ทำการประมวลผลจาก Event ที่ได้รับคู่กับคำสั่งที่เขียนไว้คือเพิ่มค่าข้อมูล โดยคำสั่งที่ถูกส่งมาจะถูกฝั่ง JavaScript เรียกใช้งานตามตัวอย่างด้านล่าง

```
Call(`EventEmitter`, `receiveTouches`, [`end`, {`x`:42, `y`:106}])
```

- ฝั่ง JavaScript : หลังจากประมวลผลเสร็จฝั่ง JavaScript จะสร้างคำสั่งเพื่ออัพเดตผลลัพธ์กลับไปยังฝั่ง Native อีกรอบโดยมีลักษณะตัวอย่างด้านล่าง

```
[UIManager updateView:18 props:@{@\"text\":@\"43\"}]  
[DataManager query:@\"post\" url:@\"http://...\"]
```

- ฝั่ง Bridge : ทำการสร้างข้อมูลในรูปแบบ JSON แล้วส่งไปยังฝั่ง Native

```
[ [UIManager updateView:18 props:@{@\"text\":@\"43\"}],  
[DataManager query:@\"post\" url:@\"http://...\"]]
```

- ฝั่ง Native : เมื่อได้รับคำสั่งทำการประมวลผลคำสั่งที่ได้รับ

```
[UIManager updateView:18 props:@{@\"text\":@\"43\"}]  
[DataManager query:@\"post\" url:@\"http://...\"]
```

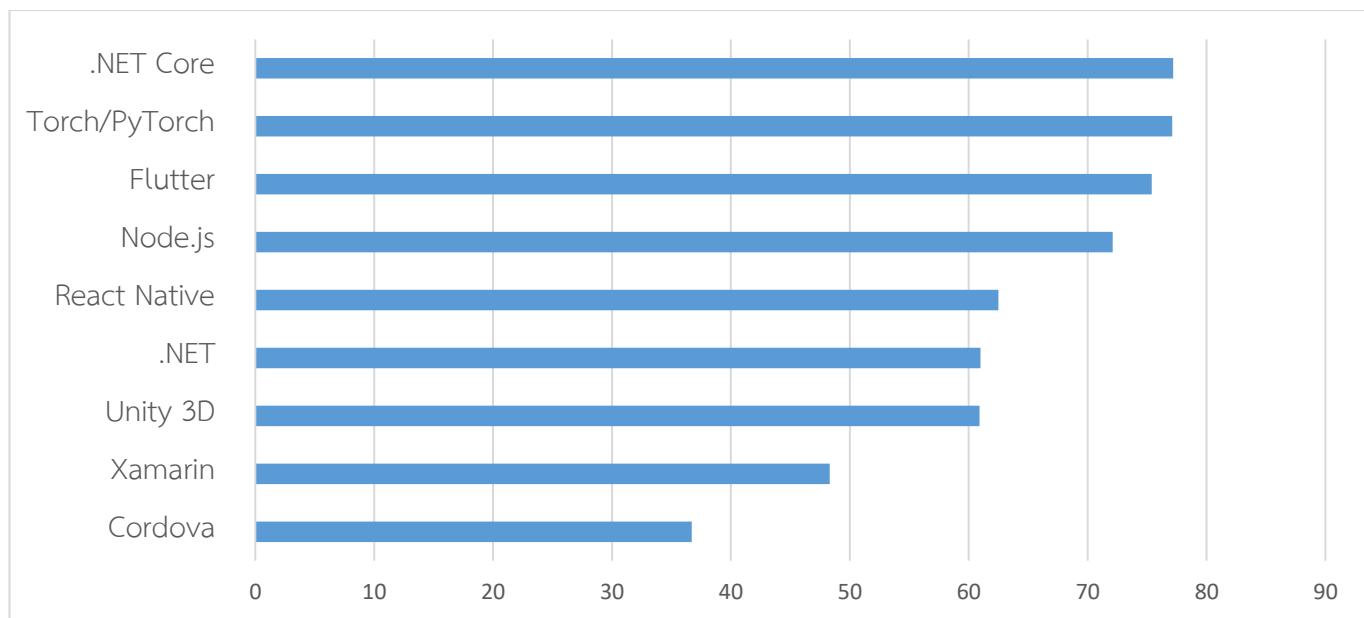
- ฝั่ง Native : ทำการ Update text เป็น 43

Digital Academy Thailand

จากตัวอย่างการทำงานข้างต้นตามโครงสร้างของ react native จะมีการรับส่งคำสั่งระหว่าง 3 ส่วนนี้ซึ่งคำสั่งนั้นจะขึ้นอยู่กับลักษณะของ Event ข้อมูลจากฝั่ง Native และผลลัพธ์จากการคำนวณในฝั่ง JavaScript โดยมี Bridge เป็นตัวแปลงรูปแบบข้อมูลให้เหมาะสมระหว่างสองฝั่งนี้ ซึ่งนักพัฒนาไม่จำเป็นจะต้องเข้าไปเขียนหรือจัดการในส่วนนี้

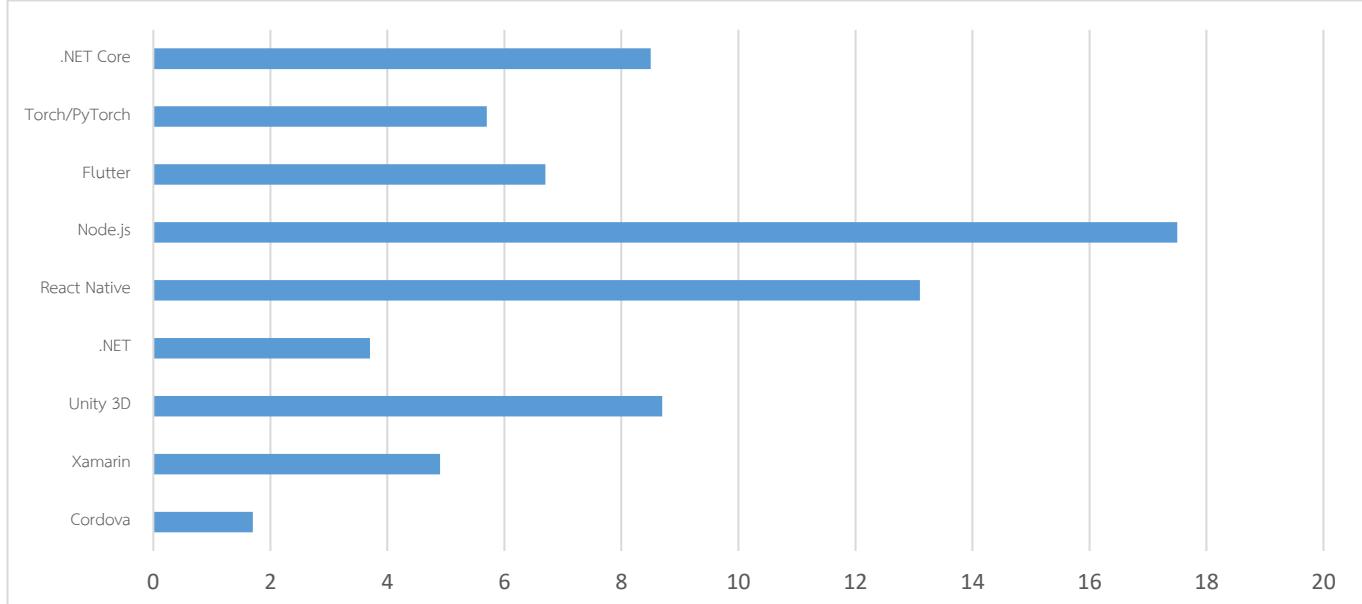
ความนิยม

ในช่วงหลายปีที่ผ่านมาได้มีแพลตฟอร์มในการพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟนมากมายทั้งแบบที่เป็น native platform, cross platform หรือแบบที่เป็น Hybrid platform ซึ่งแต่ละแบบจะมีเครื่องในเลือกมากมาย จากผลสำรวจ framework and tools ที่ใช้ในการพัฒนาโปรแกรมที่กำลังได้รับความนิยมในเวปไซต์ stackoverflow ตามภาพด้านล่าง ในส่วนของเครื่องมือที่ใช้ในการพัฒนาแอพพลิเคชั่นมีดังนี้ Flutter กำลังเป็นที่นิยมมากสุดและตามมาด้วย React-Native ทั้งนี้หากลองพิจารณาแอพพลิเคชั่นที่ถูกพัฒนาด้วย Flutter และให้บริการบน Play store และ App Store นั้น แทบจะไม่มีเลยด้วยซ้ำแต่ในขณะที่ React-Native นั้นมีแอพพลิเคชั่นที่ประสบความสำเร็จและให้บริการจริงมากกว่า



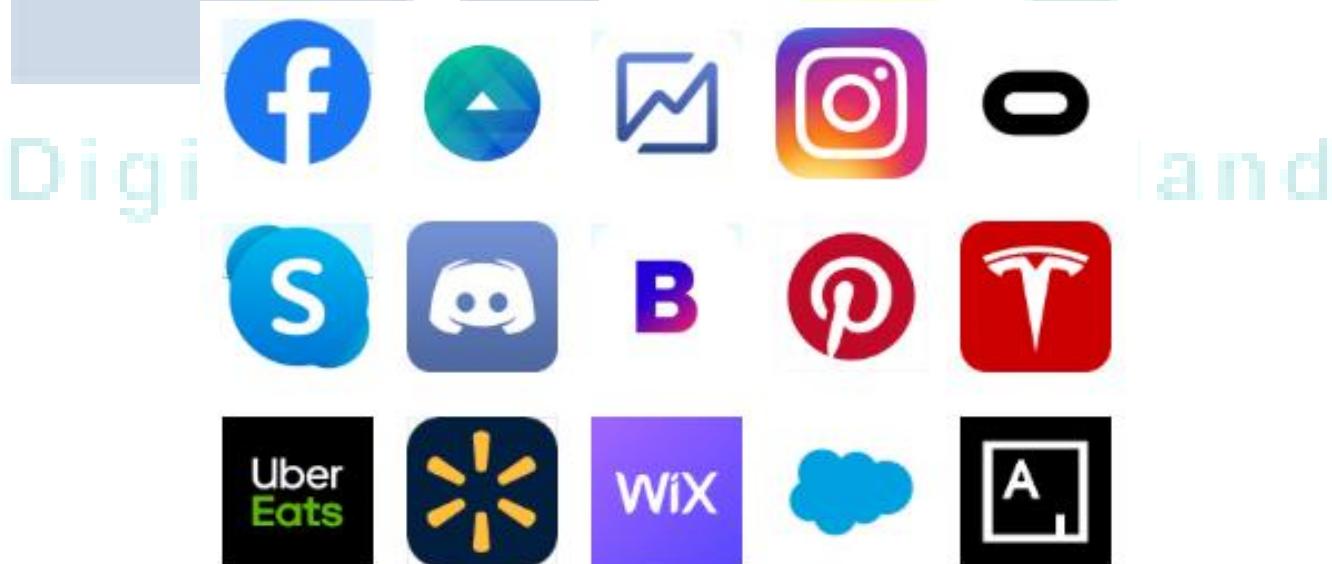
ภาพที่ 4 ผลสำรวจ framework and tools ที่กำลังได้รับความนิยมในปัจจุบัน

Digital Academy Thailand
ตำแหน่งงานนักพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟนในปัจจุบันนั้นมีความต้องการสูงในทุกๆ ประเทศซึ่งทาง stackoverflow ได้สำรวจงานในด้านนี้ซึ่งผลสำรวจตามภาพด้านล่างนั้นแสดงให้เห็นว่า React-native นั้นมีความต้องการสูงสุดเมื่อเทียบกับความต้องนักพัฒนาแอพพลิเคชั่นบนเครื่องมืออื่นๆ ซึ่งในประเทศไทยเองก็มีการเปิดรับนักพัฒนาที่มีความรู้ความสามารถในการพัฒนาด้วย React-native นี้เข่นเดียวกัน



ภาพที่ 5 ผลสำรวจ framework and tools ที่กำลังได้รับความนิยมในปัจจุบัน

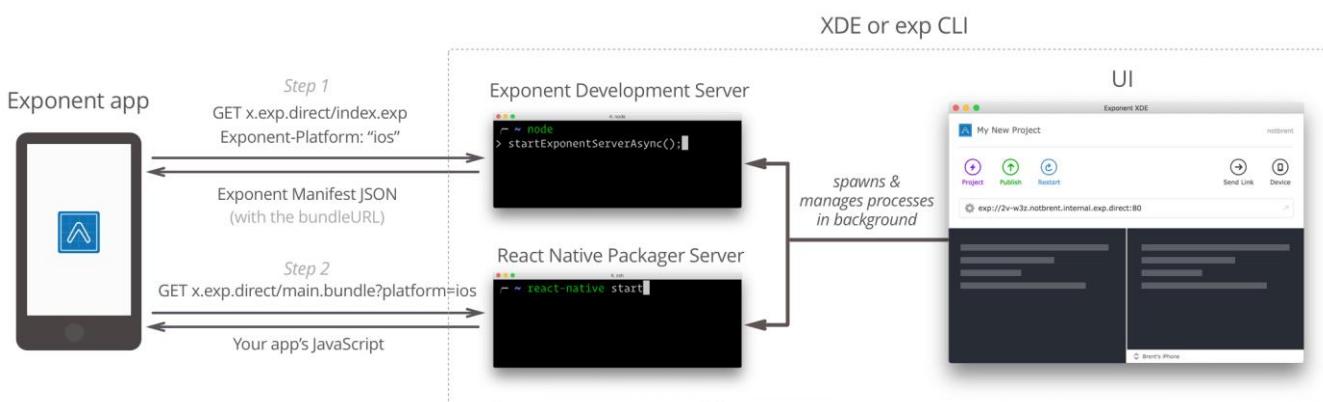
แอพพลิเคชันที่ถูกพัฒนาด้วย react native ที่ให้บริการอยู่ในปัจจุบันทั้งใน play store และ App Store นั้นมีอยู่มากมายหลากหลายตัวอย่างแอพพลิเคชันที่เป็นที่รู้จักทั่วโลกได้แก่ Facebook, Instagram, Baidu, Tesla, Wix เป็นต้น อีกมากมายตัวอย่างตามรูปด้านล่าง จะเห็นได้ว่า react native มีความน่าเชื่อถือและมีความน่าเชื่อถือมีแอพพลิเคชันที่ประสบความสำเร็จมากmany ที่ถูกพัฒนาด้วยเครื่องมือตัวนี้ฉะนั้นก็สามารถมั่นใจได้ในการเลือกใช้ React native เพื่อพัฒนาแอพพลิเคชันบนสมาร์ทโฟนจะต้องประสบความสำเร็จแน่นอน



ภาพที่ 6 ตัวอย่างแอพพลิเคชันที่ถูกพัฒนาด้วย React-native

Expo คืออะไร

Expo หรือ Exponent เป็น SDK ชุดหนึ่งที่เข้ามาช่วยให้การพัฒนา React-Native แอพพลิเคชันบนสมาร์ตโฟน ซึ่งมีการจัดการสิ่งต่าง ๆ ที่จำเป็นในการทำงานและเผยแพร่แอพพลิเคชันให้นักพัฒนาโดยไม่ต้องเข้าจัดการเขียน Native Module แต่อย่างใด อีดี้ทั้งยังมีชุดคำสั่งพร้อมต่อการใช้งานโดยที่นักพัฒนาไม่จำเป็นจะต้องเขียนเพิ่มเติม ซึ่งหัวใจสำคัญคือ XDE (The Expo Development Environment) ตามภาพด้านล่าง โดยจะประกอบได้ด้วย 3 ส่วนคือ



ภาพที่ 7 โครงสร้างของ Expo Development Environment

1. Exponent Development Server คือ Server ที่ใช้ในการสื่อสารระหว่าง Exponent app (แอพพลิเคชันของ Expo ที่ติดตั้งบนสมาร์ตโฟนเพื่อจำลองการทำงาน) โดยมีวัตถุประสงค์ในการรับส่งคำสั่งและค่าต่างๆที่อยู่ภายในไฟล์ Exponent Manifest
2. React Native Package Server คือ Server ที่ใช้จัดการ JavaScript package โดยมีวัตถุประสงค์หลักสองอย่างคือ
 - a. ทำการ compile ไฟล์ JavaScript ที่เขียนให้ร่วมเป็นไฟล์เดียวและทำการแปลงให้เหมาะสมกับเครื่องสมาร์ตโฟนปลายทาง อย่างที่เราทราบกันดีว่าสมาร์ตโฟนในท้องตลาดสามารถแบ่งตามระบบปฏิบัติการได้เป็นสองประเภทใหญ่คือ Android และ iOS :ซึ่งทั้งสองประเภทนี้มี JavaScript engine ที่ต่างกันจึงจะต้องมีการแปลง JSX(JavaScript Extension) ซึ่งเป็นภาษา JavaScript ที่ถูกพัฒนาเพิ่มเติมโดย Facebook เพื่อให้ง่ายต่อการพัฒนานั้นสามารถทำงานได้ปกติ
 - b. เพื่อส่งไฟล์ข้อมูลเก็บไว้ใน assets ไปยัง Exponent app ในการพัฒนา Exponent app การพัฒนาแอพพลิเคชันบางครั้งอาจจะมีการเรียกใช้งานรูปภาพที่เก็บไว้ในแอพพลิเคชันเอง

โดย Server นี้จะทำการรับส่งไฟล์รูปภาพเหล่านั้นที่มีการบีบอัดความละเอียดให้เหมาะสม กับเครื่องปลายทางนั้นๆ โดยที่ผู้ใช้งานไม่จำเป็นจะต้องอัพโหลดไฟล์ข้อมูลทั้งหมดไว้ในสมาร์ทโฟน

3. Web User Interface คือ ส่วนที่ใช้ในการติดต่อกับนักพัฒนา สามารถ Log การทำงานและสั่งการผ่าน web Interface ได้

จุดเด่นของ Expo

ซึ่งทาง Facebook เองได้มีการแนะนำ Expo สำหรับนักพัฒนาเดิมและผู้ที่สนใจได้ใช้งานแบบฟรี โดยมีบริการต่างๆ ที่เพียบพร้อมโดย Expo จะมีจุดเด่นหลายประการดังนี้

1. ผู้ที่เคยเขียนเว็บไซต์มาก่อนสามารถเปลี่ยนมาใช้งาน React Expo ได้ทันทีโดยไม่จำเป็นจะต้องเรียนรู้การเขียน Application ใหม่ทั้งหมด
2. สามารถเริ่มพัฒนาแอพพลิเคชันจนถึงขั้น Deploy แอพพลิเคชันนี้ขึ้นไปยัง App Store และ Play Store ได้อย่างง่ายดายโดยไม่จำเป็นต้องรู้ Native เลยก็ได้
3. ประสิทธิภาพในการทำงานในฝั่ง native ยังคงเดิม
4. สามารถใช้ JavaScript Library ที่มีอยู่อย่างมากมายได้อิสระโดยสามารถติดตั้งผ่าน Node package manager ได้
5. สามารถพัฒนา iOS แอพพลิเคชันด้วยเครื่องวินโดว์พีซีได้
6. สามารถจำลองการทำงานผ่าน Link หรือ Scan QR Code ได้สามารถอัพเดจแอพพลิเคชันให้ลูกค้าใช้โดยไม่ต้องรอขึ้น App Store หรือ Play Store ก่อน
7. ในกรณีที่นักพัฒนาจำเป็นจะต้องทำงานจำเพาะที่ Expo ไม่สามารถทำได้นักพัฒนาสามารถแปลงโปรเจคไปสู่ React Native ได้โดยง่าย

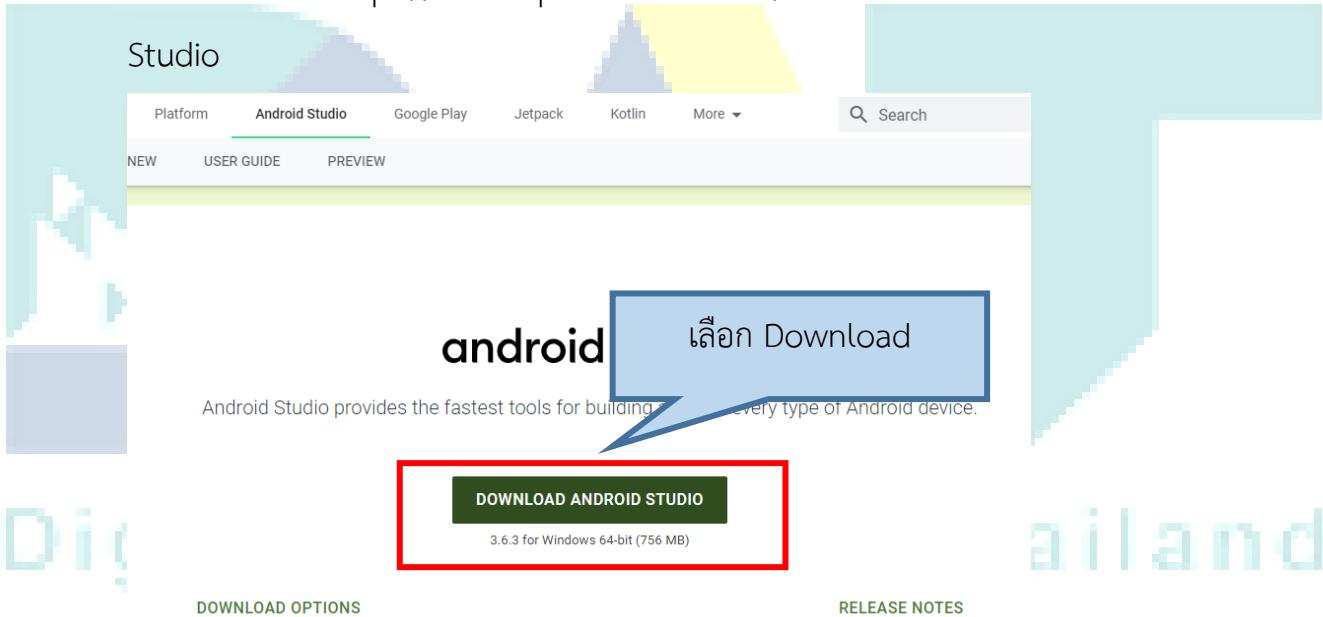
เครื่องมือในการพัฒนาและการติดตั้งเครื่องมือ

1. Android Studio

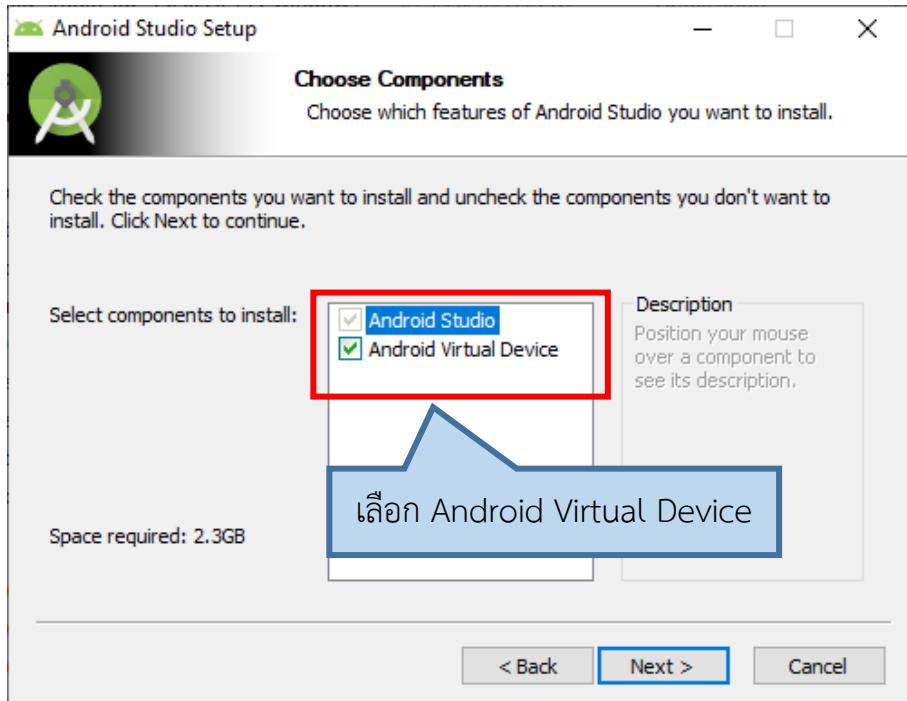
Android Studio เป็นซอฟแวร์สำหรับการทำแอปพลิเคชันบน Android แบบเนทีฟ (Native) ซึ่งเป็นการทำงานที่ออกแบบมาเฉพาะสำหรับแอปพลิเคชันบน Android เท่านั้น ทำให้แอปพลิเคชันที่ถูกพัฒนามีการทำงานที่รวดเร็วและมีประสิทธิภาพมากขึ้น ด้วยเครื่องมือในลักษณะอื่นที่จะมีการทำงานที่ซับซ้อนกว่า สำหรับเครื่องมือตัวนี้จะใช้ภาษา Java หรือ Kotlin ในการพัฒนา ซึ่ง Android Studio นี้จะมี Android Emulator ติดมาด้วยซึ่งเราจะใช้ในการจำลองการทำงานของแอปพลิเคชันของเราได้

วิธีการติดตั้ง

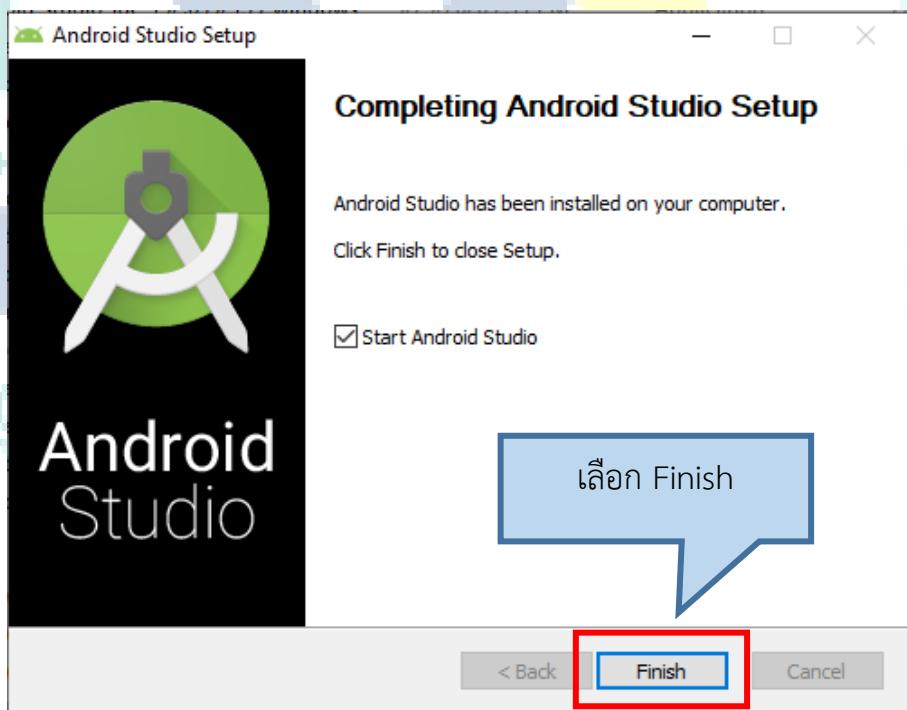
- เข้าไปที่เว็บไซต์ <https://developer.android.com/studio> และคลิก Download Android



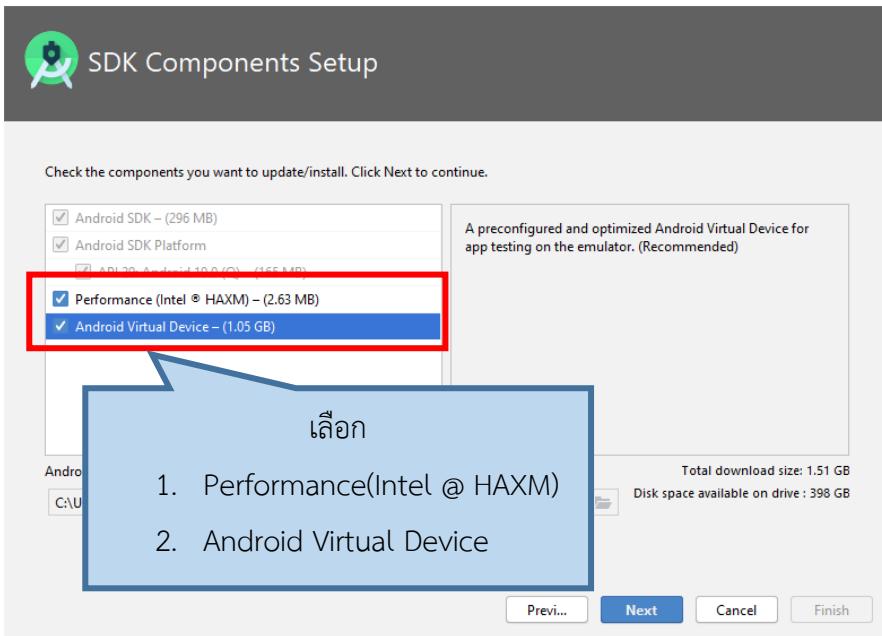
- ทำการติดตั้งโดยเลือก Android Virtual Device และกดปุ่ม Next



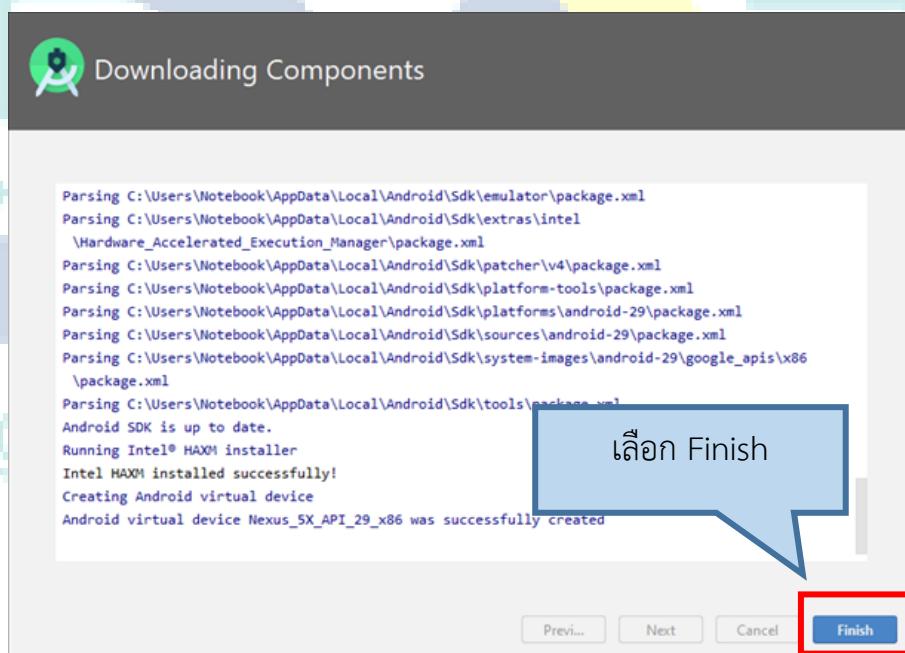
3. รอสักครู่จนกว่าติดตั้งโปรแกรมเสร็จ จากนั้นกดปุ่ม Finish



4. จากนั้น Android Studio จะให้ทำการตั้งค่าเพิ่มเติมในส่วนของ SDK Components Setup ให้เลือก Performance(Intel @ HAXM) และ Android Virtual Device



5. ระบบจะทำการ Download และติดตั้ง package เพิ่มเติมให้รอจนกว่าจะเสร็จสิ้นแล้วกด Finish

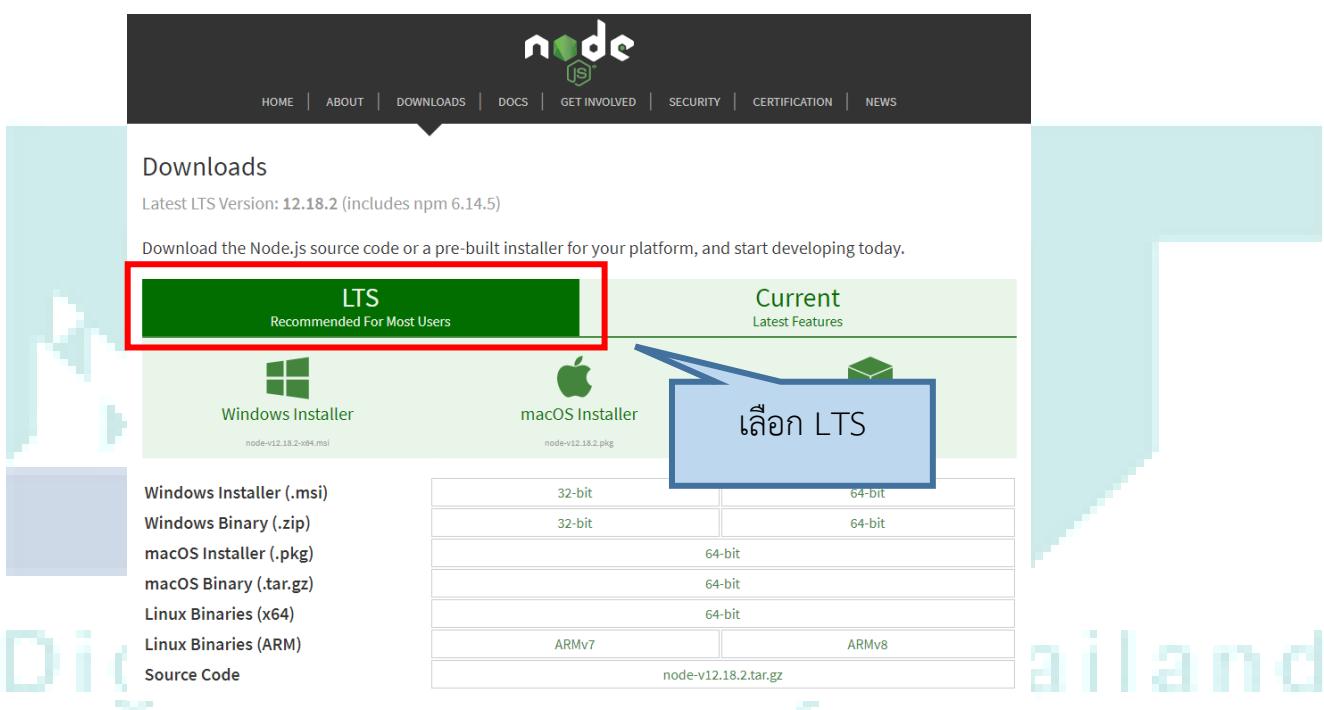


2. NodeJS

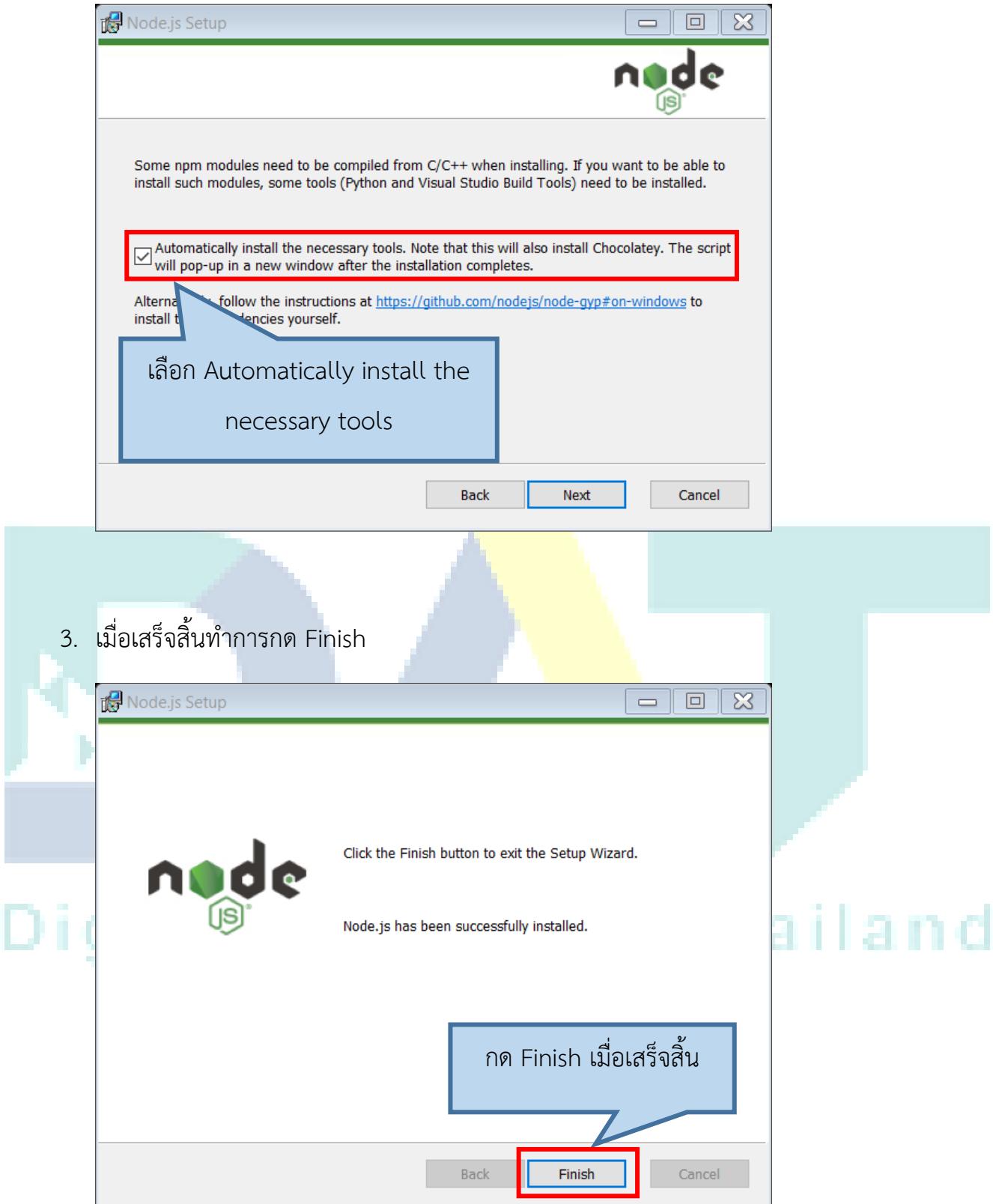
NodeJS คือ Java script runtime ที่สามารถทำงานในรูปแบบของ back-end ที่จำเป็นจะต้องใช้ในการพัฒนา Back-end service และแอพพลิเคชันต่างๆรวมถึงแอพพลิเคชันบนมือถือด้วยและใน Node.js จะมีเครื่องมือที่สำคัญอีกอย่างคือ Node package manager ที่ใช้ในการติดตั้ง dependency และ package ที่จำเป็นในการพัฒนาแอพพลิเคชันบนมือถือ

วิธีการติดตั้ง

- ไปยังเว็บไซต์และทำการ Download NodeJS ตามระบบปฏิบัติการของเครื่องคอมพิวเตอร์โดยแนะนำให้ดาวน์โหลดแบบ LTS ตามลิงค์ <https://nodejs.org/en/download>



- ทำการติดตั้งโดยในส่วนของ Tools for Native Module ให้เลือก Automatic install the necessary tools.



4. ระบบจะทำการเปิดหน้าต่าง PowerShell และติดตั้ง package ต่างๆเพิ่มเติมให้รองกว่าจะดำเนินการเสร็จ

```
Administrator: Windows PowerShell
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
  Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
(i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

WARNING: You can safely ignore errors related to missing log files when
upgrading from a version of Chocolatey less than 0.9.9.
'Batch file could not be found' is also safe to ignore.
'The system cannot find the file specified' - also safe.
chocolatey.nupkg file not installed in lib.
Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding
g...
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\tammu\OneDrive\Documents\WindowsPowerShell\Microsoft_PowerShell_profile.ps1'.
Chocolatey (choco.exe) is n
You can call choco from any
Run choco /? for a list of
You may need to shut down a
first prior to using choco
Ensuring chocolatey command are on the path
Ensuring chocolatey.nupkg is in the lib folder
Chocolatey v0.10.15
Upgrading the following packages:
python, v0.10.15 Upgrading the following packages:
By upgrading you accept licenses for the packages.
python is not installed. Installing...
Progress: Downloading python3 3.8.4... 7%
```

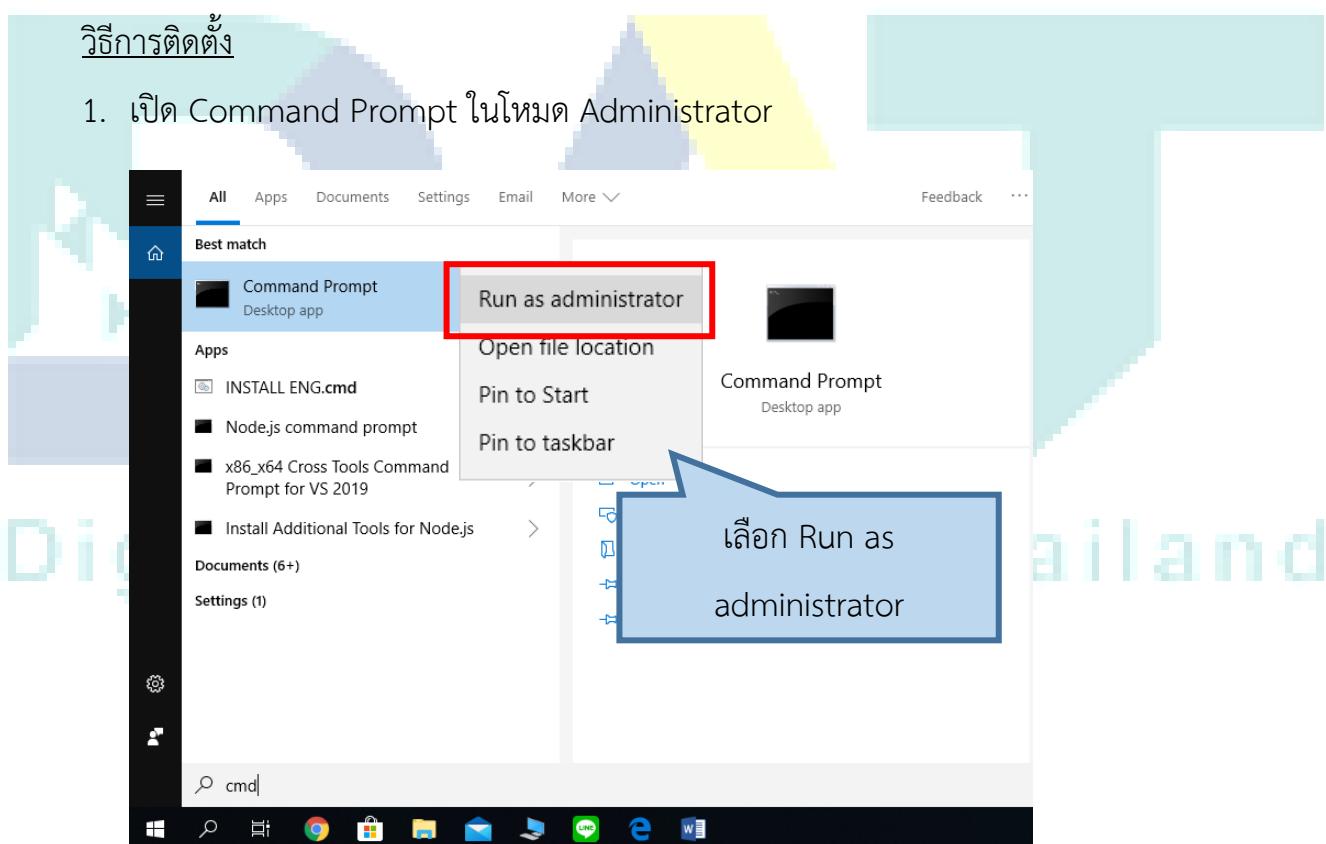
Digital Academy Thailand

3. Python2 JDK8

Python คือ ภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่ง ซึ่งถูกพัฒนาขึ้นมาโดยไม่ยึดติดกับแพลตฟอร์ม ซึ่งหมายความว่าอาจมีข้อสงสัยว่าเหตุใดจึงต้องติดตั้งเครื่องมือในการพัฒนาแอพพลิเคชันด้วยภาษาหนึ่งด้วยซึ่งทาง Facebook ได้ออกมาชี้แจงว่า React-native ได้ถูกสร้างด้วย Python2 เกือบทั้งหมด เพราะฉนั้นทาง Facebook จึงแนะนำให้ทำการติดตั้งด้วย

JDK8 หรือ Java Development Kit เป็นชุดคำสั่งในการพัฒนาโปรแกรมด้วยภาษาจาวา ซึ่งชุดพัฒนาโปรแกรม JDK นี้จะมีส่วนสำคัญในการสร้างและ Build แอพพลิเคชันที่รันบนระบบปฏิบัติการ Android

ในการติดตั้ง Package ทั้งคู่ เราสามารถติดตั้งพร้อมกันได้ผ่าน Chocolaty ที่ถูกติดตั้งมาพร้อมกับ NodeJS ที่เราติดตั้งมาก่อนหน้านี้



2. พิมพ์คำสั่งด้านล่างจากนั้นกด Enter

```
choco install -y python2 jdk8
```

```
C:\Users\tammu>choco install -y python2 jdk8
Chocolatey v0.10.15
2 validations performed. 1 success(es), 1 warning(s), and 0 error(s).

Validation Warnings:
- A pending system reboot has been detected, however, this is being ignored due to the current Chocolatey configuration. If you want to halt when this occurs, use:
  choco feature enable -name=stop-on-validation-warning
  or pass the option --exit-on-validation-failure.

Chocolatey detected you are not running from an elevated shell (cmd/powershell).
You may experience errors - you must have admin rights. Only advanced users should run choco w/out an elevated shell. When you open the command shell, you should ensure that you do so with "Run as Administrator" selected. If you are attempting to use Chocolatey in a non-administrator setting, you must select a different location other than the default install location. See
https://chocolatey.org/install#non-administrative-install for details.

For the question below, you have 20 seconds to make a selection.

Do you want to continue?([Y]es/[N]o):
```



Digital Academy Thailand

4. Expo CLI

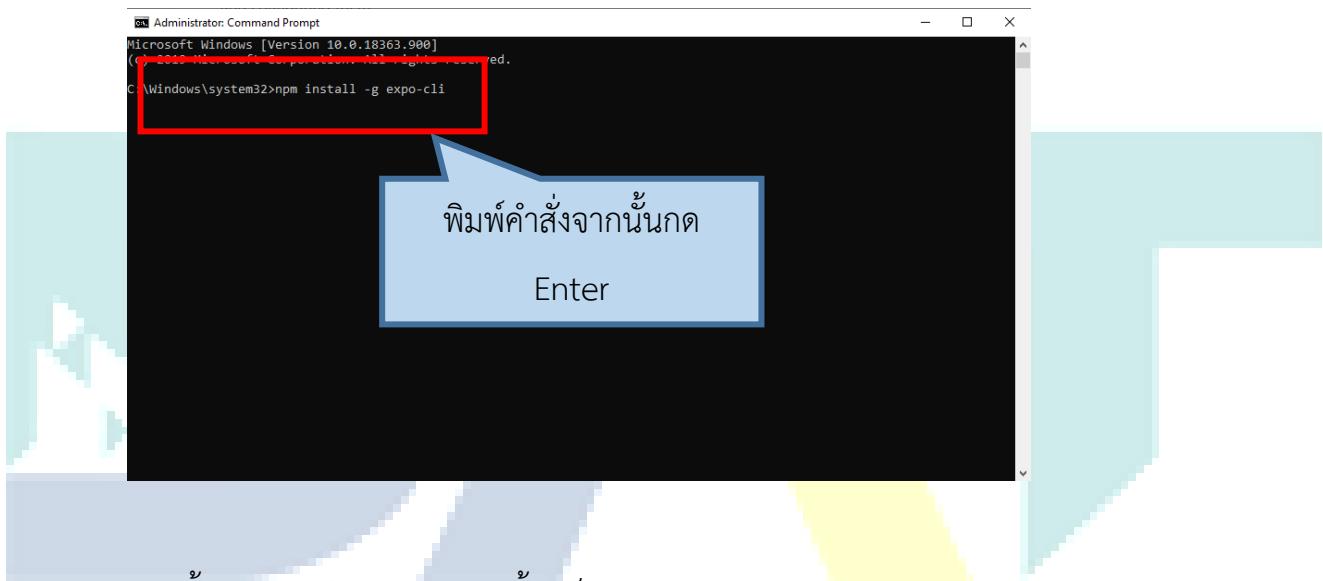
Expo CLI หรือ SDK ชุดหนึ่งที่เข้ามาช่วยให้การพัฒนา React-Native แอปพลิเคชันบนสมาร์ตโฟนที่ได้กว่าเอาไว้ข้างต้นซึ่งการติดตั้งเราสามารถติดตั้งด้วย Node Package Manager ได้

วิธีการติดตั้ง

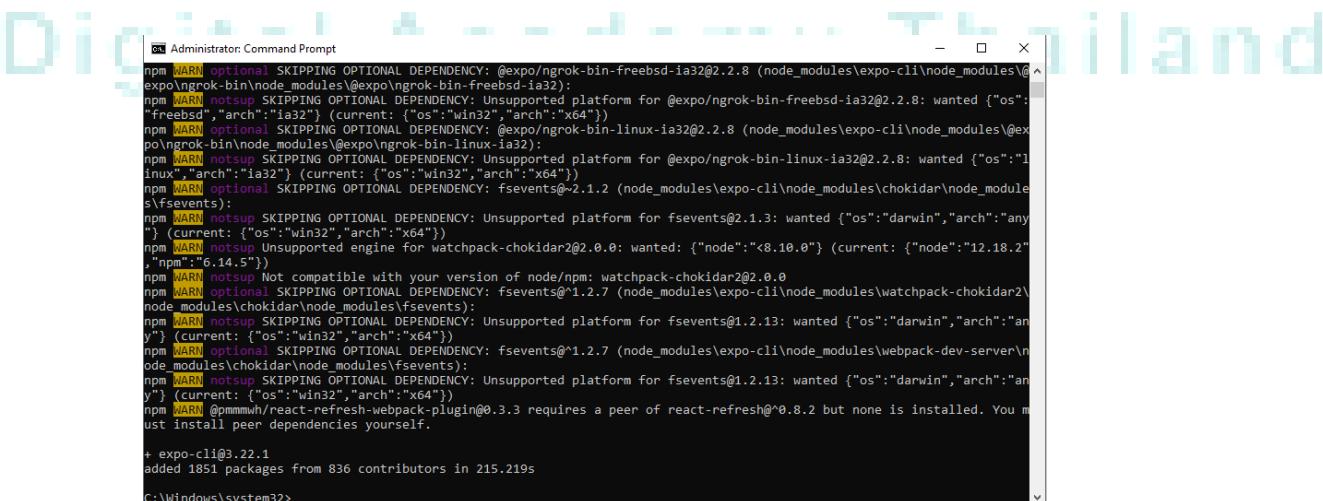
1. เปิด Command Prompt ในโหมด Administrator

2. พิมพ์คำสั่งด้านล่างลงไปแล้วกด Enter

```
npm install -g expo-cli
```



3. จากนั้น รอจนกว่าจะทำการติดตั้งเสร็จ

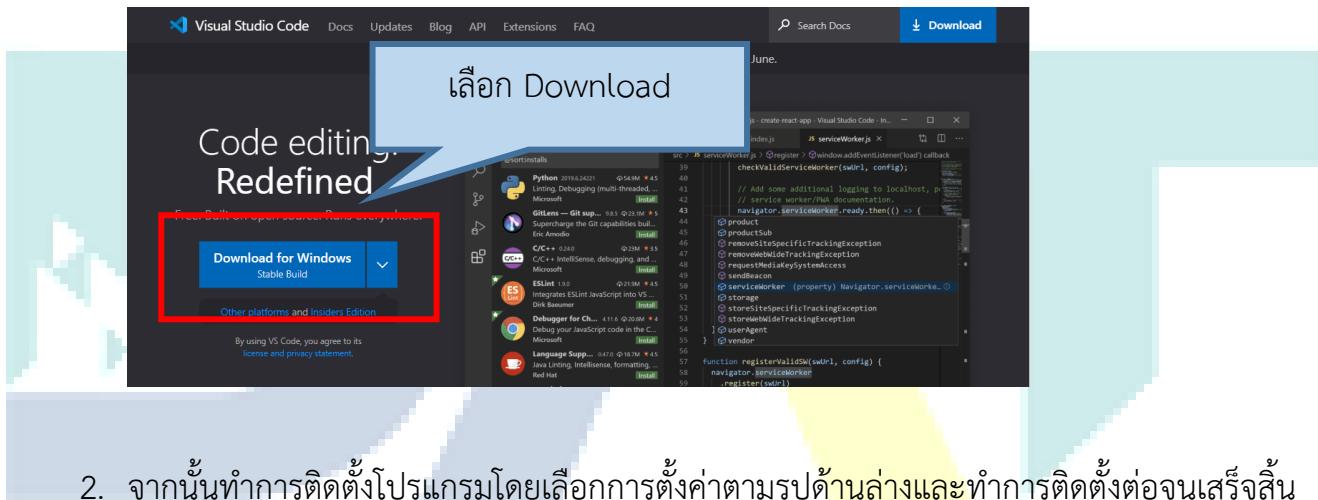


5. Visual Studio Code

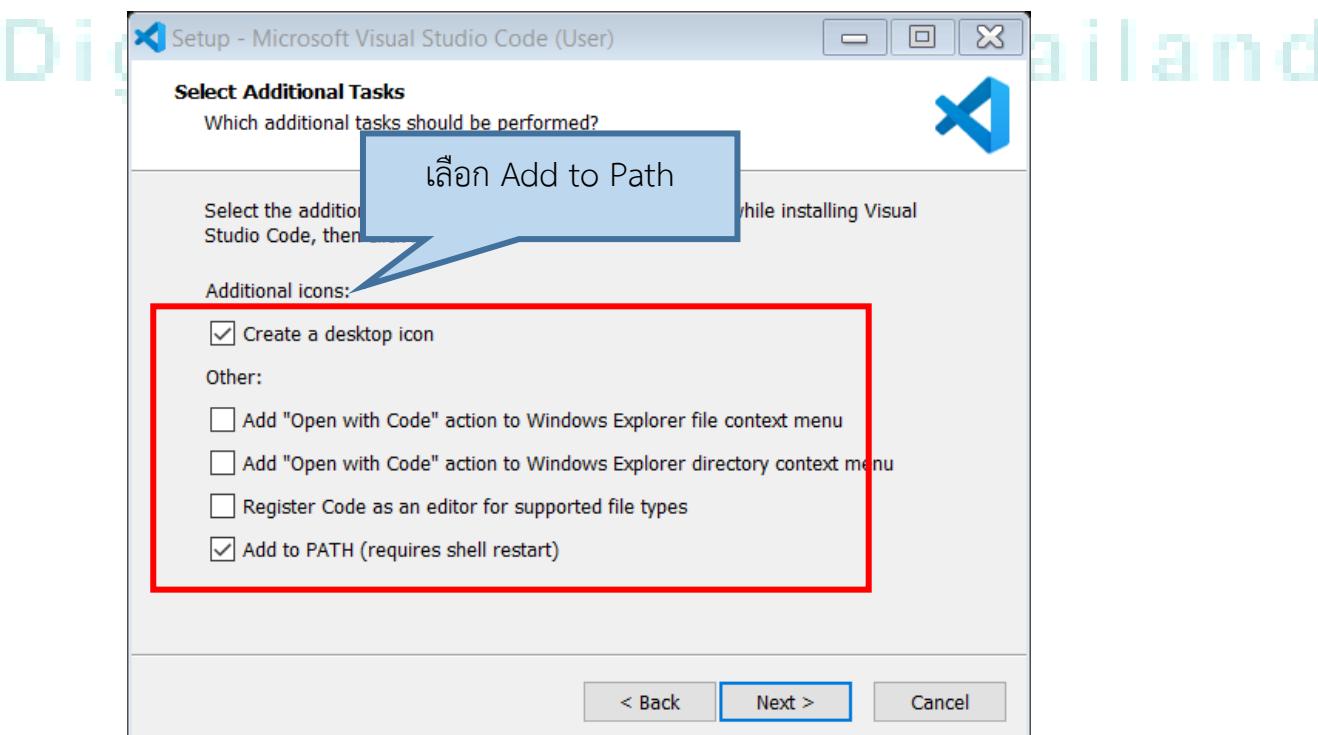
Visual Studio Code เป็นโปรแกรม Code Editor ที่ใช้ในการแก้ไขและปรับแต่งโค้ดจากค่ายไมโครซอฟท์ ที่มีการพัฒนาอุปกรณ์ในรูปแบบของ OpenSource จึงสามารถนำมาใช้งานได้แบบฟรีๆ ซึ่ง Visual Studio Code นั้นเหมาะสมสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานกับแพลตฟอร์ม มีการรองรับการใช้งานทั้งบน Windows, macOS และ Linux มีการสนับสนุนทั้งภาษา JavaScript, TypeScript และ Node.js อีกทั้งยังสามารถเรียกใช้ Command Prompt ในตัวโปรแกรมได้ทันที

วิธีการติดตั้ง

- ไปยังเว็บไซต์และทำการ Download Microsoft Visual Code ตามระบบปฏิบัติการของเครื่องคอมพิวเตอร์ตามลิงค์ <https://code.visualstudio.com/>



- จากนั้นทำการติดตั้งโปรแกรมโดยเลือกการตั้งค่าตามรูปด้านล่างและทำการติดตั้งต่อจนเสร็จสิ้น



การสร้าง Expo โปรเจค

ก่อนที่จะเริ่มพัฒนาสิ่งแรกต้องทำกีคือการสร้างโปรเจคโดยการสร้าง Project ด้วย Expo นี้พัฒนา จะต้องเข้าใจในเรื่องของ work flow หรือลักษณะของโปรเจคที่ต้องการสร้างเสียก่อน โดย Expo สามารถสร้าง Project โดยมีลักษณะที่แตกต่างกันได้ 2 ประเภทใหญ่ๆดังนี้

1. Managed workflow เป็นการสร้างโปรเจคที่มีการจัดการมีการจัดการ Package หรือ Bundle ต่างๆ. ทั้งในฝั่งของ Android และ iOS ให้ โดยที่นักพัฒนาไม่จำเป็นต้องเข้าไปจัดการเองในส่วนนี้ ซึ่งจะช่วยให้นักพัฒนาไม่จำเป็นจะต้องทำการติดตั้งโปรแกรม XCode สำหรับการพัฒนา Application ในระบบปฏิบัติการ iOS หรือ Android Studio สำหรับการพัฒนา Application ในระบบปฏิบัติการ Android เพียงแต่เขียนโปรแกรมด้วยภาษา JavaScript และการกำหนดตั้งค่าที่สำคัญในไฟล์ app.json เท่านั้น
2. Bare workflow เป็นการสร้างโปรเจคที่นักพัฒนาสามารถควบคุมทุกอย่างของโปรเจคได้และยังสามารถใช้ API หรือชุดคำสั่งต่างๆที่อยู่ใน Expo sdk ได้เช่นกัน แต่จะไม่สามารถใช้งานในส่วนของ application config หรือ app.json ที่ช่วยลดความยุ่งยากในการตั้งค่าได้ ซึ่งนักพัฒนาจะต้องทำการตั้งค่าต่างๆเองในส่วนของฝั่ง native เอง ซึ่งนักพัฒนาจะต้องมีความรู้ในการพัฒนาแบบ Native มาก่อนอีกทั้งยังต้องติดตั้งโปรแกรมที่ใช้ในการพัฒนาในฝั่ง Native อีกด้วย

การเปรียบเทียบ Workflow

Feature	Managed workflow	Bare workflow
ใช้ภาษา JavaScript/TypeScript เท่านั้น	ใช่	ไม่ใช่
สามารถใช้บริการของ Expo ในการสร้าง iOS และ Android Application	ได้	ไม่ได้
สามารถใช้ push notification ของ Expo ได้	ได้	ได้
สามารถใช้ Expo's over the air updates ได้	ได้	ได้
สามารถใช้งาน Expo client app ได้	ได้	ได้
สามารถใช้งาน Expo SDK ได้	ได้	ได้
สามารถปรับแต่ง Code ในฝั่ง Native ได้	ไม่ได้	ได้

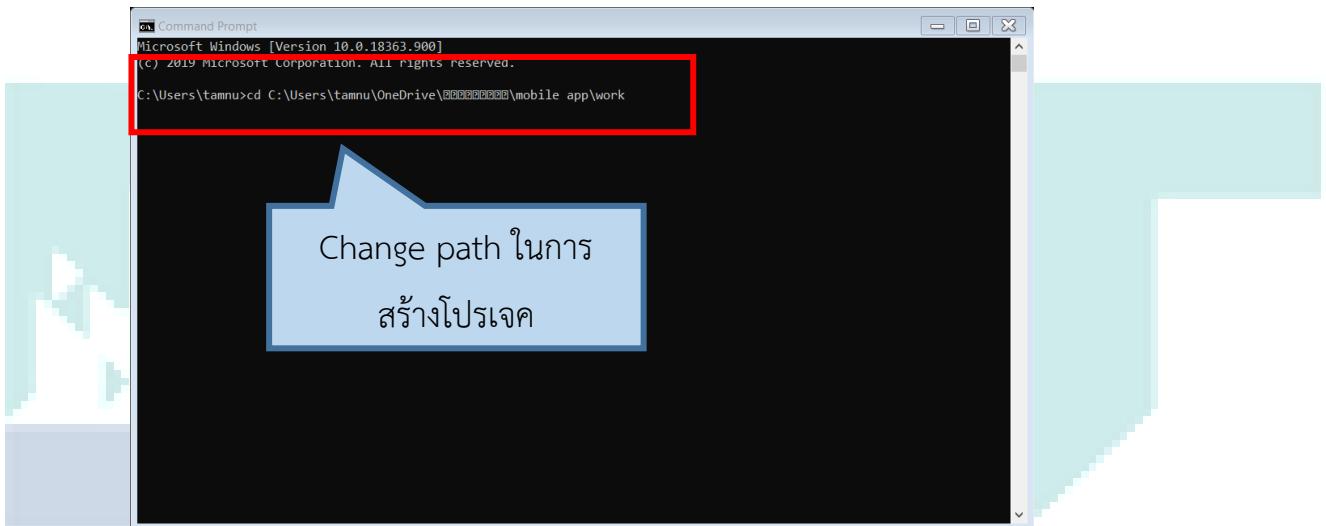
การตัดสินใจเลือก workflow การทำงานนั้น เป็นองค์ประกอบที่สำคัญจะแนะนำให้เริ่มใช้งานจะ managed work Flow ก่อน ซึ่งสะดวกและง่ายต่อการใช้งานเหมาะสมสำหรับผู้ที่กำลังเริ่มศึกษาและนักพัฒนาที่มี

ประสบการณ์แล้วก็จะช่วยลดระยะเวลาในการเขียน Code ให้สั้นขึ้น หากระหว่างการพัฒนานักพัฒนาเลือกเห็นว่า managed work flow ไม่สามารถตอบโจทย์การพัฒนาได้ทั้งหมดนักพัฒนาสามารถกลับมาเปลี่ยน Project กลับไปเป็น React Native Project ได้ ทั้งนี้สำหรับรายวิชานี้ใช้ managed workflow เป็นหลักเพื่อให้ง่ายต่อการเรียนการสอน

การสร้างโปรเจคแบบ managed work flow

1. เปิด Command Prompt ขึ้นมา
2. จากนั้นทำการ change directory ไปยังตำแหน่งที่ต้องการสร้างโปรเจค ตัวอย่าง

```
cd C:\Users\tamnu\OneDrive\เดสก์ท็อป\mobile app\work
```



3. จากนั้นพิมพ์คำสั่งในการสร้างโปรเจค
`expo init YourProjectName`
จากตัวอย่างได้สร้างโปรเจคชื่อ Hello
4. ทำการเลือกโปรเจคแบบ blank Managed workflow

```
C:\ Command Prompt - expo init Hello
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\tammu>cd C:\Users\tammu\OneDrive\██████████\mobile app\work

C:\Users\tammu\OneDrive\██████████\mobile app\work>expo init Hello
? Choose a template: (Use arrow keys)
    ---- Managed workflow ----
    > blank           a minimal app as clean as an empty canvas
    blank (TypeScript) same as blank but with TypeScript configuration
    tabs (TypeScript)   several example screens and tabs using react-navigation and TypeScript
    ---- Bare workflow ----
    minimal          bare and minimal, just the essentials to get you started
    minimal (TypeScript) same as minimal but with TypeScript configuration
```

สร้างโปรเจกและเลือก workflow

5. ระบบจะทำการติดตั้ง package ให้ร้อนกว่าจะเสร็จสิ้น

```
C:\ Command Prompt
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\tammu>cd C:\Users\tammu\OneDrive\██████████\mobile app\work

C:\Users\tammu\OneDrive\██████████\mobile app\work>expo init Hello
? Choose a template: expo-template-blank

Using npm to install packages. You can pass --yarn to use Yarn instead.

Downloaded and extracted project files.
Installed JavaScript dependencies.

Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd Hello
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web

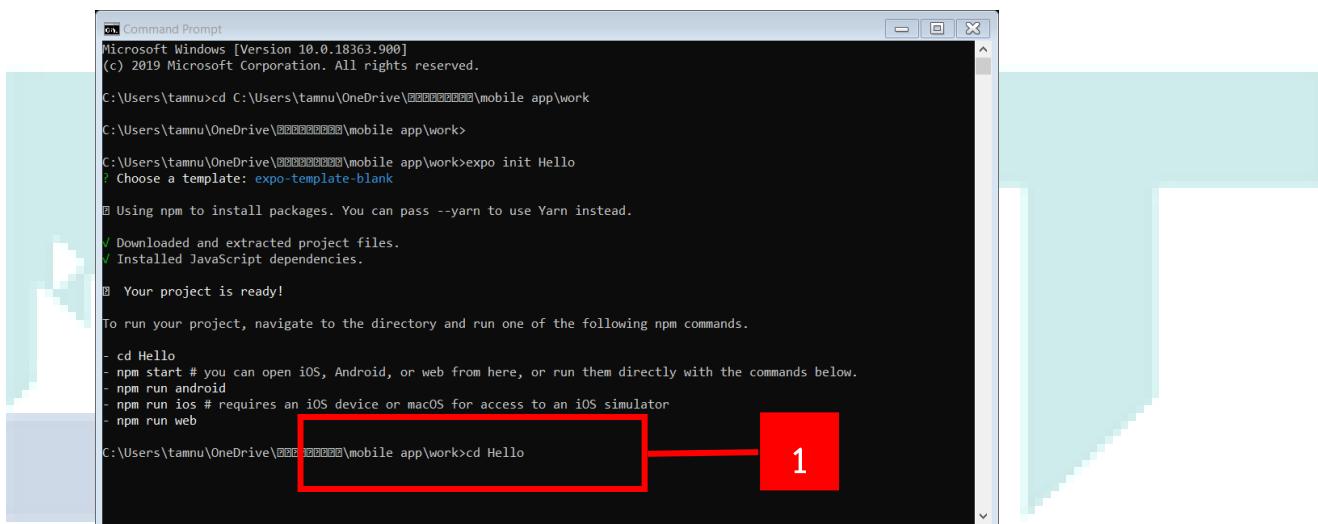
C:\Users\tammu\OneDrive\██████████\mobile app\work>
```

การทดสอบด้วย Emulator

การจำลองการทำงานด้วย Emulator(Android) นักพัฒนาจำเป็นจะต้องลงโปรแกรม Emulator เสียก่อนซึ่งโปรแกรมสำหรับจำลองการทำงานในส่วนของ Android นั้นมีมากมายให้เลือก :ซึ่งในที่นี่เราได้ทำการติดตั้งโปรแกรม Android Studio ซึ่งมีตัว Android Emulator ติดมาด้วยการจำลองการทำงานได้โดยมีวิธีการดังนี้

1. หลังจากสร้างโปรเจคเสร็จสิ้นให้ทำการ Change Directory ไปยัง Project ที่สร้าง

```
cd Hello
```



```
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\tamnu>cd C:\Users\tamnu\OneDrive\เอกสารสอน\mobile app\work
C:\Users\tamnu\OneDrive\เอกสารสอน\mobile app\work>
C:\Users\tamnu\OneDrive\เอกสารสอน\mobile app\work>expo init Hello
? Choose a template: expo-template-blank
Using npm to install packages. You can pass --yarn to use Yarn instead.
✓ Downloaded and extracted project files.
✓ Installed JavaScript dependencies.

Your project is ready!
To run your project, navigate to the directory and run one of the following npm commands.

- cd Hello
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web

C:\Users\tamnu\OneDrive\เอกสารสอน\mobile app\work>cd Hello
```

2. ทำการเปิด Server เพื่อจำลองการทำงานโดยใช้คำสั่งด้านล่าง

```
expo start หรือ npm start
```

3. ระบบจะทำการ Run Server เพื่อใช้ในการจำลองการทำงานโดยจะแสดง QR Code

```

npm
C:\Users\tamnu\OneDrive\שולחן העבודה\Hello
C:\Users\tamnu\OneDrive\שולחן העבודה\Hello>expo start
Starting Metro Bundler at C:\Users\tamnu\OneDrive\שולחן העבודה\Hello
Expo DevTools is running at http://localhost:19002
Opening DevTools in the browser... (press shift-d to disable)
Starting Metro Bundler on port 19001.

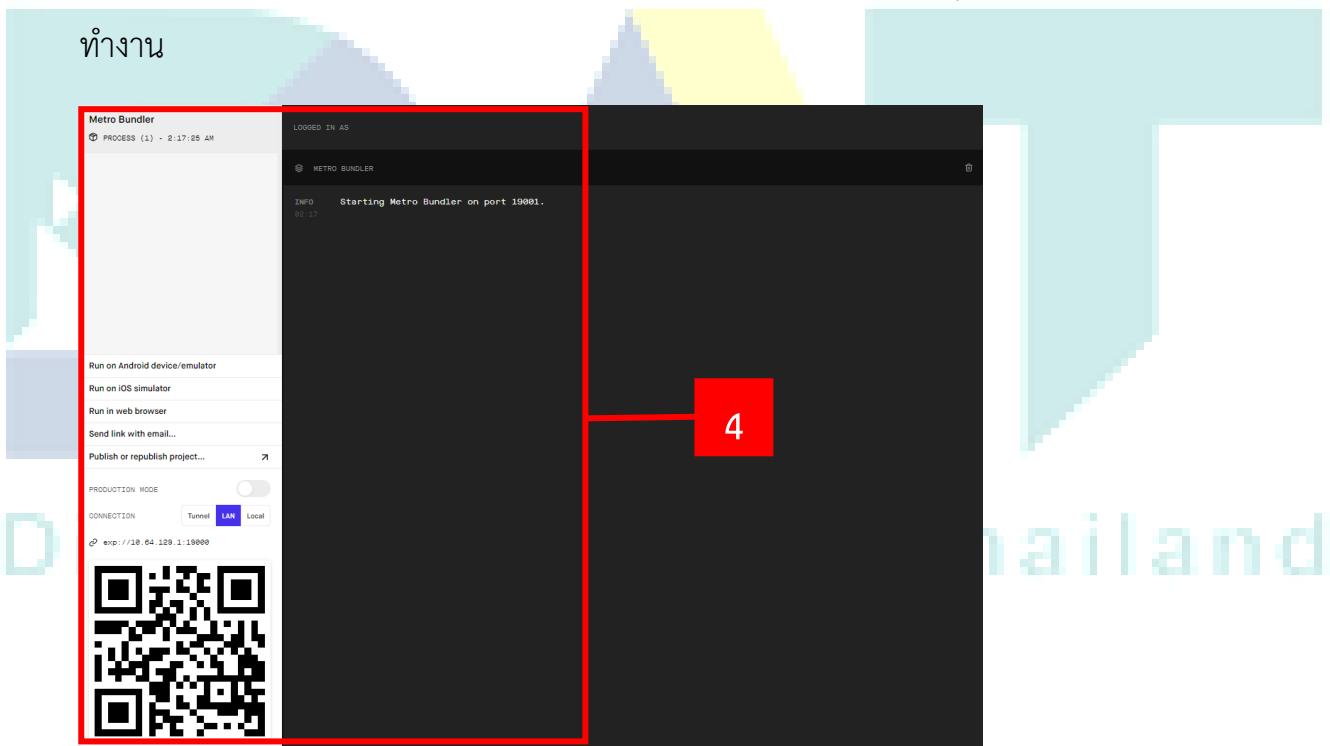
exp://10.64.129.1:19000

```

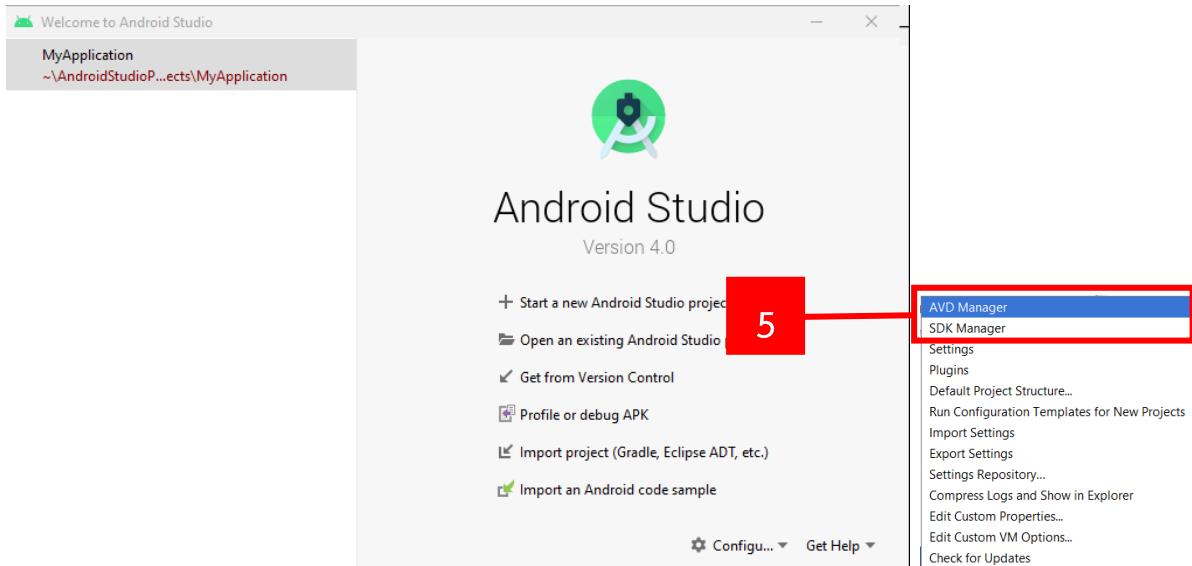
2

3

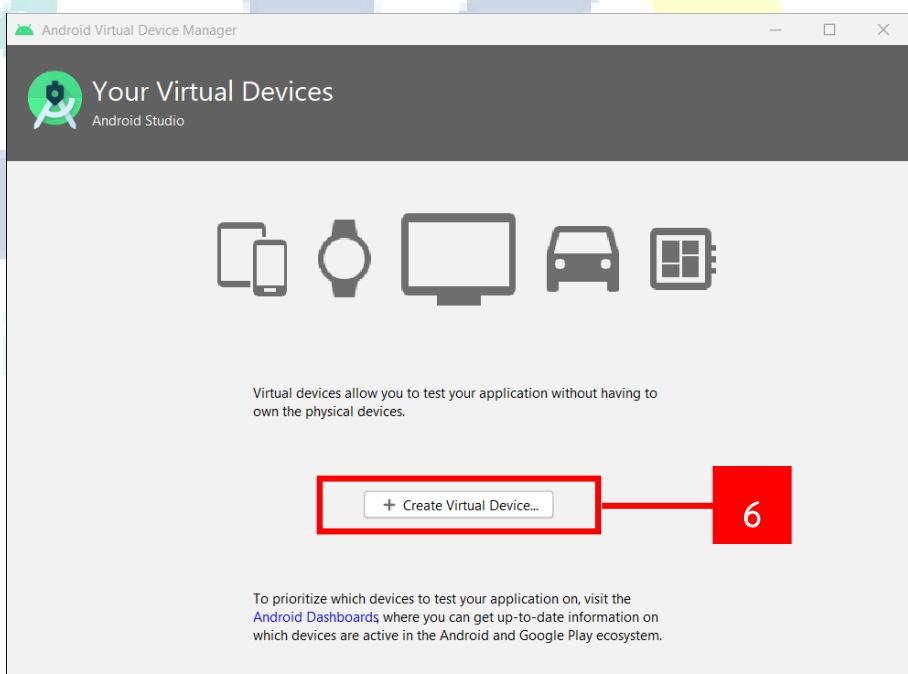
4. ระบบจะทำการเปิดเว็บเพจเพื่อรับการสั่งงานจาก User เมื่อมาก็จะดูว่า Server พร้อมที่จะทำงาน



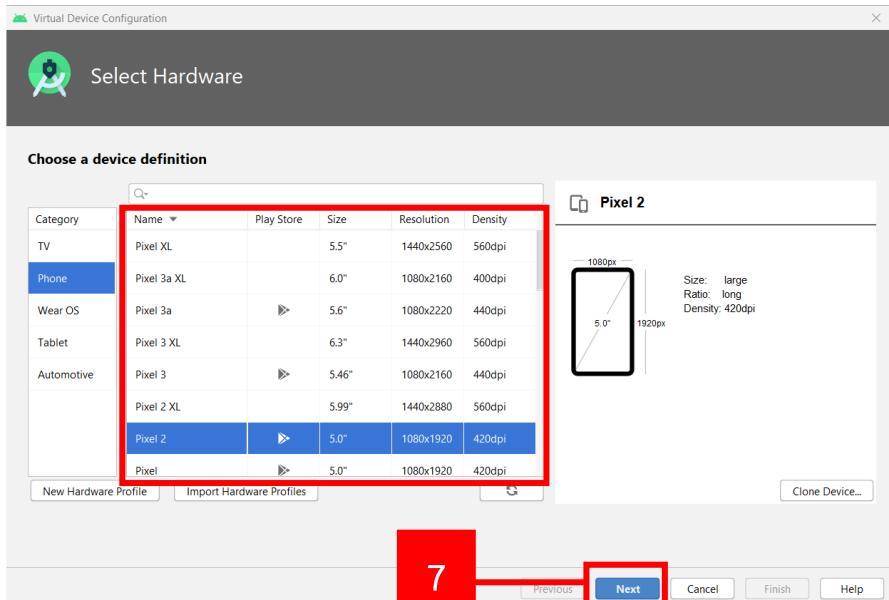
5. จากนั้นทำการรัน Android emulator โดยเปิดโปรแกรม Android Studio ขึ้นมาจากนั้นคลิก Configuration แล้วเลือก AVD Manager



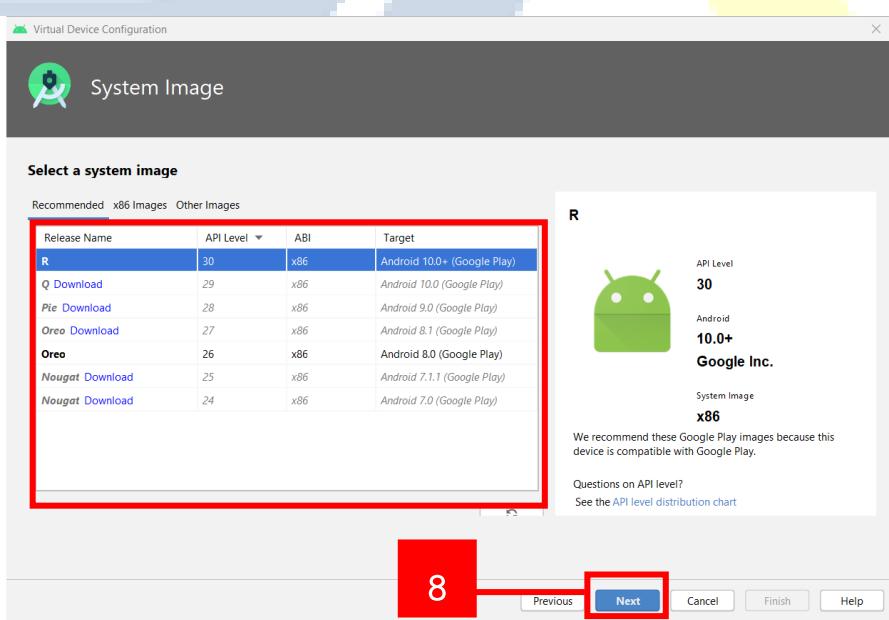
6. Android Studio จะเปิดหน้าต่างขึ้นมาเพื่อทำการสร้าง emulator จากนั้นให้เลือก create Virtual device



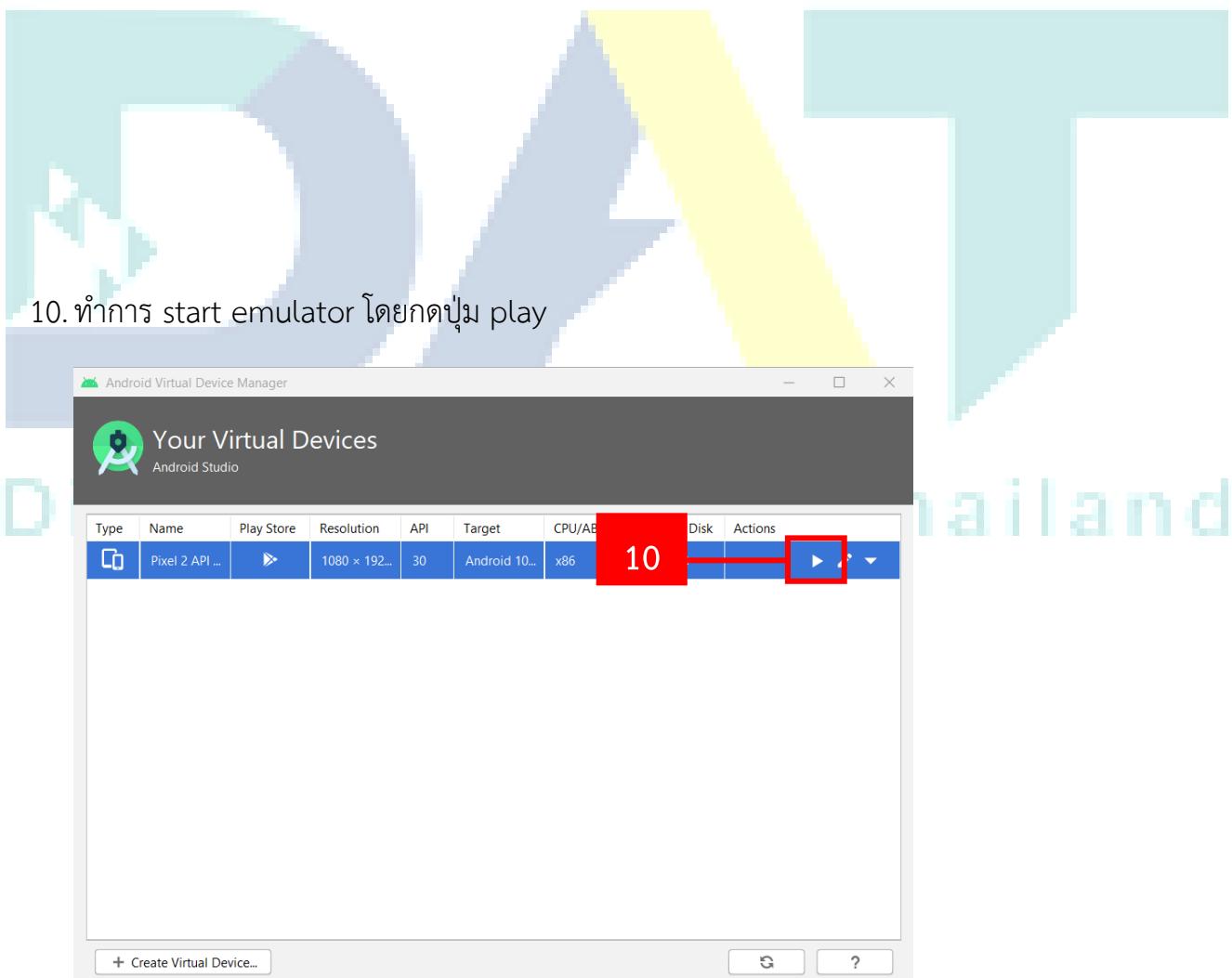
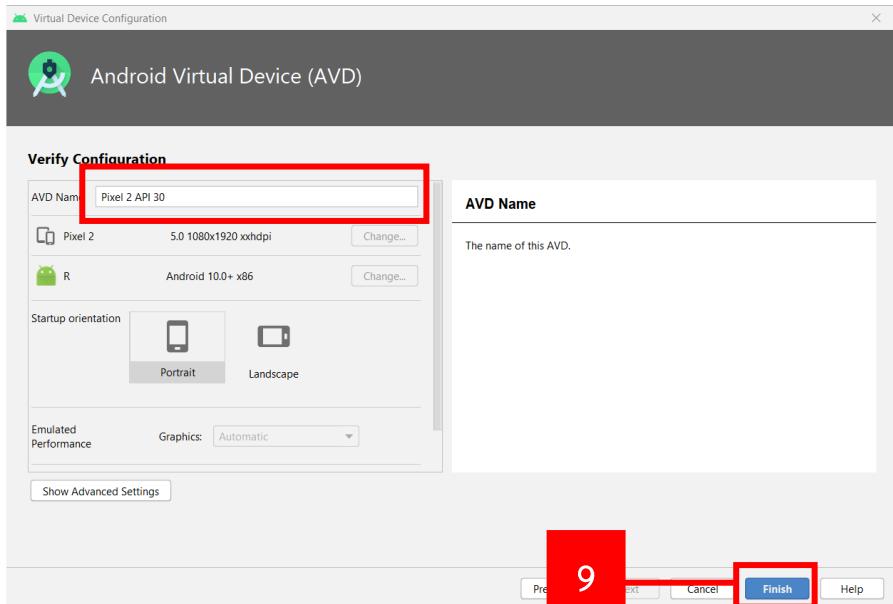
7. ทำการเลือกรุ่นโทรศัพท์มือถือที่ต้องการ จากนั้นกด next



8. ทำการเลือกเวอร์ชันของ Android ที่ต้องการ หากยังไม่ทำการดาวน์โหลดให้ทำการดาวน์โหลดเสียก่อน
จากนั้นกด next

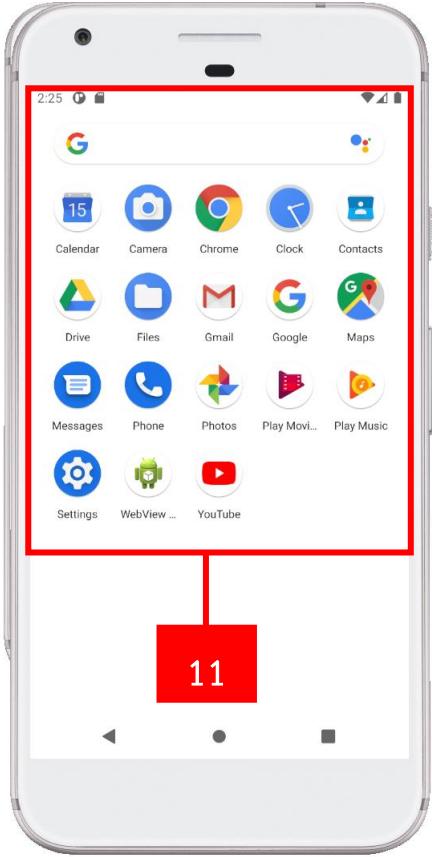


9. ทำการตั้งชื่อ emulator ตามที่ต้องการ จากนั้นกด finish



11. จากนั้นรอสักครู่ Android emulator จะปรากฏขึ้นมา

เอกสารโดย อาจารย์กរุณนวัฒน์ วาลีประโคน



12. จากนั้นทำการรัน Project ที่สร้างขึ้นโดยกลับไปยังหน้าเว็บเพจจากนั้นกดเลือก Run On Android

Device connected

Metro Bundler

PROCESS (1) - 2:17:25 AM

```
LOGGED IN AS
METRO BUNDLER
INFO Starting Metro Bundler on port 19001.
```

Run on Android device/emulator

12

Run in web browser

Send link with email...

Publish or republish project...

PRODUCTION MODE

CONNECTION Tunnel LAN Local

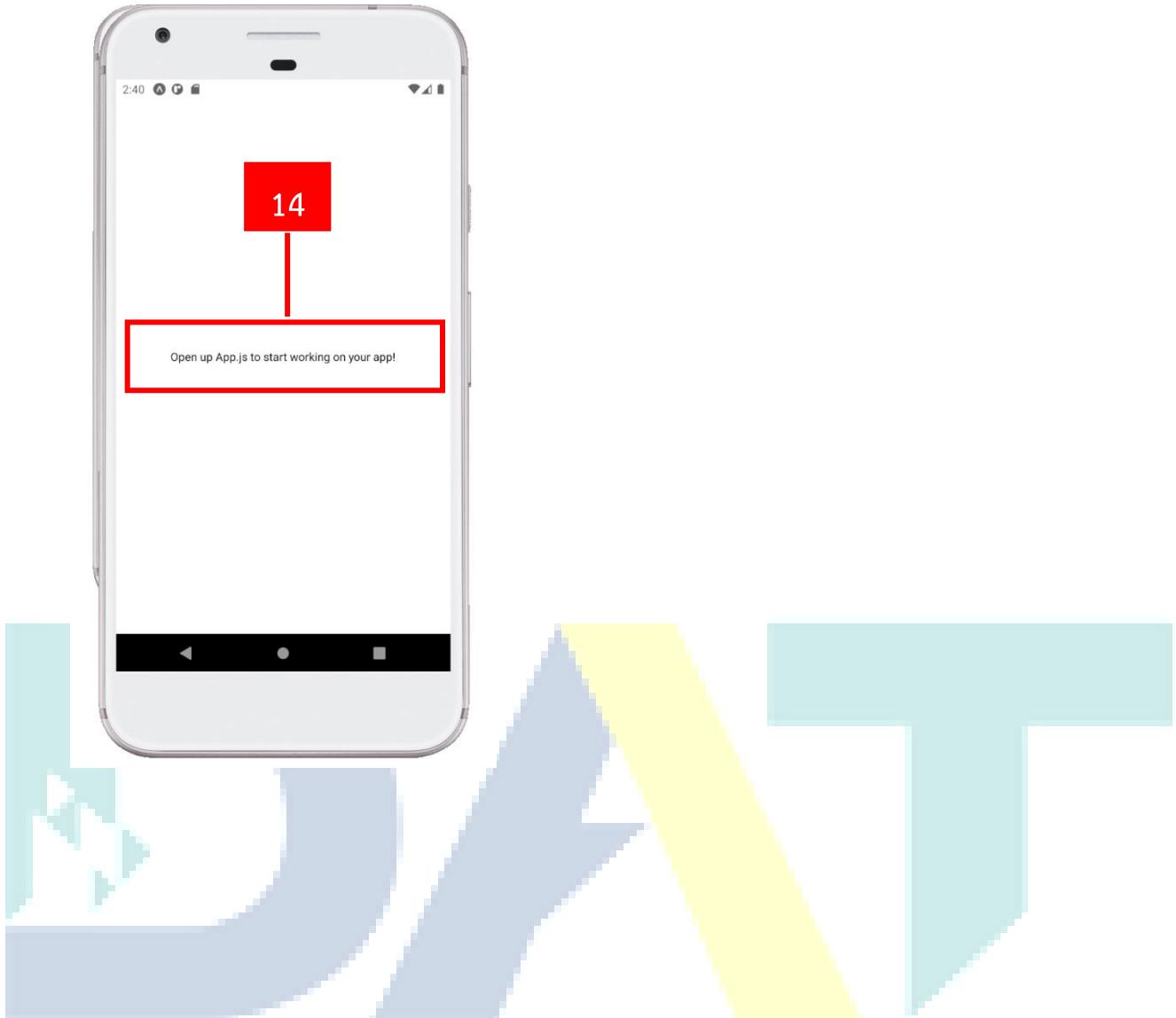
exp://10.64.129.1:19001

13. จากนั้นรอสักครู่ Server จะทำการ Download Package ต่างๆและUpload ติดตั้งลงไปยัง Android emulator



14. เมื่อเสร็จสิ้นจะแสดงข้อความอาภาพด้านล่าง

Digital Academy Thailand

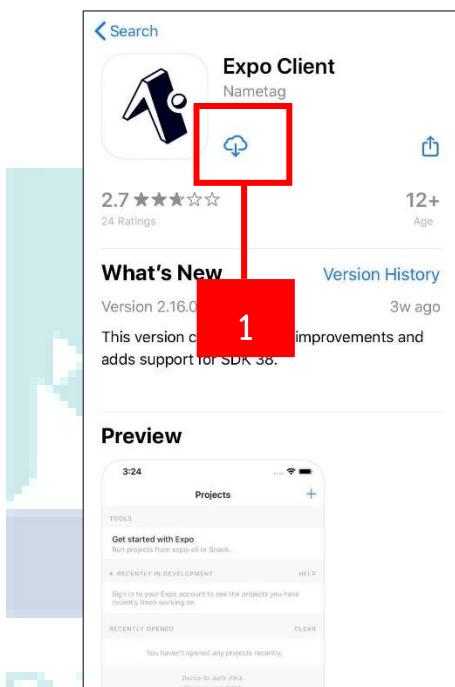


Digital Academy Thailand

การทดสอบด้วยเครื่องจริง

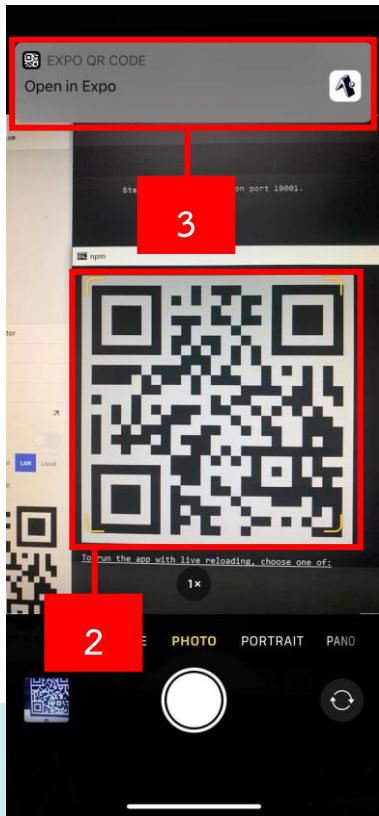
ในการทดสอบ Application ในเครื่องจริงนักพัฒนาจำเป็นจะต้องติดตั้งซอฟต์แวร์เพิ่มเติมลงบนสมาร์ทโฟนโดย Application นั้นมีชื่อว่า Expo Client สามารถดาวน์โหลดติดตั้งได้ระบบ Android และ iOS ซึ่ง Expo Client นี้จะทำหน้าที่เชื่อมต่อระหว่าง Expo Server คดยรับส่ง Package ของ Application ที่ถูกเขียนขึ้นโดยขั้นตอนการทดสอบโดยเครื่องจริงนั้นมีดังนี้

- ทำการ Download และติดตั้ง Expo Client ลงบนสมาร์ทโฟน

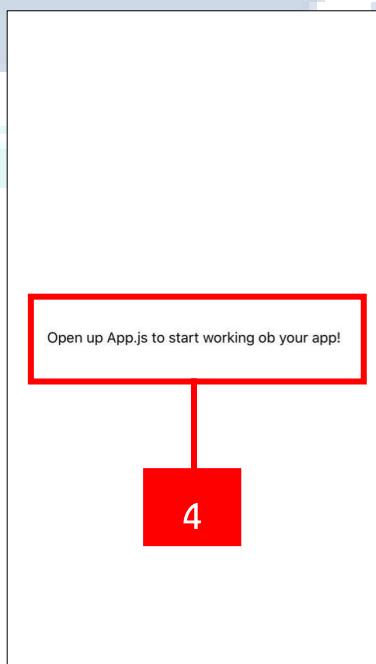


Digital Academy Thailand

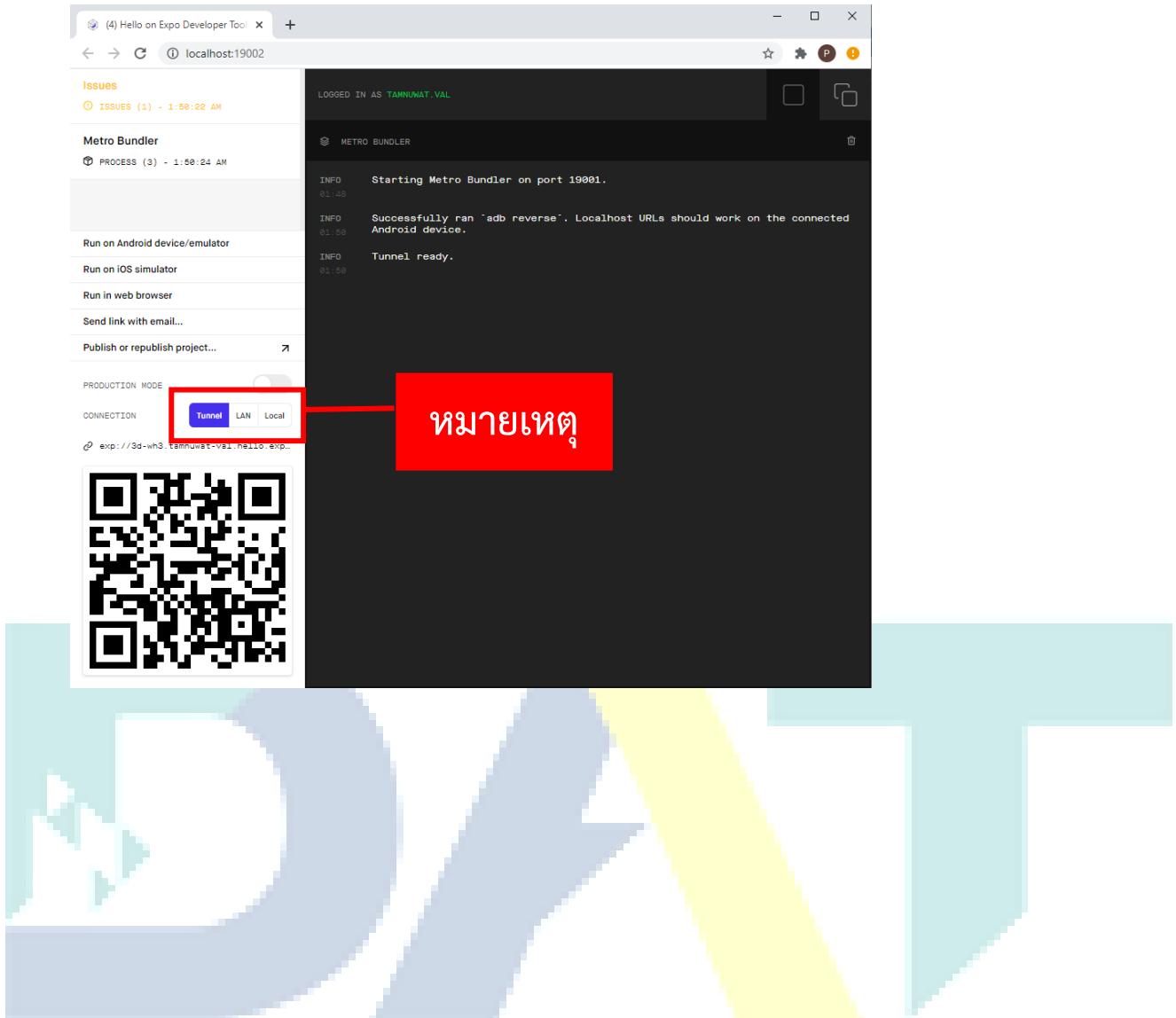
- เปิดโปรแกรมถ่ายภาพในโทรศัพท์มือถือแล้วทำการสแกน QR Code บนหน้า Command พร้อมหรือในเว็บเพจที่เปิด
- เมื่อสแกน QR Code แสดง Link ไปยังโปรแกรม Expo Client ให้ทำการกดไปยัง Link ที่สแกนได้



4. จากนั้นระบบจะทำการดาวน์โหลด Package ต่างๆ เอาไว้ที่สมาร์ทโฟนและทำการรันโปรแกรมขึ้นมาผลลัพธ์ที่ได้แสดงดังภาพด้านล่าง



หมายเหตุ หากไม่สามารถรันได้ให้เปิดเว็บเบราว์เซอร์และทำการเปลี่ยน Connection เป็น Tunnel แล้วทำการสแกน QR Code ใหม่อีกครั้ง

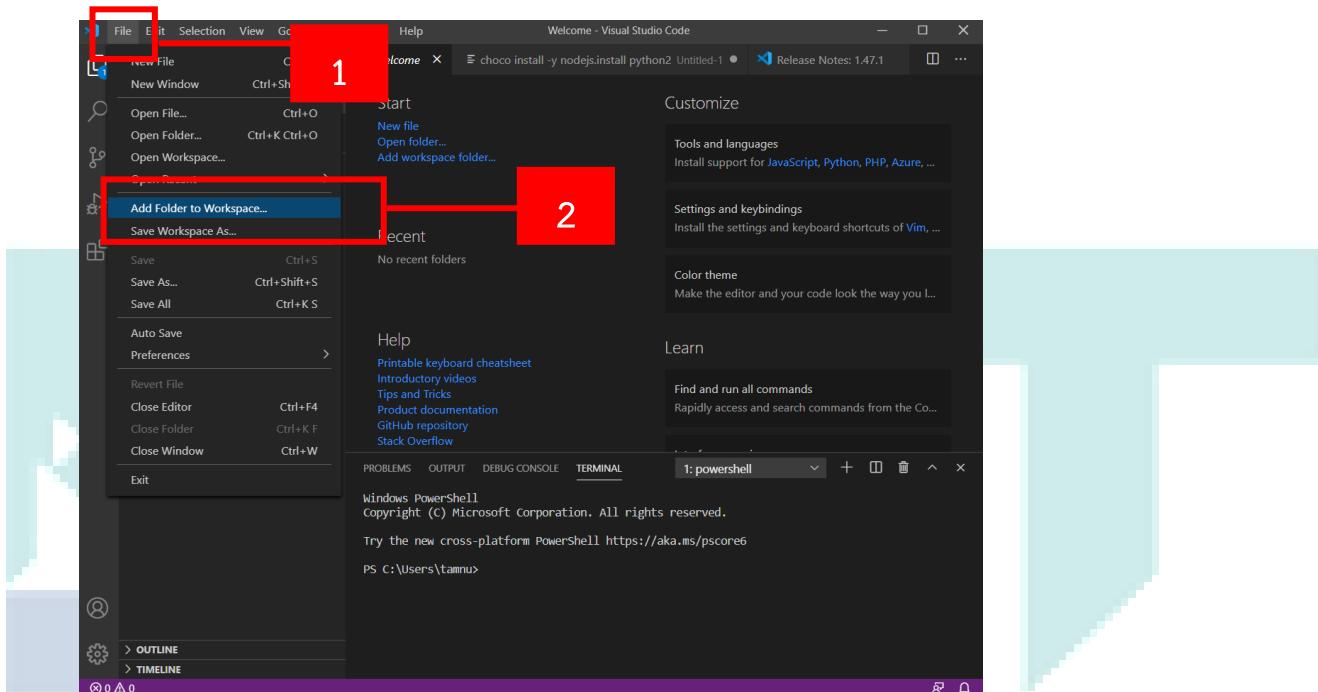


Digital Academy Thailand

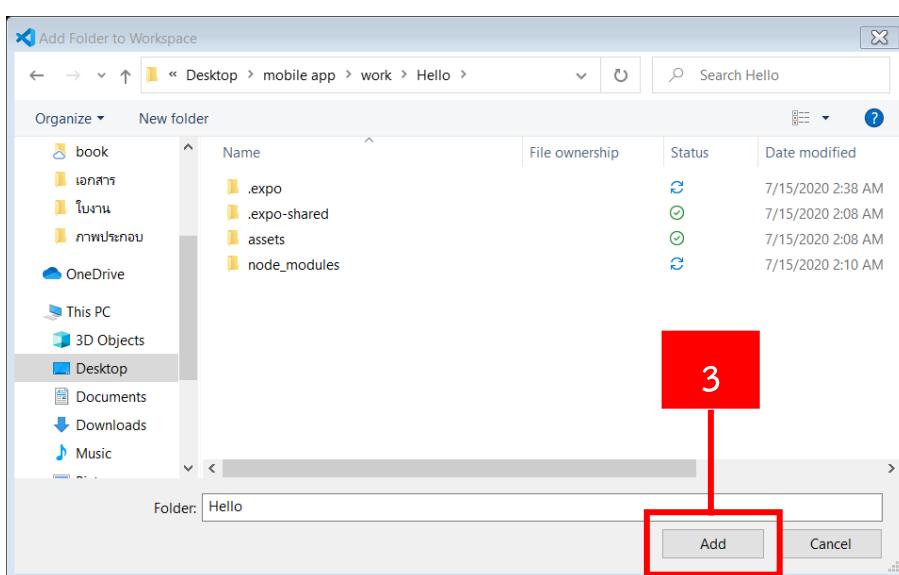
การแก้ไข Code ใน React-Native โปรเจค

การแก้ไข Project เป็นต้นนับนักพัฒนา เป็นต้นนักพัฒนาจะต้องติดตั้งคอม เพื่อใช้แก้ไขโค้ดโปรแกรม Visual Studio Code การดังนี้

1. เปิดโปรแกรม Visual Studio Code จากนั้นเลือกไปที่ File
2. เลือกโฟลเดอร์ Add Folder to Workspace



3. ทำการเลือกโฟลเดอร์ Project Expo ที่เราทำการสร้างขึ้นกด Add



4. Project จะถูกนำเข้าสู่โปรแกรม Visual Studio Code พร้อมให้เราทำการแก้ไขโดยในที่นี่เราจะทำการแก้ไขโค้ดโดยเปลี่ยนข้อความที่แสดงบนหน้าจอเป็นข้อความอย่างอื่นให้กดไปที่ App.js

```

1 import { StatusBar } from 'expo-status-bar';
2 import React from 'react';
3 import { StyleSheet, Text, View } from 'react-native';
4
5 export default function App() {
6   return (
7     <View style={styles.container}>
8       <Text>61xxxxxxxxx_work01</Text>
9       <StatusBar style="auto" />
10    </View>
11  );
12}
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });
22

```

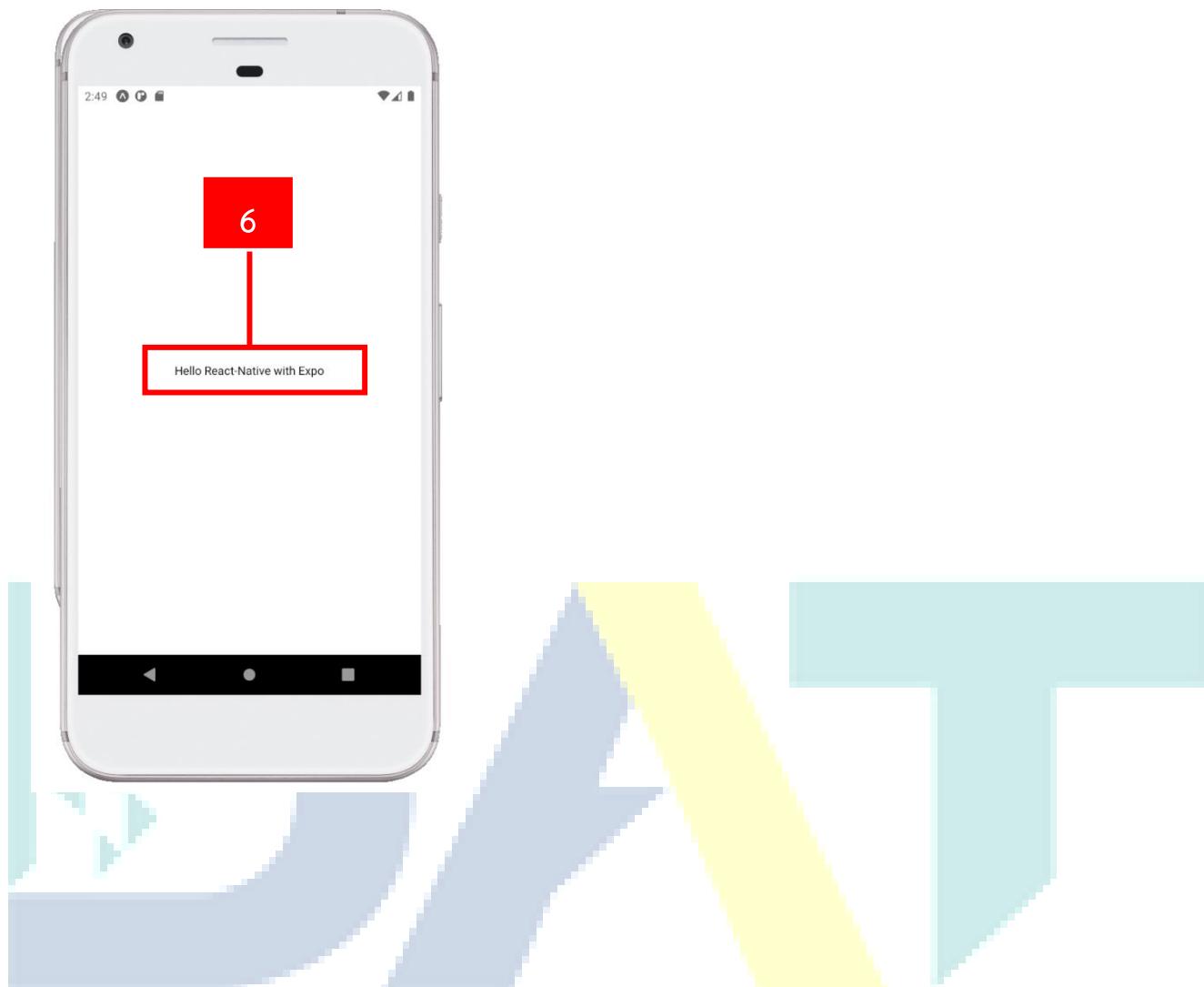
5. จานั้นแก้ไขโปรแกรมตามโค้ดด้านล่าง

```

1. import { StatusBar } from 'expo-status-bar';
2. import React from 'react';
3. import { StyleSheet, Text, View } from 'react-native';
4.
5. export default function App() {
6.   return (
7.     <View style={styles.container}>
8.       <Text>61xxxxxxxxx_work01</Text>
9.       <StatusBar style="auto" />
10.    </View>
11.  );
12. }
13.
14. const styles = StyleSheet.create({
15.   container: {
16.     flex: 1,
17.     backgroundColor: '#fff',
18.     alignItems: 'center',
19.     justifyContent: 'center',
20.   },
21. });

```

6. จากนั้นทำการ Save Project Emulator จะทำการอัพเดจผลลัพธ์ให้อัตโนมัติ



Digital Academy Thailand

บทที่ 2 การใช้งานพื้นฐาน Core Components

การสร้าง GUI

การสร้าง Mobile Application สิ่งที่สำคัญเป็นอันดับแรกๆ ที่นักพัฒนาต้องพบรือคือการออกแบบหน้าตาแอปพลิเคชันหรือนักพัฒนาส่วนใหญ่มักจะเรียกว่าการออกแบบ GUI ซึ่ง ย่อมาจาก Graphical User Interface GUI คือ การติดต่อกับผู้ใช้โดยใช้ภาพสัญลักษณ์ เป็นการออกแบบส่วนของโปรแกรมคอมพิวเตอร์ให้มีการโต้ตอบกับผู้ใช้ โดยการใช้ Icon , รูปภาพ และสัญลักษณ์อื่นๆ เพื่อแทนลักษณะต่างๆ ของโปรแกรม แทนที่ผู้ใช้จะพิมพ์คำสั่งต่างๆ ในการทำงาน ช่วยทำให้ผู้ใช้งานสามารถทำงานได้ง่าย และรวดเร็วขึ้น ไม่จำเป็นต้องจำคำสั่งต่างๆ ของโปรแกรมมากนัก ถือเป็นวิธีการให้ความสะดวกแก่ผู้ใช้งานแอปพลิเคชันให้ติดต่อสื่อสารกับระบบโดยผ่านทางภาพ ซึ่งการสร้าง mobile Application ด้วย react native นี้ก็จะมีวิธีการสร้าง User Interface ที่ต้องเรียนรู้และเข้าใจโดยพื้นฐานของแอปพลิเคชันทั่วๆ ไปจะประกอบด้วย component พื้นฐานซึ่งทาง react-native จะเรียกว่า Basic components โดยการเรียกใช้แต่ละ components จะต้องคำนึงถึงองค์ประกอบกลับ 2 อย่างคือ

1. การเลือก Components การเลือกใช้งาน component แต่ละอย่างนั้นจะขึ้นอยู่กับลักษณะการใช้งานหรือลักษณะความต้องการของ User ตัวอย่างเช่น หากต้องการรับค่าจากผู้ใช้งานโดยผู้ใช้งานจะต้องกรอกตัวเลข อายุ โดย Components ที่เหมาะสมสำหรับเหตุการณ์นี้ก็คือ TextInput ซึ่งสามารถรับข้อมูลได้ทั้งตัวเลขและตัวอักษรจากผู้ใช้งานได้ถ้าหากต้องการข้อมูลที่เป็นตัวเลขนักพัฒนาสามารถตั้งค่า Property เพิ่มเติมลงใน TextInput ได้ ซึ่งการเลือกใช้ component แต่ละอย่างให้เหมาะสมกับงานนั้นนักพัฒนาจะต้องมีประสบการณ์และเข้าใจถึง component ทั้งหมดก่อนจึงจะสามารถพัฒนาแอปพลิเคชันได้อย่างมีประสิทธิภาพ
2. การปรับแต่ง Components นอกจากการเลือก Components ให้เหมาะสมกับงานแล้วนั้นสิ่งที่สำคัญลำดับถัดมา ก็คือการปรับแต่ง Style ให้เข้ากับ theme ที่วางไว้หากเราเคยมีประสบการณ์ในการพัฒนาเว็บไซต์มาก่อน เราจะคงคุ้นเคยกับคำว่า CSS ซึ่งเป็นภาษาในการปรับแต่ง เว็บไซต์ให้มีสีสันสวยงาม ซึ่งการพัฒนา mobile Application ด้วย react นี้ก็จะมี Style Sheet ซึ่งจะมีลักษณะคล้ายคลึงกับ CSS ในเว็บไซต์ที่ใช้ในการปรับแต่งแอปพลิเคชันให้สวยงามได้เช่นกันในการเขียนโปรแกรมและออกแบบตกแต่งแอปพลิเคชันพร้อมๆ กัน เป็นเรื่องที่ยากมากเนื่องจากนักพัฒนาจะต้องใช้ทั้งศาสตร์และศิลป์ควบคู่กันกล่าวคือนักพัฒนาจะต้องรู้จักการเรียกใช้งาน comment นั้นและยังจะต้องรู้จักการปรับแต่ง component ให้สวยงาม ตัวอย่างเช่น Application ต้องการแสดงรูปภาพไฟล์โดยลักษณะกรอบของรูปเป็นวงกลมเพื่อให้เข้ากับ theme ที่วางไว้ ในที่นี่เราสามารถเลือกใช้งาน Image Component โดยมีการปรับแต่งค่าความมนของขอบ Border Radius ให้เป็นวงกลมได้

สังเกตว่าเราจะต้องเขียนโปรแกรมและออกแบบแอปพลิเคชันให้สวยงามพร้อมๆ กันซึ่งเรื่องยากมากที่จะทำสองอย่างนี้ให้ได้โดยบริษัทบางบริษัทมักจะมีการแบ่งงานทั้งสองนี้ออกจากกันซึ่งจะมีทีมในการออกแบบส่วนที่ติดต่อผู้ใช้งาน

โดยเฉพาะเราจะเรียกว่า Front End Developer และทีมที่ใช้สำหรับเขียน Programming Logic ซึ่งรีบว่า Backend Developer ทั้งสองทีมนี้จะต้องทำงานร่วมกันและจะต้องมีความรู้ทางด้าน Front-End และ Back-End แต่จะทำงานได้ดีในเฉพาะงานของตน ในบทนี้เราจะมาทำการเรียนรู้การใช้งาน Component พื้นฐานซึ่งประกอบด้วย View, Text, Image, TextInput, ScrollView, Button, Switch และการจัดวาง layout การปรับแต่ง Component สวยงามด้วย Style



Digital Academy Thailand

Style

Style คือการกำหนดรูปแบบของ Component ต่างๆ ไม่ว่าจะเป็นสี, ขนาด, รูปร่าง เป็นต้น โดยมีลักษณะคล้ายคลึงกับ CSS ในการเขียนเว็บไซต์ โดยการกำหนดสไตล์นั้นสามารถเปลี่ยนได้ด้วยภาษา JavaScript ซึ่งสามารถกระทำได้หลายรูปแบบตามตัวอย่างด้านล่าง



```
1. import React from 'react';
2. import { StyleSheet, Text, View } from 'react-native';
3. const LotsOfStyles = () => {
4.   return (
5.     <View style={{marginTop: 50,}}>
6.       <Text style={{color:
7.         'green',fontSize:50}}>just Green</Text>
8.       <Text style={styles.red}>just Red</Text>
9.       <Text style={styles.blue}>just Blue</Text>
10.      <Text style={[styles.blue,
11.        styles.red]}>Blue,red</Text>
12.    );
13.  const styles = StyleSheet.create({
14.    blue: {
15.      color: 'blue',
16.      fontWeight: 'bold',
17.      fontSize: 30,
18.    },
19.    red: {
20.      color: 'red',
21.    },
22.  });
23.  export default LotsOfStyles;
```

Digital Academy Thailand

จากตัวอย่างสังเกตได้ว่าโปรแกรมนี้จะทำการแสดงข้อความที่มีลักษณะต่างกันโดยการใช้ Text ซึ่งข้อความที่แสดงจะมีการ Style ที่แตกต่างกันดังนี้

1. Component Text ที่แสดงข้อความ Just Green มีการกำหนด Style prop โดยกำหนดไว้ในส่วนที่ประกาศ <Text> โดยกำหนดให้มีสีตัวอักษรเป็นสีเขียว color: 'green' และขนาดของตัวอักษรมีขนาดเท่ากับ 50 fontSize:50
2. Component Text ที่แสดงข้อความ just Red มีการกำหนด Style โดยใช้วิธีการสร้าง Style Prop(เดียวจะกล่าวในบทถัดไป) ด้วยฟังก์ชัน StyleSheet.create ที่มีชื่อว่า red : style={styles.red} โดยมีการกำหนดให้สีตัวอักษรเป็นสีแดง color: 'red'
3. Component Text ที่แสดงข้อความ just blue มีการกำหนด Style โดยใช้วิธีการสร้าง Style Prop (เดียวจะกล่าวในบทถัดไป) ที่มีชื่อว่า blue : style={styles.blue} ด้วยฟังก์ชัน StyleSheet.create โดยมีการกำหนดให้สี

ตัวอักษรเป็นสีน้ำเงิน color: 'Blue' ขนาดตัวอักษรแบบหนา fontWeight: 'bold' และขนาดของตัวอักษรมีขนาดเท่ากับ 50 fontSize:50

4. Component Text ที่แสดงข้อความ Blue,Red มีการเรียกใช้งาน Style Prop ที่ถูกสร้างก่อนหน้านี้พร้อมกันทั้งสองตัวโดยมีการเรียกใช้งานในลักษณะที่เป็นแบบ Array : style={[styles.blue, styles.red]} การแสดงผลลัพธ์จะทำการแสดงตัวอักษรตาม Style Prop ที่ถูกกำหนดไว้ก่อน(styles.blue) จากนั้นจะถูกกำหนดค่าเพิ่มเติมตาม Style Prop ที่อยู่ลำดับหลังของ Array โดยหากมี Prop ใดที่เหมือนกันจะยึดตามอะเรตัวสุดท้าย

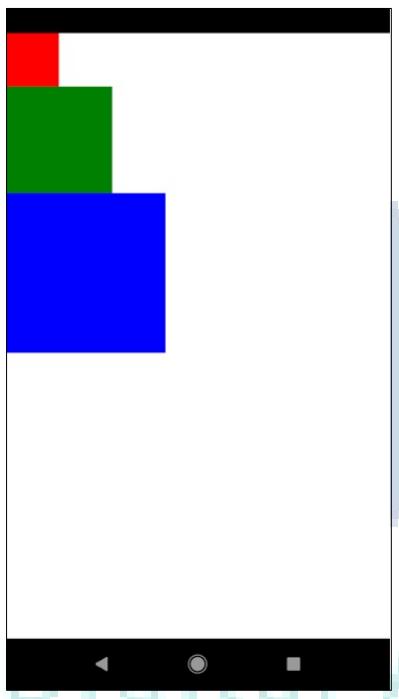


Digital Academy Thailand

Width and Height

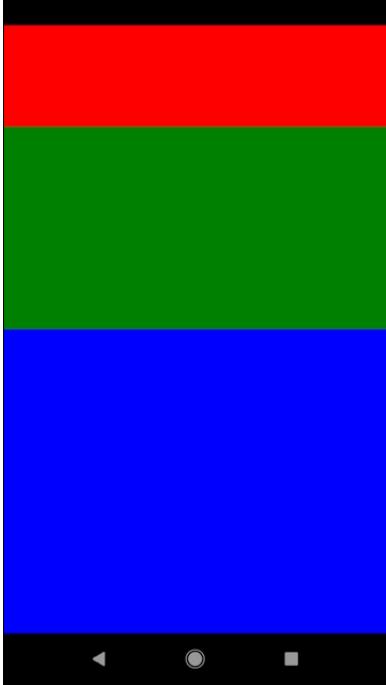
การสร้าง component โดยทั่วไปก็จะต้องมีการกำหนดขนาดให้ component นานๆโดยการกำหนดขนาดของคอมพิวเตอร์ใน react-native สามารถทำได้ 2 วิธีดังนี้

1. Fixed Dimensions เป็นการกำหนดขนาดแบบตายตัวไม่มีการเปลี่ยนแปลง เมื่อนำ Application ที่มีการกำหนดขนาด component แบบ Fixed Dimensions นี้ไปรันในโทรศัพท์สมาร์ทโฟนรุ่นต่างๆที่มีขนาดหน้าจอต่างกันขนาดของ component นั้นจะคงที่เสมอ ซึ่งการกำหนดลักษณะแบบนี้สามารถกระทำได้โดยการสร้าง Style Prop และหนندค่า Width และ Height ด้วยค่าคงที่หรือเปอร์เซ็นต์ตามตัวอย่างด้านล่าง



```
1. import React from 'react';
2. import { View } from 'react-native';
3. const FixedDimensionsBasics = () => {
4.   return (
5.     <View>
6.       <View style={{width: 50, height: 50,
backgroundColor: 'red'}} />
7.       <View style={{width: 100, height: 100,
backgroundColor: 'green'}} />
8.       <View style={{width: '30%', height:
'30%', backgroundColor: 'blue'}} />
9.     </View>
10.   );
11. };
12. export default FixedDimensionsBasics;
```

2. Flex Dimensions เป็นการกำหนดขนาด Component แบบยืดหยุ่นไม่ตายตัวและเต็มพื้นที่(fill) โดยเมื่อนำ Application ที่มีกำหนดขนาด component แบบ Flex Dimensions นี้ไปรันในโทรศัพท์รุ่นต่างๆที่มีขนาดหน้าจอต่างกันขนาดของคอมพิวเตอร์นั้นจะขยายตามขนาดหน้าจอของมือถือแต่ละรุ่น เราสามารถกระทำได้โดยการสร้าง Style Prop ด้วย Flex



```
1. import React from 'react';
2. import { View } from 'react-native';
3. const FlexDimensionsBasics = () => {
4.   return (
5.     <View style={{flex: 1}}>
6.       <View style={{flex: 1, backgroundColor: 'red'}} />
7.       <View style={{flex: 2, backgroundColor: 'green'}} />
8.       <View style={{flex: 3, backgroundColor: 'blue'}} />
9.     </View>
10.   );
11. };
12. export default FlexDimensionsBasics;
```

วิธีการคำนวณพื้นที่ในการใช้งาน Flex

เมื่อมีการใช้งาน Flex กับ component ที่อยู่ใน Container เดียวกันหลายๆ component สิ่งที่ต้องระวังที่สุดนั่นก็คือ การคำนวณพื้นที่ในการแสดงผลจากตัวอย่างโค้ดด้านบน มีการสร้าง View เป็น Container และบรรจุ View ที่เป็น Children อีก 3 อันคือ Red View, Green View และ Blue View แต่ละนั้นมีการกำหนดขนาด Flex ดังนี้ 1, 2 และ 3 รวมขนาด Flex ทั้งสิ้น $1+2+3 = 6$ โดยขนาดของ Red View จะมีขนาดพื้นที่เท่ากับ $1/6$ ของพื้นที่ทั้งหมด, Green View จะมีขนาดพื้นที่เท่ากับ $2/6$ ของพื้นที่ทั้งหมดและ Blue View จะมีขนาดพื้นที่เท่ากับ $3/6$ ของพื้นที่ทั้งหมด ทั้งนี้การใช้ Flex สามารถใช้งานกับ component ได้หลากหลายไม่จำเป็นที่จะใช้เฉพาะ View อย่างเดียว เท่านั้น

View

View เป็น component พื้นฐานที่สุดที่ใช้สำหรับการสร้าง User Interface ในทุกๆ Application ซึ่ง View ก็คือตัวที่บรรจุ Components อื่นๆหรือตัวที่ใช้ในการจัดวาง layout ลักษณะต่างๆของ Application หากเปรียบเทียบกับการเขียน Application แบบ Native ใน Android View ก็อาจเทียบได้ว่าคือ UIView นั่นเอง โดยที่ View ถูกออกแบบให้สามารถบรรจุ Component อื่นๆหรือตัว View เองได้ เช่นกัน

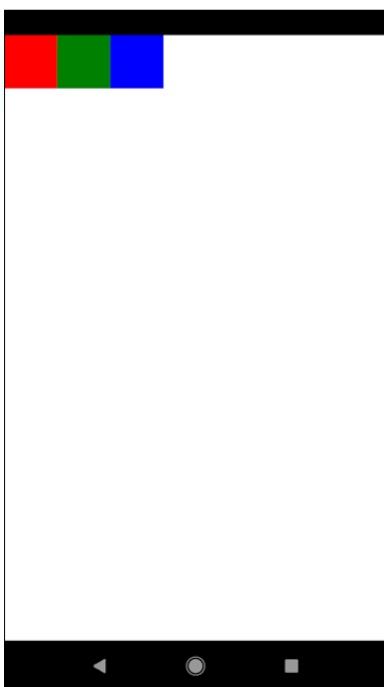
ในการใช้งาน View เราจะต้องทำการ Import Component ที่อยู่ 'react-native' ตามตัวอย่างด้านล่าง

```
1. import { View } from 'react-native';
```

ในการสร้าง layout ของ Application ด้วย View เราจะใช้การกำหนดขนาดแบบ Flex Dimensions เป็นหลัก เพื่อที่จะให้ Application ที่เราออกแบบสามารถทำงานบนโทรศัพท์สมาร์ทโฟนหลายขนาดได้ โดยการใช้งาน Flex Dimensions เราสามารถกำหนดคุณลักษณะของมันได้ดังนี้

1. การกำหนดทิศทาง(Flex Direction)

Flex Direction เป็นการควบคุมทิศทางของ Children Components ที่อยู่ภายใน View ที่เป็น Container โดยสามารถกำหนดค่าใน Style Prop ด้วย Parameter flexDirection ให้กับ Container ตามตัวอย่างโปรแกรมด้านล่าง จะประกอบไปด้วย Container View ที่มีการกำหนดให้ flexDirection: 'row' และประกอบด้วย Children Components 3 ตัวคือ View Red, View Green, ViewBlue. โดยเมื่อทำการจัดเรียง Children แล้วจะทำให้ Children Components ทั้ง 3 เรียงกันในแนวนอน



```
1. import React from 'react';
2. import { View } from 'react-native';
3.
4. const FlexDirectionBasics = () => {
5.   return (
6.     <View style={{flex: 1, flexDirection: 'row'}>
7.       <View style={{width: 50, height: 50, backgroundColor: 'red'} />
8.       <View style={{width: 50, height: 50, backgroundColor: 'green'} />
9.       <View style={{width: 50, height: 50, backgroundColor: 'blue'} />
10.    </View>
11.  );
12.};
13. export default FlexDirectionBasics;
```

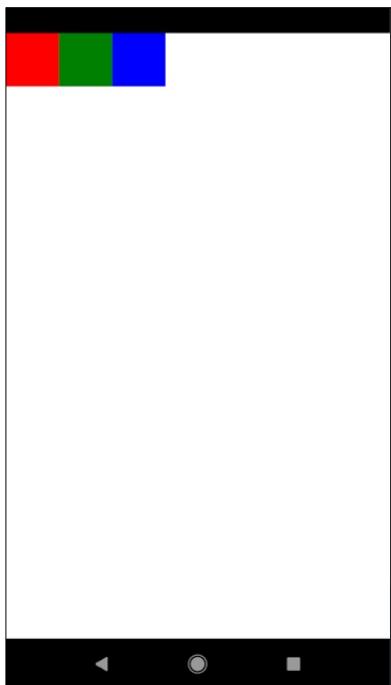
ในการกำหนดค่า Flex Direction สามารถกำหนดค่าได้ดังนี้

- row : เป็นการจัดเรียง children component จากซ้ายไปขวาที่อยู่ภายใต้ View ที่เป็น Container
- column (default value) : เป็นการจัดเรียง children component จากบนลงล่างที่อยู่ภายใต้ View ที่เป็น Container
- row-reverse : เป็นการจัดเรียง children component จากขวาไปซ้ายที่อยู่ภายใต้ View ที่เป็น Container
- column-reverse : เป็นการจัดเรียง children component จากล่างขึ้นบนที่อยู่ภายใต้ View ที่เป็น Container

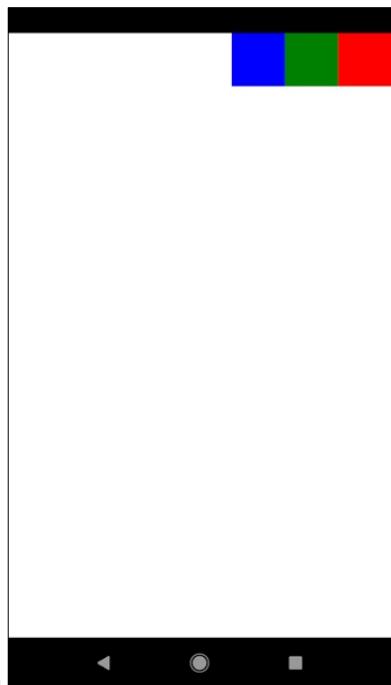


ตัวอย่างการใช้ Flex Direction

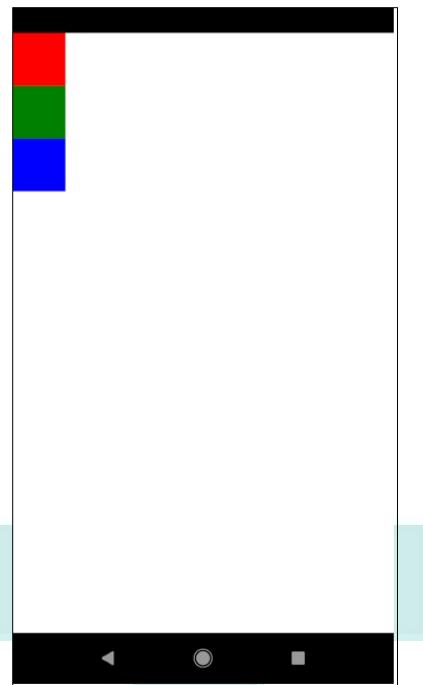
row



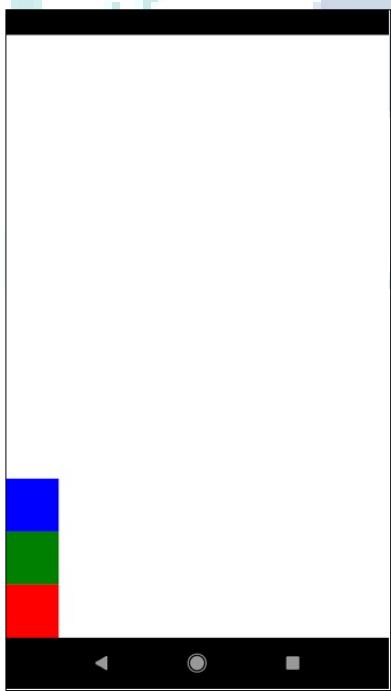
row-reverse



Column



column-reverse



Academy Thailand

2. การจัดแบบเรียงตามแนว Children(Justify Content)

Justify Content เป็นการจัดเรียง Children ภายใน Container ให้อยู่ในตำแหน่งตามที่กำหนดไว้ในแนวเดียวกับ Flex Direction โดยสามารถกำหนดค่าใน Style Prop ด้วย Parameter justifyContent ให้กับ Container ตามตัวอย่างโปรแกรมด้านล่างจะประกอบไปด้วย Container View ที่มีการกำหนดให้ justifyContent: 'space-between' และประกอบด้วย Children Components 3 ตัวคือ View Red, View Green, ViewBlue. โดยเมื่อทำการจัดเรียง Children แล้วจะทำให้ Children Components ทั้ง 3 เรียงกันโดยมีช่องว่างคืน



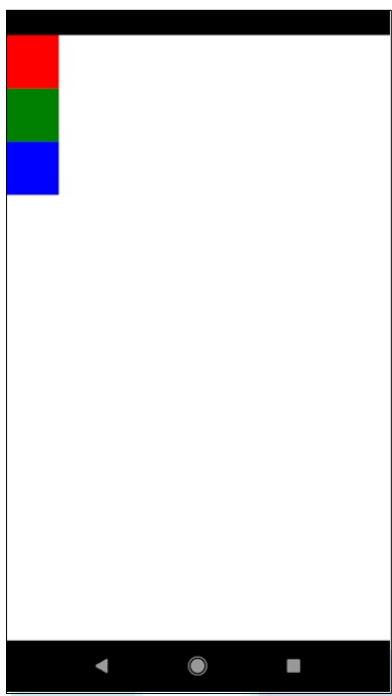
```
1. import React from 'react';
2. import { View } from 'react-native';
3. const JustifyContentBasics = () => {
4.   return (
5.     <View style={{
6.       flex: 1,
7.       flexDirection: 'column',
8.       justifyContent: 'space-between',
9.     }}>
10.    <View style={{width: 50, height: 50,
11.      backgroundColor: 'red'}} />
12.    <View style={{width: 50, height: 50,
13.      backgroundColor: 'green'}} />
14.    <View style={{width: 50, height: 50,
15.      backgroundColor: 'blue'}} />
16.  );
17. export default JustifyContentBasics;
```

ในการกำหนดค่า Justify Content สามารถกำหนดค่าได้ดังนี้

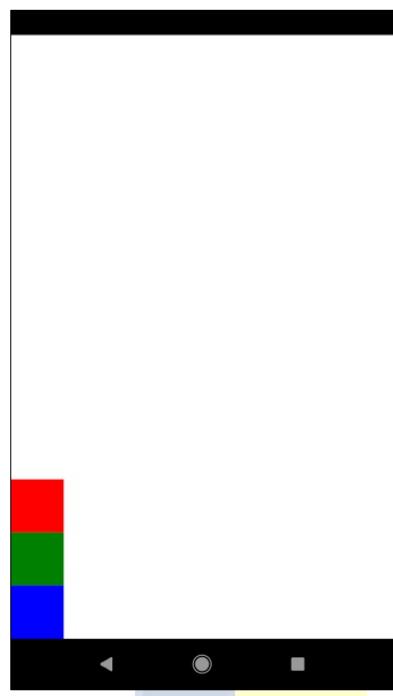
- flex-start(default value) : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งเริ่มต้นของ Container
- flex-end : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งสิ้นสุด Container
- center : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งตรงกลางของ Container
- space-between : เป็นการจัดเรียง Children ภายใน Container โดยจะเว้นช่องว่างระหว่าง Children
- space-around : เป็นการจัดเรียง Children ภายใน Container โดยจะเว้นช่องว่างรอบๆ Children
- space-evenly : เป็นการจัดเรียง Children ภายใน Container โดยจะเว้นช่องว่างรอบๆ Children เท่าๆ กัน

ตัวอย่างการใช้ Justify Content

flex-start(default value)



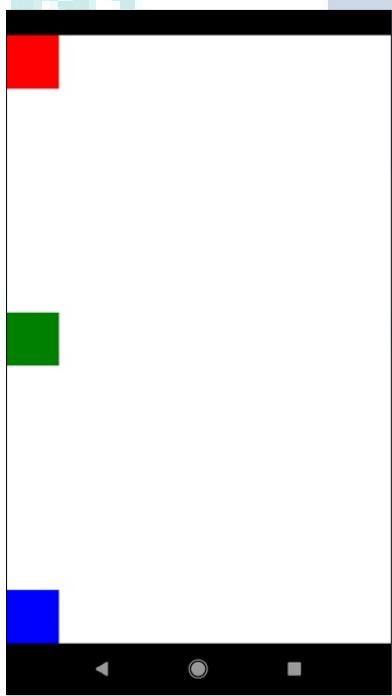
flex-end



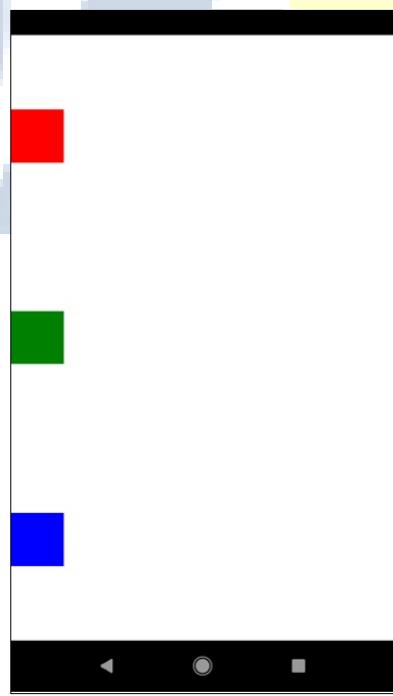
center



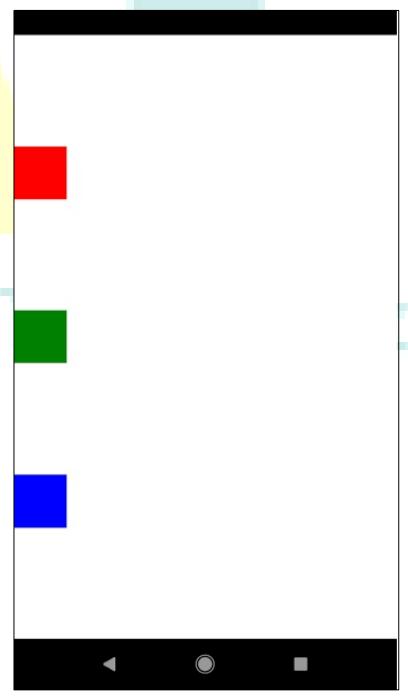
space-between



space-around

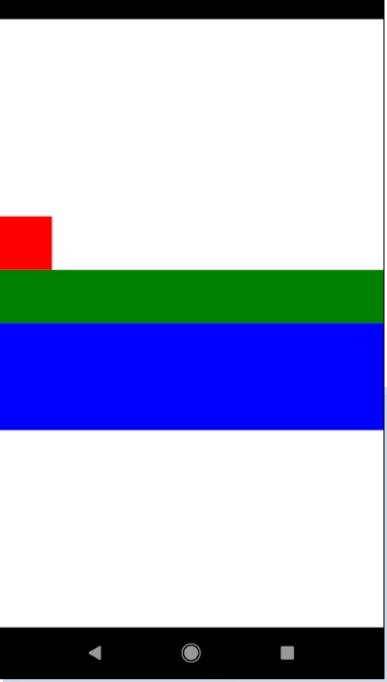


space-evenly



3. การจัดเรียงแบบตรงข้ามแนว (Align Items)

Align Items เป็นการจัดเรียง Children ภายใน Container ให้อยู่ในตำแหน่งตามที่กำหนดไว้ในแนวตรงข้ามกับ Flex Direction โดยสามารถกำหนดค่าใน Style Prop ด้วย Parameter alignItems ให้กับ Container ตามตัวอย่างโปรแกรมด้านล่างจะประกอบไปด้วย Container View ที่มีการกำหนดให้ alignItems: 'stretch' และประกอบด้วย Children Components 3 ตัวคือ View Red, View Green, ViewBlue โดยเมื่อทำการจัดเรียง Children แล้วจะทำให้ Children Components ทั้ง 3 ขยายจนสุดขอบในทิศที่ตรงข้ามกับ Flex Direction

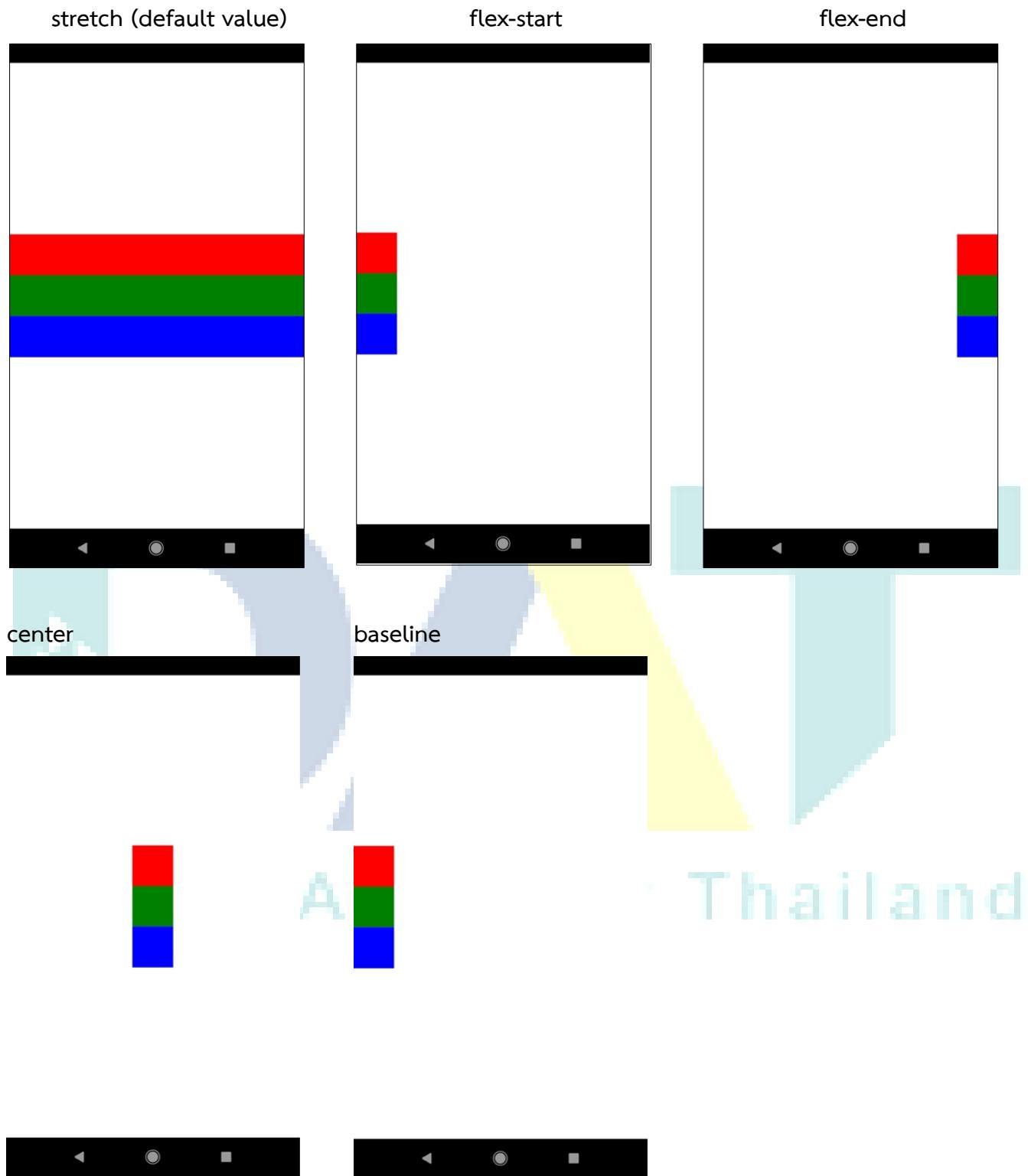


```
1. import React from 'react';
2. import { View } from 'react-native';
3. const AlignItemsBasics = () => {
4.   return (
5.     <View style={{
6.       flex: 1,
7.       flexDirection: 'column',
8.       justifyContent: 'center',
9.       alignItems: 'stretch',
10.      }}>
11.       <View style={{width: 50, height: 50,
12.         backgroundColor: 'red'}} />
13.       <View style={{height: 50,
14.         backgroundColor: 'green'}} />
15.       <View style={{height: 100,
16.         backgroundColor: 'blue'}} />
17.     </View>
18.   );
19. };
20. export default AlignItemsBasics;
```

ในการกำหนดค่า Align Items สามารถกำหนดค่าได้ดังนี้

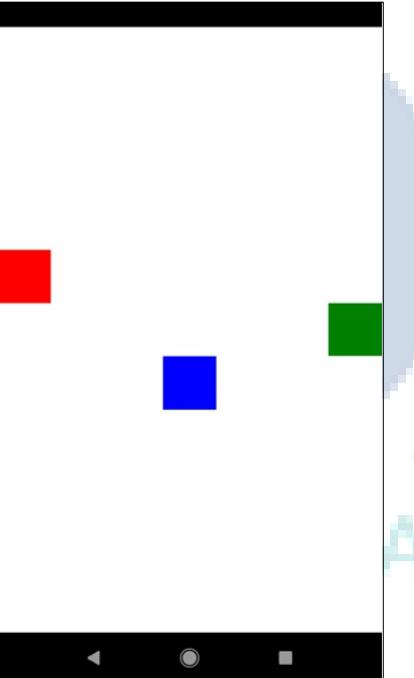
- stretch (default value) : เป็นการจัดเรียง Children ภายใน Container ให้ขยายจนชนขอบ
- flex-start : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งเริ่มต้นของ Container
- flex-end : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งสิ้นสุด Container
- center : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งตรงกลางของ Container
- baseline : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งชิด Baseline ของแต่ละ Children หากมีการตั้งค่าไว้

ตัวอย่างการใช้ Align Items



4. การจัดเรียงแบบตรงข้ามแนว Children(Align Self)

Align Self เป็นการจัดเรียง Children ภายใน Container ให้อยู่ในตำแหน่งตามที่กำหนดไว้ในแนวตรงข้ามกับ Flex Direction เมื่อเทียบกับ Align Items แต่จะทำการกำหนดให้ Children Components แทนที่จะกำหนดให้ Container Direction โดยสามารถกำหนดค่าใน Style Prop ด้วย Parameter alignSelf ให้กับ Children Components ตามตัวอย่างโปรแกรมด้านล่างจะประกอบไปด้วย Container View ที่มีการกำหนดให้และประกอบด้วย Children Components 3 ตัวคือ View Red, View Green, View Blue โดย View Red ทำการกำหนด alignSelf:"flex-start" โดยผลลัพธ์ View Red จะถูกจัดให้อยู่ในตำแหน่งเริ่มต้น, View Green ทำการกำหนด alignSelf:"flex-end" โดยผลลัพธ์ View Green จะถูกจัดให้อยู่ในตำแหน่งสิ้นสุด และ View Blue การกำหนด alignSelf:"center" โดยผลลัพธ์ View Blue ทำการกำหนด alignSelf:"center" โดยผลลัพธ์ View Green จะถูกจัดให้อยู่ในตำแหน่งกลาง



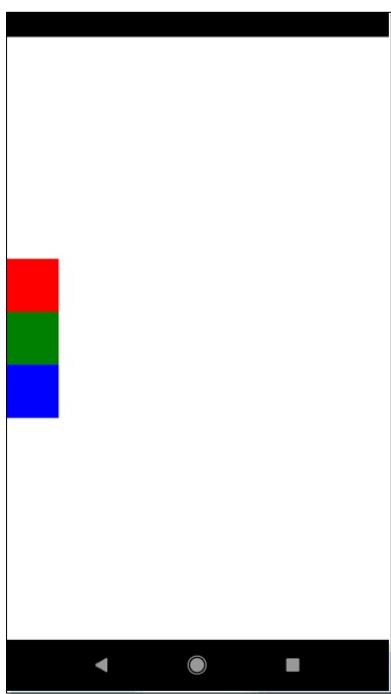
```
1. import React from 'react';
2. import { View } from 'react-native';
3. const AlignSelf = () => {
4.   return (
5.     <View style={{
6.       flex: 1,
7.       flexDirection: 'column',
8.       justifyContent: 'center',
9.     }}>
10.      <View style={{alignSelf:"flex-
start",width: 50, height: 50, backgroundColor:
'red'}} />
11.      <View style={{alignSelf:"flex-end",width:
50, height: 50, backgroundColor: 'green'}} />
12.      <View style={{alignSelf:"center",width:
50, height: 50, backgroundColor: 'blue'}} />
13.    </View>
14.  );
15. };
16. export default AlignSelf;
```

ในการกำหนดค่า Align Self สามารถกำหนดค่าได้ดังนี้

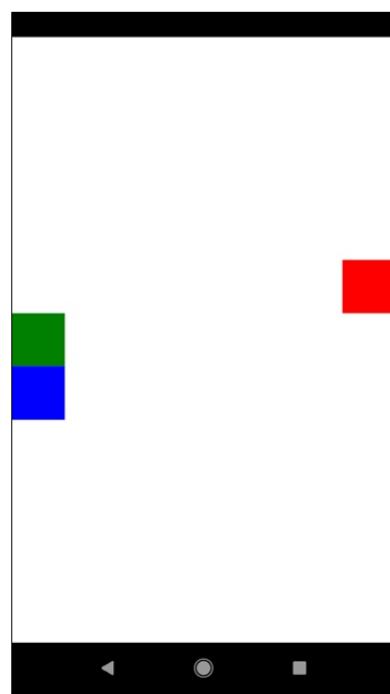
- stretch (default value) : เป็นการจัดเรียง Children ภายใน Container ให้ขยายจนชนขอบ
- flex-start : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งเริ่มต้นของ Container
- flex-end : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งสิ้นสุด Container
- center : เป็นการจัดเรียง Children ภายใน Container โดยให้อยู่ในตำแหน่งตรงกลางของ Container

ตัวอย่างการใช้ Align Self

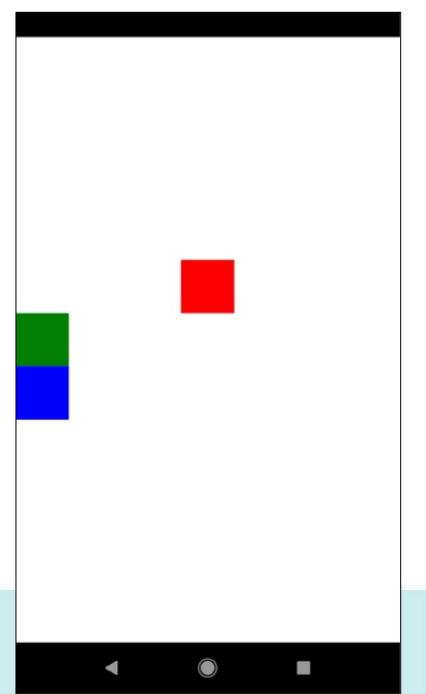
flex-start



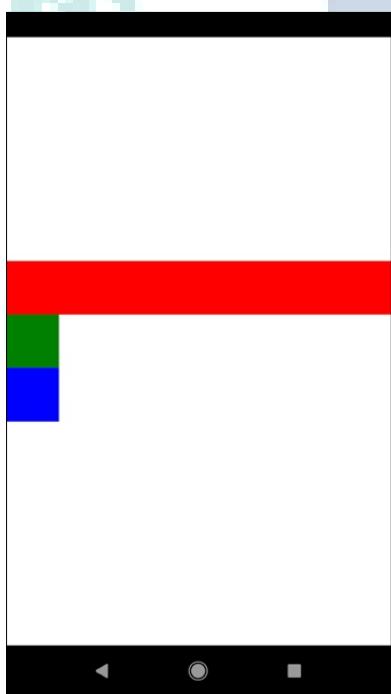
flex-end



center



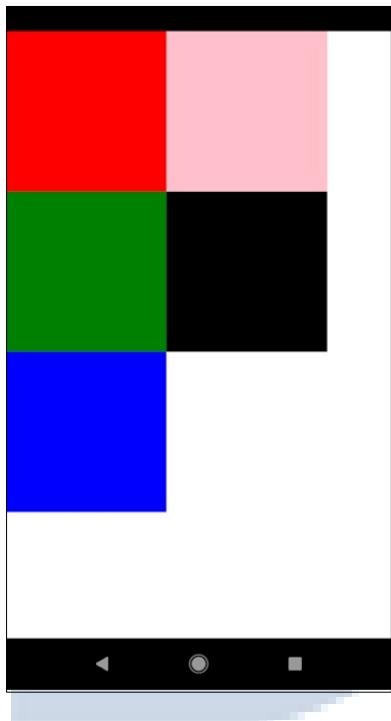
stretch



Academy Thailand

5. การปัดการแสดงผล (Flex Wrap)

Flex Wrap เป็นการกำหนดค่าและควบคุม Containers ในการแสดง Children ในกรณีที่มีการบรรจุ Children เกินกว่าขนาดของ Containers จะสามารถบรรจุได้ในแนวเดียวกันกับ Flex Direction โดยทั่วไปเมื่อมีการบรรจุ Children เกินกว่าขนาดของ Containers ส่วนที่เกินมักจะตกลงมาไว้ แต่ถ้าหากมีการกำหนดค่า Style Prop ด้วย flexWrap:"wrap" จะทำการแสดงส่วนที่เกินมาได้โดยทำการปัดตามตัวอย่างโค้ดด้านล่าง



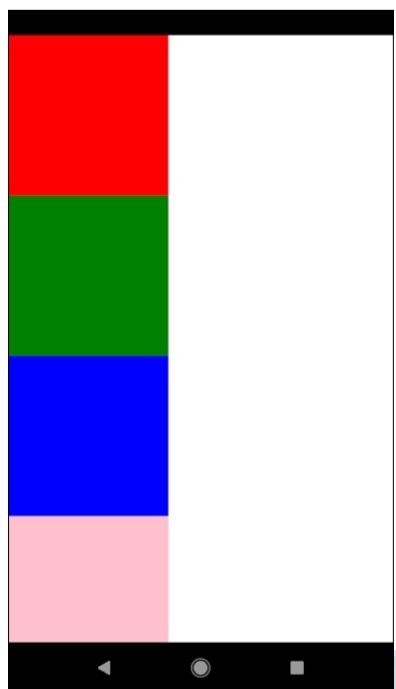
```
1. import React from 'react';
2. import { View } from 'react-native';
3. const AlignSelf = () => {
4.   return (
5.     <View style={{
6.       flex: 1,
7.       flexDirection: 'column',
8.       flexWrap:"wrap"
9.     } }>
10.    <View style={{width: 150, height: 150,
11.      backgroundColor: 'red'}} />
12.    <View style={{width: 150, height: 150,
13.      backgroundColor: 'green'}} />
14.    <View style={{width: 150, height: 150,
15.      backgroundColor: 'blue'}} />
16.    <View style={{width: 150, height: 150,
17.      backgroundColor: 'pink'}} />
18.    <View style={{width: 150, height: 150,
19.      backgroundColor: 'black'}} />
20.  );
21. };
22. export default AlignSelf;
```

ในการกำหนดค่า Flex Wrap สามารถกำหนดค่าได้ดังนี้

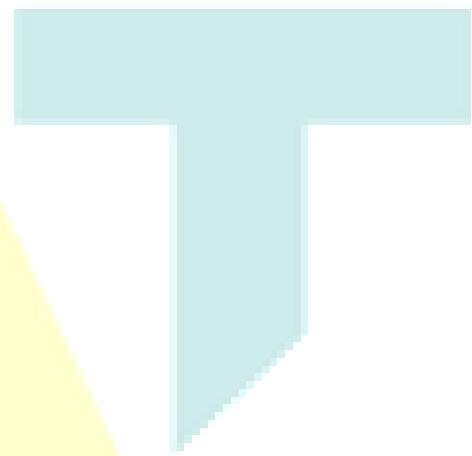
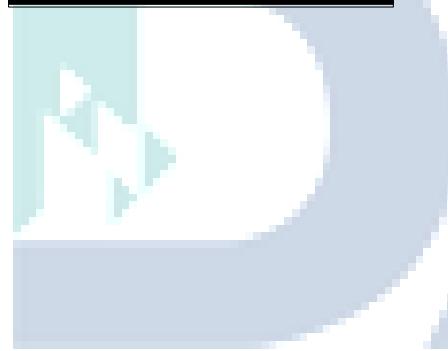
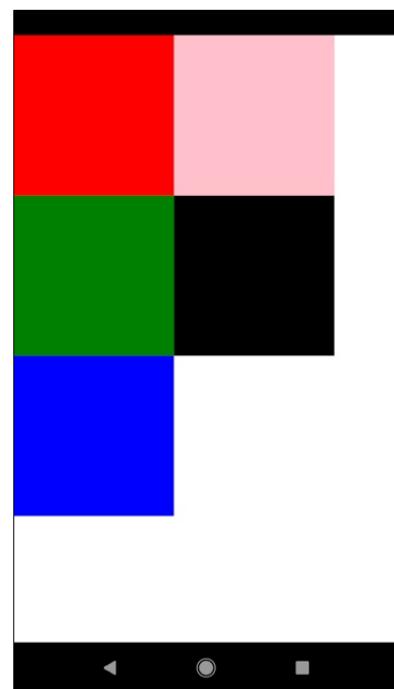
- wrap : ปัดการแสดงผล
- nowrap : ไม่มีการปัดการแสดงผล

ตัวอย่างการใช้ Flex Wrap

nowrap



wrap



Digital Academy Thailand

Text

Text เป็นการแสดงข้อความในโทรศัพท์สมาร์ทโฟน โดยความสามารถกำหนด Style และควบคุม Event ต่างๆ ของ Text ได้ (การควบคุม Event จะกล่าวในบทถัดไป)

การติดตั้ง

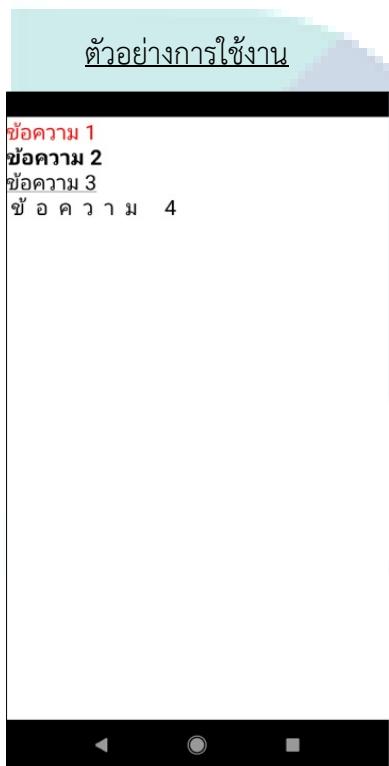
ในการใช้งาน Text จะต้องทำการ Import Component Text ที่อยู่ในไลบรารี 'react-native' เสียก่อนตามตัวอย่างด้านล่าง

```
import { Text } from 'react-native';
```

การเรียกใช้งาน

การใช้งาน Text โดยส่วนมากมักใช้ในการแสดงข้อความทั่วไปตามวิธีการด้านล่าง

```
<Text style={{...}}>ตัวอย่าง</Text>
```



```
1. import React, { useState } from "react";
2. import { Text, StyleSheet } from "react-native";
3. const TextInANest = () => {
4.   return (
5.     <Text style={styles.baseText}>
6.       <Text style={styles.txt1}>{"ข้อความ 1\n"}</Text>
7.       <Text style={styles.txt2}>{"ข้อความ 2\n"}</Text>
8.       <Text style={styles.txt3}>{"ข้อความ 3\n"}</Text>
9.       <Text style={styles.txt4}>{"ข้อความ 4\n"}</Text>
10.    </Text>
11.  );
12. };
13. const styles = StyleSheet.create({
14.   baseText: {
15.     fontFamily: "Cochin",
16.     fontSize: 20
17.   },
18.   txt1: {
19.     color: "red"
20.   },
21.   txt2: {
22.     fontWeight: "bold"
23.   },
24.   txt3: {
25.     textDecorationLine: "underline"
26.   },
27.   txt4: {
28.     letterSpacing: 10
29.   },
30. });
31. export default TextInANest;
```

Image

Image เป็นการแสดงรูปภาพที่อยู่ในภายหน่วยความจำหรือแม้กระทั้งแสดงรูปภาพใน Network ในแอพพลิเคชันที่เราเขียนขึ้นมา โดยความสามารถกำหนด Style และควบคุม Event ต่างๆ ของได้ (การควบคุม Event จะกล่าวในบทถัดไป)

การติดตั้ง

ในการใช้งาน Image จะต้องทำการ Import Component Image ที่อยู่ในไลบรารี 'react-native' เสียก่อนตามตัวอย่างด้านล่าง

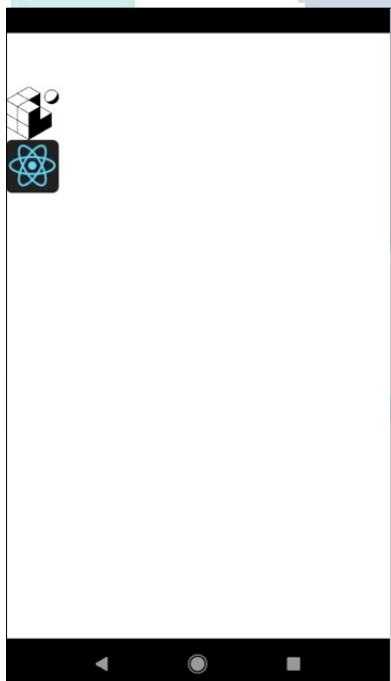
```
import { Image } from 'react-native';
```

การเรียกใช้งาน

การใช้งาน Image โดยส่วนมากมักใช้ในการแสดงรูปภาพทั่วๆ ไปตามวิธีการด้านล่าง

```
<Image source={image} style={styles.image}/>
```

ตัวอย่างการใช้งาน



```
1. import React from 'react';
2. import { View, Image, StyleSheet } from 'react-native';
3.
4. const styles = StyleSheet.create({
5.   container: {
6.     paddingTop: 50,
7.   },
8.   tinyLogo: {
9.     width: 50,
10.    height: 50,
11.  },
12. });
13.
14. const DisplayAnImage = () => {
15.   return (
16.     <View style={styles.container}>
17.       <Image
18.         style={styles.tinyLogo}
19.         source={require('assets/snack-icon.png')}
20.       />
21.       <Image
22.         style={styles.tinyLogo}
23.         source={{
24.           uri: 'https://reactnative.dev/img/tiny_
25.             logo.png',
26.         }}
27.       />
28.     );
29. }
30.
31. export default DisplayAnImage;
```

ImageBackground

ImageBackground เป็นการแสดงพื้นหลังด้วยรูปภาพที่อยู่ในภายหน่วยความจำหรือแม้กระทั้งแสดงรูปภาพใน Network ในแอพพลิเคชันที่เราเขียนขึ้นมา โดยเรารสามารถกำหนด Style

การติดตั้ง

ในการใช้งาน Image จะต้องทำการ Import Component ImageBackground ที่อยู่ในไลบรารี 'react-native' เสียก่อนตามตัวอย่างด้านล่าง

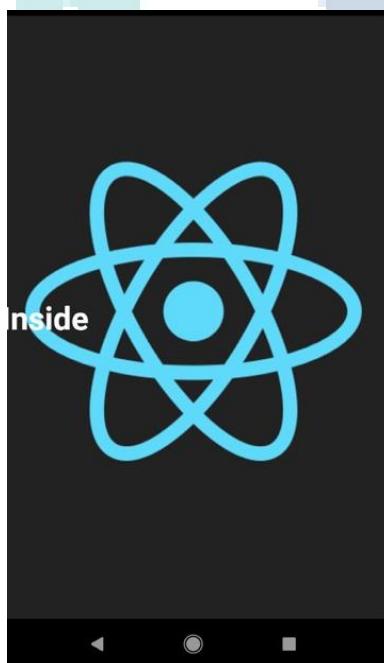
```
import { ImageBackground } from 'react-native';
```

การเรียกใช้งาน

การใช้งาน ImageBackground โดยส่วนมากมักใช้ในการแสดงข้อความทั่วไปตามวิธีการด้านล่าง

```
<ImageBackground source={image} style={styles.image}>....</ImageBackground>
```

ตัวอย่างการใช้งาน



```
1. import React from "react";
2. import { ImageBackground, StyleSheet, Text, View } from "react-native";
3. const image = { uri: "https://reactjs.org/logo-og.png" };
4. const App = () => (
5.   <View style={styles.container}>
6.     <ImageBackground source={image} style={styles.image}>
7.       <Text style={styles.text}>Inside</Text>
8.     </ImageBackground>
9.   </View>
10. );
11. const styles = StyleSheet.create({
12.   container: {
13.     flex: 1,
14.     flexDirection: "column"
15.   },
16.   image: {
17.     flex: 1,
18.     resizeMode: "cover",
19.     justifyContent: "center"
20.   },
21.   text: {
22.     color: "grey",
23.     fontSize: 30,
24.     fontWeight: "bold"
25.   }
26. });
27. export default App;
28.
```

Button

Button คือปุ่มที่ใช้รับคำสั่งหรือควบคุมการทำงานเมื่อมีการกดจากผู้ใช้งาน โดยเราสามารถควบคุม Event ต่างๆ ของได้(การควบคุม Event จะกล่าวในบทถัดไป)

การติดตั้ง

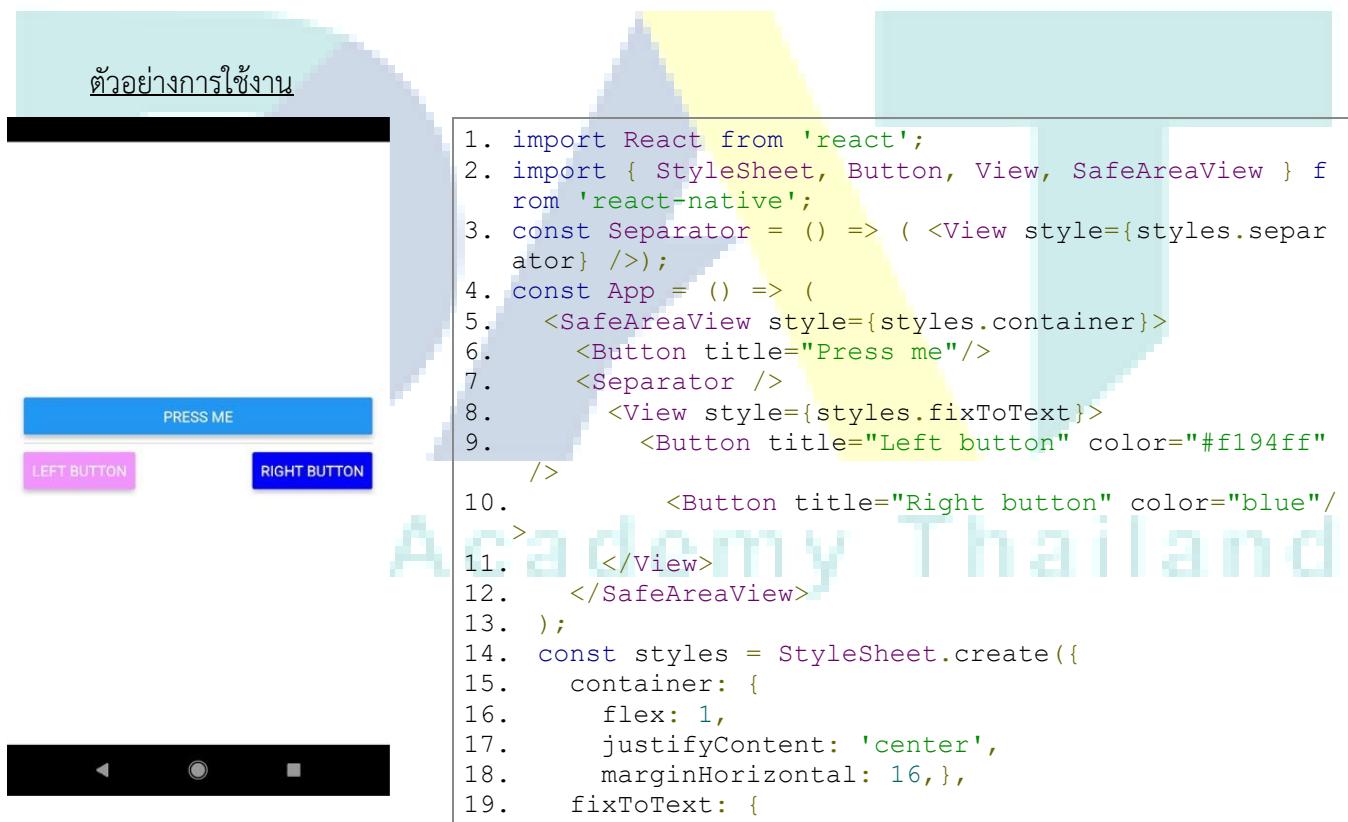
ในการใช้งาน Button จะต้องทำการ Import Component Button ที่อยู่ในไลบรารี 'react-native' เสียก่อนตามตัวอย่างด้านล่าง

```
import { Button } from 'react-native';
```

การเรียกใช้งาน

การใช้งาน Button โดยส่วนมากมักใช้ในการแสดงข้อความที่่่ไปตามวิธีการด้านล่าง

```
<Button title="title name" onPress={onPress} />
```



```
1. import React from 'react';
2. import { StyleSheet, Button, View, SafeAreaView } from 'react-native';
3. const Separator = () => ( <View style={styles.separator} /> );
4. const App = () => (
5.   <SafeAreaView style={styles.container}>
6.     <Button title="Press me"/>
7.     <Separator />
8.     <View style={styles.fixToText}>
9.       <Button title="Left button" color="#f194ff" />
10.      <Button title="Right button" color="blue"/>
11.    </View>
12.  </SafeAreaView>
13. );
14. const styles = StyleSheet.create({
15.   container: {
16.     flex: 1,
17.     justifyContent: 'center',
18.     marginHorizontal: 16,
19.   },
20.   fixToText: {
21.     flexDirection: 'row',
22.     justifyContent: 'space-between',
23.   },
24.   separator: {
25.     marginVertical: 8,
26.     borderBottomColor: '#737373',
27.     borderBottomWidth: StyleSheet.hairlineWidth,
28.   },
29. });
30. export default App;
```

TouchableOpacity

TouchableOpacity คือปุ่มที่ใช้รับคำสั่งหรือควบคุมการทำงานเมื่อมีการกดจากผู้ใช้งานคล้ายคลึงกับ Button โดยสามารถกำหนด Style ได้มากกว่า Button และควบคุม Event ต่างๆของได้ (การควบคุม Event จะกล่าวในบทถัดไป)
การติดตั้ง

ในการใช้งาน Button จะต้องทำการ Import Component Button ที่อยู่ในไลบรารี 'react-native' เสียก่อนตามตัวอย่างด้านล่าง

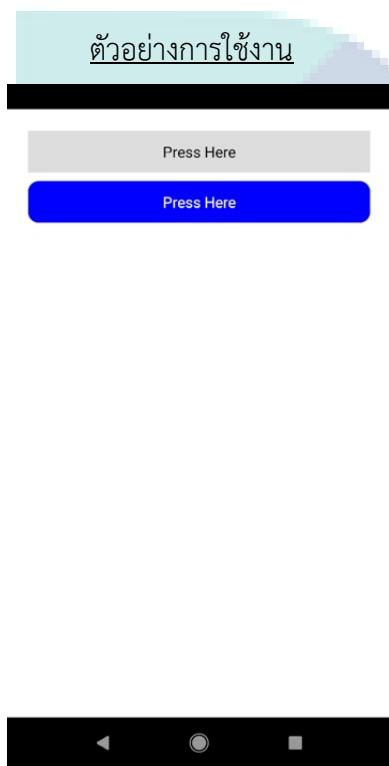
```
import { TouchableOpacity } from 'react-native';
```

การเรียกใช้งาน

การใช้งาน Button โดยส่วนมากมักใช้ในการแสดงข้อความที่่่ไปตามวิธีการด้านล่าง

```
<TouchableOpacity style={styles.button} onPress={onPress}></TouchableOpacity>
```

ตัวอย่างการใช้งาน



```
1. import React from 'react';
2. import { StyleSheet, TouchableOpacity, View, Text,
SafeAreaView } from 'react-native';
3. const App = () => (
4.   <SafeAreaView style={styles.container}>
5.     <TouchableOpacity style={styles.button1}>
6.       <Text>Press Here</Text>
7.     </TouchableOpacity>
8.
9.     <TouchableOpacity style={styles.button2}>
10.      <Text style={{color:"white"}}>Press
    Here</Text>
11.    </TouchableOpacity>
12.  </SafeAreaView>;
13. const styles = StyleSheet.create({
14.   container: {
15.     flexDirection: 'column',
16.     flex: 1,
17.     margin:16, },
18.   button1: {
19.     alignItems: "center",
20.     backgroundColor: "#DDDDDD",
21.     padding: 10,
22.     margin:4
23.   },
24.   button2: {
25.     alignItems: "center",
26.     backgroundColor: "blue",
27.     padding: 10,
28.     margin:4,
29.     borderRadius:10
30.   } });
31. export default App;
```

Switch

Switch คือปุ่มสวิทซ์เปิดปิดที่ใช้รับคำสั่งหรือควบคุมการทำงานเมื่อมีการกดจากผู้ใช้งาน โดยเราสามารถกำหนด Style และควบคุม Event ต่างๆ ของได้ (การควบคุม Event จะกล่าวในเบื้องต้น)

การติดตั้ง

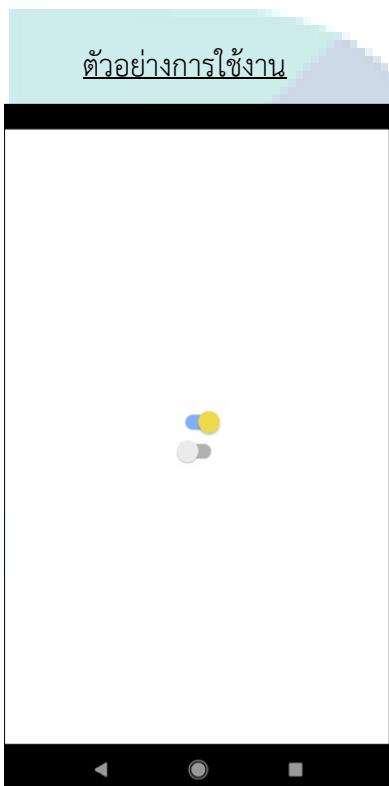
ในการใช้งาน Switch จะต้องทำการ Import Component Button ที่อยู่ในไลบรารี 'react-native' เสียก่อนตามตัวอย่างด้านล่าง

```
import { Switch } from 'react-native';
```

การเรียกใช้งาน

การใช้งาน Button โดยส่วนมากมักใช้ในการแสดงข้อความที่ว่าไปตามวิธีการด้านล่าง

```
<Switch value={isEnabled}/>
```



```
1. import React, { useState } from "react";
2. import { View, Switch, StyleSheet } from "react-native";
3. const App = () => {
4.   const [isEnabled, setIsEnabled] = useState(true);
5.
6.   return (
7.     <View style={styles.container}>
8.       <Switch
9.         trackColor={{ false: "#767577", true:
"#"81b0ff" }}
10.        thumbColor={isEnabled ? "#f5dd4b" :
"#f4f3f4"}
11.        ios_backgroundColor="#3e3e3e"
12.        value={isEnabled}
13.      />
14.      <Switch/>
15.    </View>
16.  );
17. }
18. const styles = StyleSheet.create({
19.   container: {
20.     flex: 1,
21.     alignItems: "center",
22.     justifyContent: "center"
23.   }
24. });
25. export default App;
```

ตัวอย่างการใช้งาน

โดยแรกเริ่มจะลองทำ GUI ง่ายๆ ก่อนเพื่อเรียนรู้การใช้ Component และการจัดวาง Layout ซึ่งเราจะลองทำ Login Page ก่อน แอพพลิเคชันส่วนใหญ่ในปัจจุบันโดยมากมักจะให้ผู้ใช้งานทำการสมาชิก และทำการ Login ก่อนใช้งาน ซึ่ง Page login นี้จะต้องมีองค์ประกอบ 3 ส่วนหลัก ได้แก่



1. ตัวบงชี้แอพพลิเคชัน คือ ตัวระบุว่าแอพพลิเคชันนี้คือแอพพลิเคชันอะไร อาจใช้ข้อความ, รูปภาพหรือโลโก้ต่างๆ ก็ได้

2. การยืนยันตัวบุคคล คือ ส่วนที่ให้ผู้ใช้งานในการกรอกข้อมูลเพื่อใช้ในการยืนยันตัวเองโดยมากจะมี User Name, Email, Password หรือ Telephone Number และมีปุ่มเพื่อกดยืนยันในการ login เป็นต้น

3. ส่วนขยาย คือ ส่วนที่เพิ่มเติมที่จะช่วยให้ผู้ใช้งานสามารถสมัครสมาชิกหรือทำการกู้บัญชีในกรณีที่ลืมข้อมูลของตนเองเป็นต้น

จากตัวอย่าง รูปภาพข้างต้นเป็นหน้า Page Login ที่ใช้เป็นตัวอย่างให้เราได้ลองทดสอบสร้างกันซึ่งมีองค์ประกอบครบถ้วน 3 ส่วน โดยใช้ React-native และ Expo นี้ในการสร้างและมีขั้นตอนการทำลำดับดังนี้

1. ทำการใส่พื้นหลัง

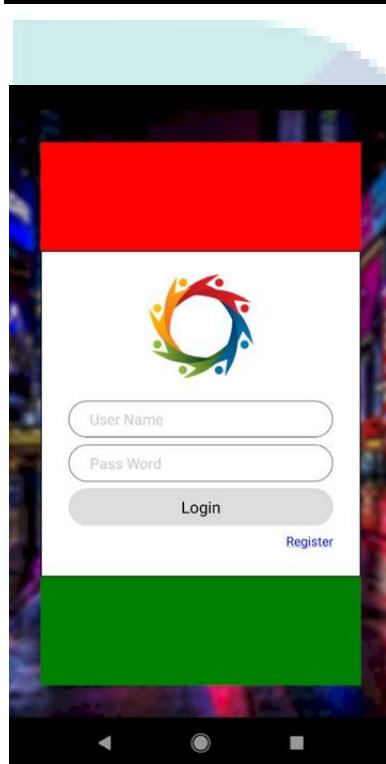
สิ่งแรกที่ต้องทำหลังจากสร้างโปรเจค คือการตกแต่งพื้นหลังของโปรเจคในจากตัวอย่างสังเกตว่าพื้นหลังจะเป็นรูปภาพและมีการเบลอภาพเล็กน้อยเพื่อความสวยงาม ซึ่งเราจะใช้ ImageBackground ใน การสร้างและมีการตั้งค่า blurRadius Prop เพื่อทำให้ภาพมีลักษณะเบลอเล็กน้อย





2. แบ่ง Layout

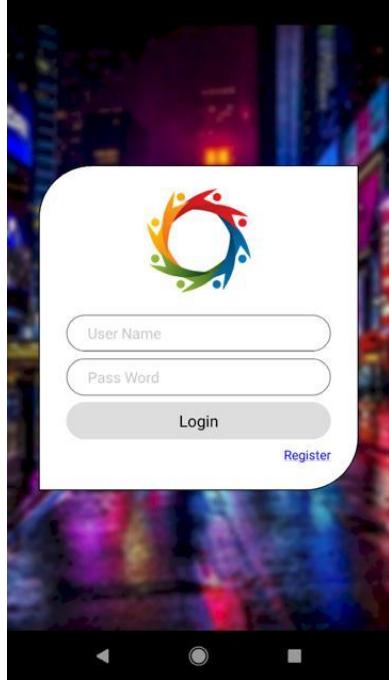
หลังจากเราทำการใส่พื้นหลังเรียบร้อยและเรามาเป็นจะต้องจัดว่าง Layout ก่อน โดยเราจะทำการแบ่ง layout ออกเป็น 3 ส่วนคือ Top, Middle, Bottom โดยแต่ละส่วนจะทำการกำหนดขนาดแบบ Flex Dimension ตามลำดับดังนี้ 1,2,5,1 หลายๆท่านอาจจะเกิดข้อสงสัยเหตุใดจึงจำเป็นต้องแบ่ง Layout ออกเป็น 3 ส่วน ทำไมไม่ทำเป็นส่วนเดียวเลย เหตุผล เพราะหากเราทำการแบ่งเป็นส่วนเดียวเราจะจำเป็นจะต้องทำการกำหนดความสูงของ layout นั้นๆซึ่งอาจจะมีปัญหาในการเอาไปรันในครีอื่นที่มีขนาดไม่เท่ากันได้



3. เพิ่ม Component ต่างๆ

เราจะทำการใส่ Component หรือใส่รายละเอียดเพิ่มเติมลงใน Page ของเรา ซึ่งเราจะทำการใส่ Logo, UserName, Password, ปุ่ม Login, Register ซึ่งทั้งหมดนี้เราจะต้องเลือกใช้ Component ให้เหมาะสมกับงานดังนี้

UI Elements	Component
Logo	Image
UserName	TextInput
Password	TextInput
Login	Text และ TouchableOpacity
Register	Text และ TouchableOpacity



4. ตกแต่งรายละเอียด

ขั้นสุดท้ายนี้เราจะทำการตั้งแต่งส่วนที่เหลือให้สมบูรณ์มากยิ่งขึ้นคือ ทำการปรับสีของ Component, ปรับตำแหน่งของ Component และรายละเอียดเพิ่มเติม เมื่อทำการตั้งแต่งเสร็จสิ้นเราก็จะทำการทดสอบในโทรศัพท์สมาร์ทโฟนหลายรุ่นที่มีขนาดหน้าจอต่างกันเพื่อดูความเปลี่ยนแปลงของ UI ว่ายังอยู่ในลักษณะเดิมหรือไม่ ซึ่งหลาย ๆ คนอาจจะมองว่าเป็นเรื่องยากในการหาโทรศัพท์หลายเครื่องมาเพื่อทดสอบ ในการนี้เราสามารถใช้ Emulator ช่วยในการทดสอบได้



Digital Academy Thailand

ตัวอย่างโค้ด Login UI

```
1. import React from "react";
2. import { View, StyleSheet, Image, TextInput, TouchableOpacity, Text, ImageBackground } from "react-native";
3.
4. const Login = () => {
5.     return (
6.
7.         <ImageBackground
8.             style={styles.imageBackground}
9.             source={{uri:'https://sv1.picz.in.th/images/2020/07/28/EYFj0b.jpg'}}
10.            blurRadius={1}>
11.
12.         <View style={styles.container}>
13.
14.             <View style={styles.top} />
15.
16.             <View style={styles.middle} >
17.                 <View style={styles.viewLogo}>
18.                     <Image
19.                         style={styles.image}
20.                         source={{uri:'https://sv1.picz.in.th/images/2020/07/28/Ez0iOl
.png'}}}
21.                     />
22.                 </View>
23.
24.                 <View style={styles.viewLogin}>
25.                     <TextInput placeholder="User Name" style={styles.textInput}/>
26.
27.                     <TextInput placeholder="Pass Word" style={styles.textInput}/>
28.
29.                     <TouchableOpacity style={styles.buttonLogin}>
30.                         <Text style={{fontSize:15}}>Login</Text>
31.                     </TouchableOpacity>
32.
33.                     <TouchableOpacity style={styles.buttonRegister}>
34.                         <Text style={{fontSize:12, textDecorationLine:"underline", colo
r:'blue'}}>Register</Text>
35.                     </TouchableOpacity>
36.
37.                 </View>
38.
39.             </View>
40.
41.             <View style={styles.button} />
42.
43.         </View>
44.
45.     </ImageBackground>
46. );
47. }
48.
49. const styles = StyleSheet.create({
50.     container: {
51.         flex: 1,
52.         flexDirection:'column',
53.         padding: 20,
54.         margin: 10,
55.     },
56.     top: {
57.         flex: 1,
58.     },
59.     middle: {
60.         flex: 1,
61.     },
62.     bottom: {
63.         flex: 1,
64.     },
65.     viewLogo: {
66.         flex: 1,
67.         width: 100,
68.         height: 100,
69.         margin: 10,
70.         alignContent: 'center',
71.         alignItems: 'center',
72.         justifyContent: 'center',
73.     },
74.     image: {
75.         width: 100,
76.         height: 100,
77.     },
78.     viewLogin: {
79.         flex: 1,
80.         width: 100,
81.         height: 100,
82.         margin: 10,
83.         alignContent: 'center',
84.         alignItems: 'center',
85.         justifyContent: 'center',
86.     },
87.     textInput: {
88.         width: 100,
89.         height: 40,
90.         margin: 10,
91.         padding: 5,
92.         border: 1px solid #ccc,
93.         borderRadius: 5,
94.         color: #333,
95.     },
96.     button: {
97.         width: 100,
98.         height: 40,
99.         margin: 10,
100.        border: 1px solid #ccc,
101.        borderRadius: 5,
102.        color: #fff,
103.        background: '#007bff',
104.        padding: 5,
105.        textTransform: 'uppercase',
106.        fontWeight: 'bold',
107.        textAlign: 'center',
108.        cursor: 'pointer',
109.    },
110.    buttonLogin: {
111.        width: 100,
112.        height: 40,
113.        margin: 10,
114.        border: 1px solid #ccc,
115.        borderRadius: 5,
116.        color: #fff,
117.        background: '#007bff',
118.        padding: 5,
119.        textTransform: 'uppercase',
120.        fontWeight: 'bold',
121.        textAlign: 'center',
122.        cursor: 'pointer',
123.    },
124.    buttonRegister: {
125.        width: 100,
126.        height: 40,
127.        margin: 10,
128.        border: 1px solid #ccc,
129.        borderRadius: 5,
130.        color: #007bff,
131.        background: '#fff',
132.        padding: 5,
133.        textTransform: 'uppercase',
134.        fontWeight: 'bold',
135.        textAlign: 'center',
136.        cursor: 'pointer',
137.    },
138. }
```

```
59.     middle: {
60.       flex: 3,
61.       backgroundColor: '#ffffff',
62.       borderWidth: 1,
63.       borderTopLeftRadius: 50,
64.       borderBottomRightRadius: 50,
65.     },
66.     bottom: {
67.       flex: 1,
68.     },
69.     image: {
70.       width: "70%",
71.       height: "70%",
72.       resizeMode: 'contain',
73.     },
74.     viewLogo: {
75.       flex: 1,
76.       flexDirection: 'column',
77.       alignItems: 'center',
78.       justifyContent: 'center'
79.     },
80.     viewLogin: {
81.       flex: 1,
82.       justifyContent: "space-between",
83.       marginLeft: 25,
84.       marginRight: 25,
85.       marginBottom: 25
86.     },
87.     imageBackground: {
88.       flex: 1,
89.       resizeMode: "cover",
90.       justifyContent: "center"
91.     },
92.     buttonLogin: {
93.       justifyContent: "center",
94.       alignItems: "center",
95.       backgroundColor: "#DDDDDD",
96.       borderRadius: 20,
97.       height: "25%"
98.   },
99.     buttonRegister: {
100.    justifyContent: "center",
101.    alignItems: "flex-end",
102.  },
103.   textInput: {
104.      borderRadius: 20,
105.      height: "25%",
106.      borderColor: 'gray',
107.      borderWidth: 1,
108.      paddingStart: 20
109.   }
110. });
111. export default Login;
```

บทที่ 3 Custom components, State and Props

Custom Component

ก่อนหน้านี้เราได้เรียนวิธีการสร้าง GUI ด้วย Core component ของ React Native จะเห็นได้ว่าเราสามารถใช้งาน component การสร้าง GUI อย่างง่ายดายด้วยตัวคนเดียวแต่ในบางครั้งแอพพลิเคชันที่เราต้องการสร้างมีขนาดใหญ่และมีความซับซ้อนมากต่อการพัฒนาด้วยตัวคนเดียวจะเป็นจะต้องใช้ผู้พัฒนาหลายคนแบ่งงานออกเป็นส่วนๆให้นักพัฒนาแต่ละคนทำในส่วนของตน การแบ่งงานออกเป็นเป็นส่วนๆเราจะต้องทำการประชุมวางแผนงานและกำหนด Component และ State ที่ใช้ร่วมกัน มากถึงจุดนี้หลายคนอาจมีข้อสงสัยเหตุใดต้องมีการกำหนด Component อีก เหตุใดจึงไม่แบ่งงานในลักษณะ Module หรือ Class แทน เมื่อนonding เช่นการพัฒนาด้วยเครื่องมือแบบ Native ทั้งนี้อาจกล่าวได้ว่า React-native มีพื้นฐานมาจาก ReactJS ที่ใช้ในการพัฒนา Website และมีการกำหนด Module ต่างๆในการพัฒนาเช่นเดียวกับแอพพลิเคชันทั่วๆไปแต่ทำการเรียก Module เหล่านั้นว่า Component นั้นเอง โดย Component ของ React Native สามารถแบ่งได้เป็นสองประเภทใหญ่ๆคือ Core component และ Custom Component ซึ่ง Core component เราได้ทำการเรียนไปบ้างแล้วในบทก่อนหน้านี้

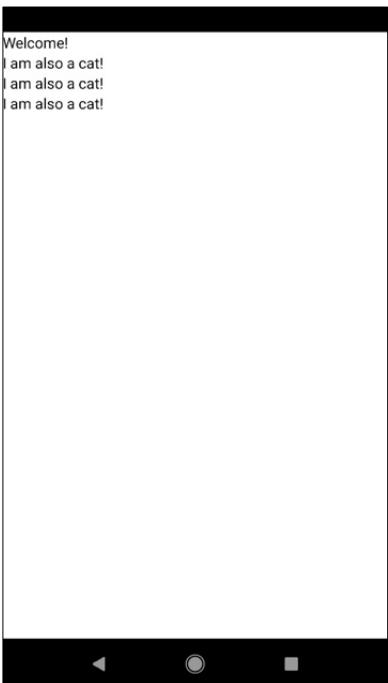
Custom Component คือ Component ที่เราสร้างขึ้นมาเองโดยจะมีลักษณะต่างจาก Core Component กล่าวคือ Custom Component อาจจะมีการสร้างโดยใช้ Core Component หลายๆตัวมาสร้างเป็นส่วนหนึ่งของแอพพลิเคชันมีการกำหนดการทำงานของ Component ไว้แล้ว แต่เมื่อมีการเรียกใช้งานจะมีวิธีการใช้งานเหมือนดังเช่น Core Component เท่าๆไป โดย Custom Component สามารถแบ่งได้ 2 ประเภทตามวิธีการสร้างได้ดังนี้

Digital Academy Thailand

Method Component

Method Component คือ Component ที่ถูกสร้างในลักษณะของ Method ซึ่งโดยทั่วๆ ไปเรามักจะสร้าง Component ในลักษณะนี้ก็ต่อเมื่อไม่มีการ Initial ค่าก่อนดำเนินการ เพราะเราไม่สามารถเรียกใช้งาน Constructor หรือ เรียกใช้งาน componentWillMount ได้

ตัวอย่างการใช้งาน Component แบบ Method



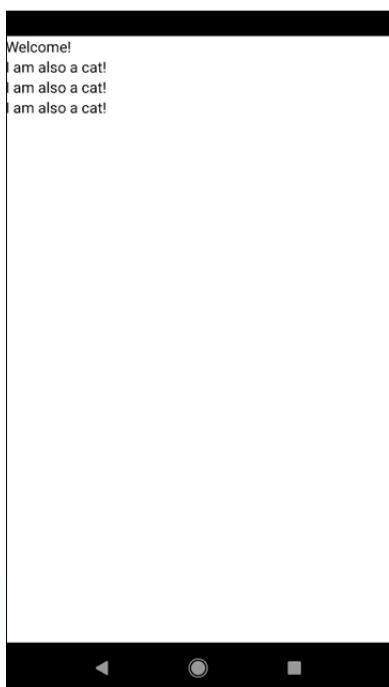
```
1. import React from 'react';
2. import { Text, TextInput, View } from 'react-native';
3.
4. const Cat = () => {
5.   return (
6.     <View>
7.       <Text>I am also a cat!</Text>
8.     </View>
9.   );
10. }
11. const Cafe = () => {
12.   return (
13.     <View>
14.       <Text>Welcome!</Text>
15.       <Cat />
16.       <Cat />
17.       <Cat />
18.     </View>
19.   );
20. }
21. export default Cafe;
```

จากตัวอย่างโปรแกรมด้านบนเมื่อทำการรันโปรแกรมผลลัพธ์ที่ได้จะแสดงดังภาพ หากเราสังเกตจะพบว่ามีส่วน ข้อความ I am also a cat! 3 บรรทัด ปรากฏขึ้นมาบนหน้าจอโทรศัพท์ ซึ่งส่วนนี้ถูกสร้างในลักษณะ Custom component ด้วย Method ที่มีชื่อว่า Cat ในการเรียกใช้งาน Custom Component แบบ Method ที่เราสร้างขึ้น สังเกตในไฟล์ App.js เราไม่ได้ทำการสร้าง Method แบบแยกไฟล์ก็จะสามารถเรียกใช้งานได้ทันที แต่ถ้าหากเราสร้างแบบแยกไฟล์เมื่อไหร่เรา จะต้องทำการ Import Method จากไฟล์ที่เราสร้างก่อนจึงจะสามารถใช้งานได้ การเรียกใช้งานเราจะทำการ insert tag ตามชื่อ method ที่เราสร้างไว้ตามตัวอย่าง Code บรรทัดที่ 15 เราจะมีการเรียกใช้งาน component Cat ที่เราสร้างขึ้น

Class Component

Class Component คือ Component ที่ถูกสร้างในลักษณะแบบ Class ซึ่งโดยทั่วไปเรามักจะสร้าง Component ในลักษณะนี้ก็ต่อเมื่อมีการ Initial ค่าก่อนดำเนินการผ่าน Constructor หรือเรียกใช้งาน componentWillMount ได้หรือเพื่อความสะดวกในการแบ่งงานกันทำงานเป็นทีมการแบ่งงานในลักษณะ Class component จะจ่ายต่อการแบ่งส่วนของงานเมื่องานมีลักษณะใหญ่มาก การสร้าง component แบบ Class จะมีวิธีการที่ยุ่งยากกว่าการสร้างแบบ method compartment ซึ่งเราจะดูต่อไปใน Life Cycle ของตัว react-native นี้จะอธิบายในบทถัดไป

ตัวอย่างการใช้งาน Component แบบ Class



```
1. import React, { Component } from "react";
2. import { Text, TextInput, View } from 'react-native';
3.
4. class Cat extends Component {
5.   render() {
6.     return (
7.       <Text>Hello, I am your cat!</Text>
8.     );
9.   }
10. }
11.
12. const Cafe = () => {
13.   return (
14.     <View>
15.       <Text>Welcome!</Text>
16.       <Cat />
17.       <Cat />
18.       <Cat />
19.     </View>
20. );
21. }
22.
23. export default Cafe;
```

จากตัวอย่างโปรแกรมด้านบนเมื่อทำการรันโปรแกรมผลลัพธ์ที่ได้จะแสดงดังภาพ หากเราสังเกตจะพบว่ามีส่วนข้อความ I am also a cat! 3 บรรทัด ปรากฏขึ้นมาบนหน้าจอโทรศัพท์เข่นเดียวกับการสร้าง Component ด้วย Method ก่อนหน้า ซึ่งส่วนนี้ถูกสร้างในลักษณะ Custom component ด้วย Class ที่มีชื่อว่า Cat การสร้าง Component ในลักษณะนี้จะต้องมีการ Extend Component Class ที่อยู่ใน react library และต้องสร้างส่วนแสดงผลภายใต้ Render function ในการเรียกใช้งานComponent แบบ Class ที่เราสร้างขึ้นนี้จะเหมือนกับการเรียกใช้งานแบบ Method โดยสังเกตในไฟล์ App.js เราไม่ได้ทำการสร้าง Class แบบแยกไฟล์ก็จึงสามารถเรียกใช้งานได้ทันที แต่ถ้าหากเราสร้างแบบแยกไฟล์เมื่อไหร่เราจะต้องทำการ Import Class จากไฟล์ที่เราสร้างก่อนจึงจะสามารถใช้งานได้ การเรียกใช้งานเราจะทำการ insert tag ตามชื่อ method ที่เราสร้างไว้ตามตัวอย่าง Code บรรทัดที่ 16 เราจะมีการเรียกใช้งาน component Cat ที่เราสร้างขึ้น

Props

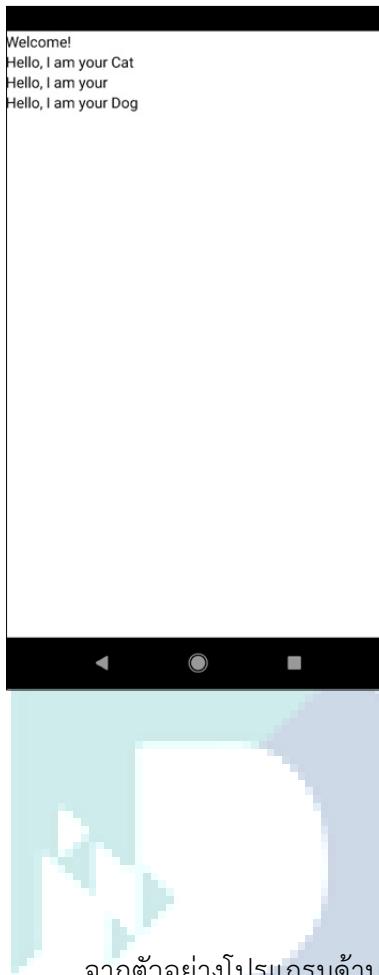
Props ย่อมาจาก Properties คือ ตัวกำหนดคุณลักษณะต่างๆของ Component หรือเป็นตัวที่ใช้ในการส่งค่าพารามิเตอร์ต่างๆไปยัง Component จาก Parent ไปยัง Children ซึ่งไม่สามารถทำการเปลี่ยนแปลงได้ในขณะที่ Run อยู่ ในบทก่อนหน้านี้เรื่องการสร้าง GUI เราได้มีการใช้งาน Props กับ Core Component ที่ใช้ในการสร้าง UI ไปแล้วหากเราสังเกตดีๆจะพบว่าทุกครั้งที่เรามีการเรียกใช้งาน UI เราก็จะมีการกำหนดค่าให้มันเสมอๆอย่างน้อยที่สุดก็จะมีการเรียกใช้งาน Pops Style ที่ใช้ในการปรับแต่ง GUI โดยในเนื้อหาส่วนนี้เราจะมาพูดถึงการใช้งาน Pops ใน Component ที่เราสร้างขึ้น

การใช้งาน Props ใน component ที่เราสร้างขึ้นโดยมากเรามักจะใช้เพื่อส่งค่าพารามิเตอร์ไปยัง Component ซึ่งเราสามารถส่งค่าพารามิเตอร์ต่างๆได้ดังนี้ Integer, Float, String, Object, Array หรือแม้แต่ Function ซึ่งการส่งค่า Props ด้วย Type ที่เราต้องการเราจะต้องทำการ Import ไลบรารี 'prop-types' ก่อนจึงจะสามารถใช้งานได้ โดยค่า Props ที่ถูกส่งไปยัง Component และเราจะไม่สามารถทำการแก้ไขค่า Pops นั้นได้หรือจะเรียกว่า Pops มีคุณสมบัติเป็น Read Only นั้นเอง

ตัวอย่างการส่งค่า Props Type ต่างๆ

Type	ตัวอย่าง
String	<MyComponent myPropsString={"Send data"} />
Float	<MyComponent myPropsFloat={3.42} />
Integer	<MyComponent myPropsInteger={342} />
Object	let obj = {id:1, name: 'i am object'}; <MyComponent myPropsObject={obj} />
Array	let arr = [{id:1, name: 'i am array'}]; <MyComponent myPropsObject={arr} />
Function	iAmFunction() { // todo something ... } <MyComponent myPropsFunction={this.iAmFunction.bind(this)} />

ตัวอย่างการใช้งาน Props



```
1. import React, { Component } from "react";
2. import { Text, TextInput, View } from 'react-native';
3. import PropTypes from 'prop-types'
4. class Animal extends Component {
5.   render(props) {
6.     return (
7.       <Text>Hello, I am your
8.         {this.props.AnimalType}</Text>
9.     );
10.   }
11. Animal.PropTypes = {
12.   AnimalType: PropTypes.string
13. };
14. const ShowAnimal = () => {
15.   return (
16.     <View>
17.       <Text>Welcome!</Text>
18.       <Animal AnimalType="Cat"/>
19.       <Animal />
20.       <Animal AnimalType="Dog"/>
21.     </View>
22.   );
23. }
24. export default ShowAnimal;
```

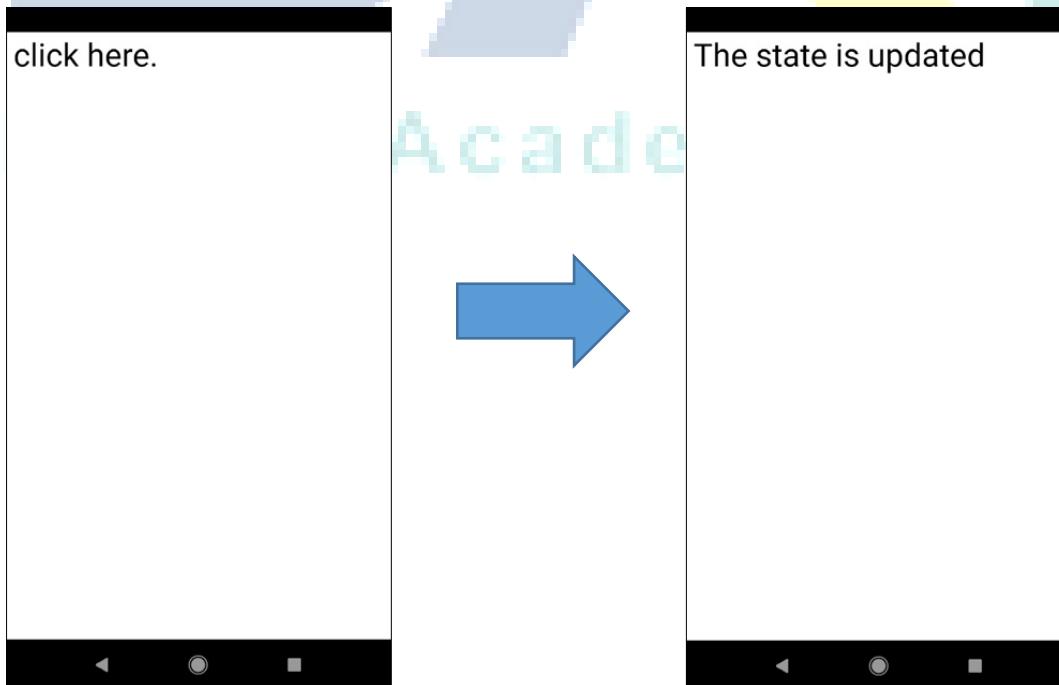
จากตัวอย่างโปรแกรมด้านบนเมื่อทำการรันโปรแกรมผลลัพธ์ที่ได้จะแสดงดังภาพ หากเราสังเกตจะพบว่ามีส่วนข้อความที่มีลักษณะต่างกัน ซึ่งเกิดจากการส่ง Props ที่มีพารามิเตอร์ที่แตกต่างกัน จากตัวอย่างโค้ดด้านบนมีการสร้าง Component ในลักษณะที่เป็นแบบ Class component ที่มีชื่อว่า Animal โดย Animal จะมีการแสดงข้อความที่ดึงมาจากค่าของ Props โดยค่าที่ถูกส่งผ่านมาใน Render function ในบรรทัดที่ 5 render(props) เราสามารถเรียกใช้งานได้โดยใช้คำสั่ง this.props.PropName โดย Props Name คือชื่อของ Props ที่เราสร้างขึ้น จากตัวอย่างโปรแกรมบรรทัดที่ 7 เราเรียกใช้งาน Props ที่ถูกสร้างขึ้นในบรรทัดที่ 12 คือ this.props.AnimalType หลังจากเราทำการสร้าง Props เรียบร้อยแล้วเราจะสามารถใช้งาน Props หรือจะไม่ใช้งานก็ได้ หากเราต้องการใช้งาน Props เราจะต้องเรียกชื่อ Props พร้อมทั้งส่งค่าพารามิเตอร์ให้ถูก โดยบรรทัดที่ 18 เราจะทำการเรียกใช้งาน Props ที่เราสร้างขึ้นโดยส่งค่าพารามิเตอร์มีครับเป็น String

State

State ค่าพารามิเตอร์ที่ใช้เก็บค่าต่างๆใน Component สามารถทำการเปลี่ยนแปลงได้ในขณะที่ Run อยู่ ซึ่งจะแตกต่างจาก Props ก่อนหน้านี้ที่ไม่สามารถแก้ไขได้ โดยทั่วไปการสร้างแต่เราสามารถสร้างได้หลายวิธีแต่วิธีที่ได้รับความนิยมและเป็นสากลที่สุดก็คือการสร้าง Stage ภายใน Constructor สังเกต react Life Cycle หากสร้างบริเวณอื่นจะต้องมีการ Re Render ใหม่ซึ่งจะเสีย Process Time ไปเพล่าๆ โดยการใช้งานใน Application ส่วนใหญ่เรามักจะใช้ Stage ควบคู่ Event หรือควบคู่ Thread Listener ต่างๆ ที่ต้องการแสดงค่าอัพเดทค่าลงบน UI เมื่อมีการเปลี่ยนแปลง การใช้งานเสร็จโดยมากเราสามารถใช้งานได้แบบตรงๆ หรือจะใช้งานผ่านตัว State Management จำพวก Redux หรือ Mobx ก็ได้โดยในเนื้อหาหนึ่งอธิบายและกล่าวถึงการใช้งาน State แบบตรงๆ ส่วนการใช้งานแบบผ่าน state management จะกล่าวถึงในบทถัดไป

การใช้งาน Stage โดยมากเราจะทำการประกาศไว้ใน Constructor โดยตัว State เองจะมี Type เป็น object ค่าพารามิเตอร์อื่นๆอยู่ภายในซึ่งเราสามารถกำหนดค่าของ Step หรือ initial ค่าภายในนี้ได้ทันที แต่ถ้าหากต้องการเปลี่ยนแปลงค่า Stage เราจะต้องใช้คำสั่ง `this.setState({YourState:Parameter})` ในการเปลี่ยนแปลง การแสดงค่าหรือเรียกใช้งาน Parameter ภายใน State เรา自身สามารถทำได้เหมือนกับการใช้งาน object ทั่วๆไปคือ `this.state.your_parameter`

ตัวอย่างการใช้งาน State กับ Text



ตัวอย่าง Code

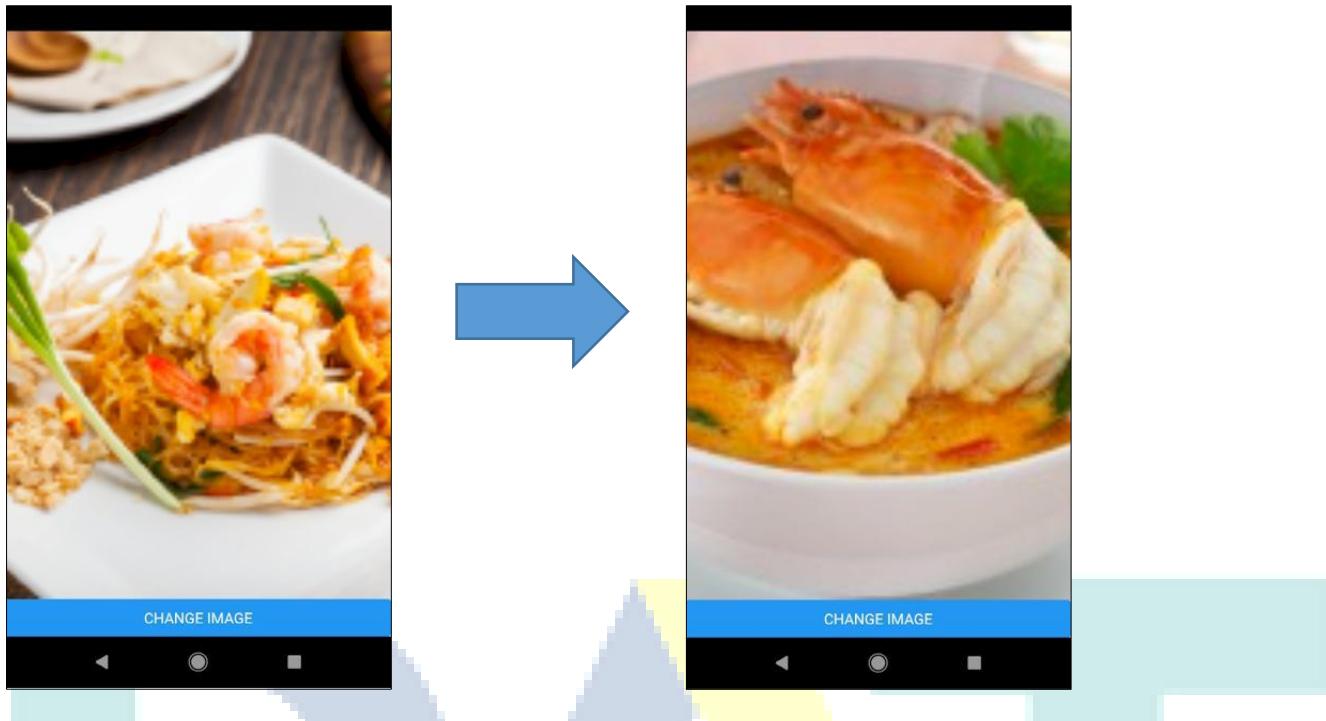
```
1. import React, {Component} from 'react';
2. import { Text, View } from 'react-native';
3.
4. export default class App extends Component {
5.     constructor(props) {
6.         super(props);
7.         this.state = {
8.             myState: 'click here.'
9.         };
10.    }
11.    updateState = () => this.setState({myState: 'The state is updated'})
12.    render() {
13.        return (
14.            <View>
15.                <Text onPress={this.updateState}> {this.state.myState} </Text>
16.            </View>
17.        );
18.    }
19. }
```

จากตัวอย่างโปรแกรมด้านบนทำการสร้างแอพพลิเคชั่นง่ายๆโดยมีการทำงานเมื่อกดตัวอักษรคำว่า Click here จะทำการเปลี่ยนตัวอักษรเป็นคำว่า The state is updated โดยตัวอย่างมีการสร้าง constructor และมีการประกาศ State โดยมีพารามิเตอร์ชื่อว่า myState มี Type เป็น String อยู่ภายใน และมีการกำหนดค่าเริ่มต้นคือ 'click here.' โดยมีกระบวนการทำงานตาม Event ดังนี้

1. ใน Tag <Text> มีการสร้าง Event onPress เมื่อทำการกดปุ่มจะทำการเรียกฟังก์ชัน updateState และทำงาน set ค่า Stage ใหม่เป็นคำว่า 'The state is updated'

Digital Academy Thailand

ตัวอย่างการใช้งาน State กับ Image



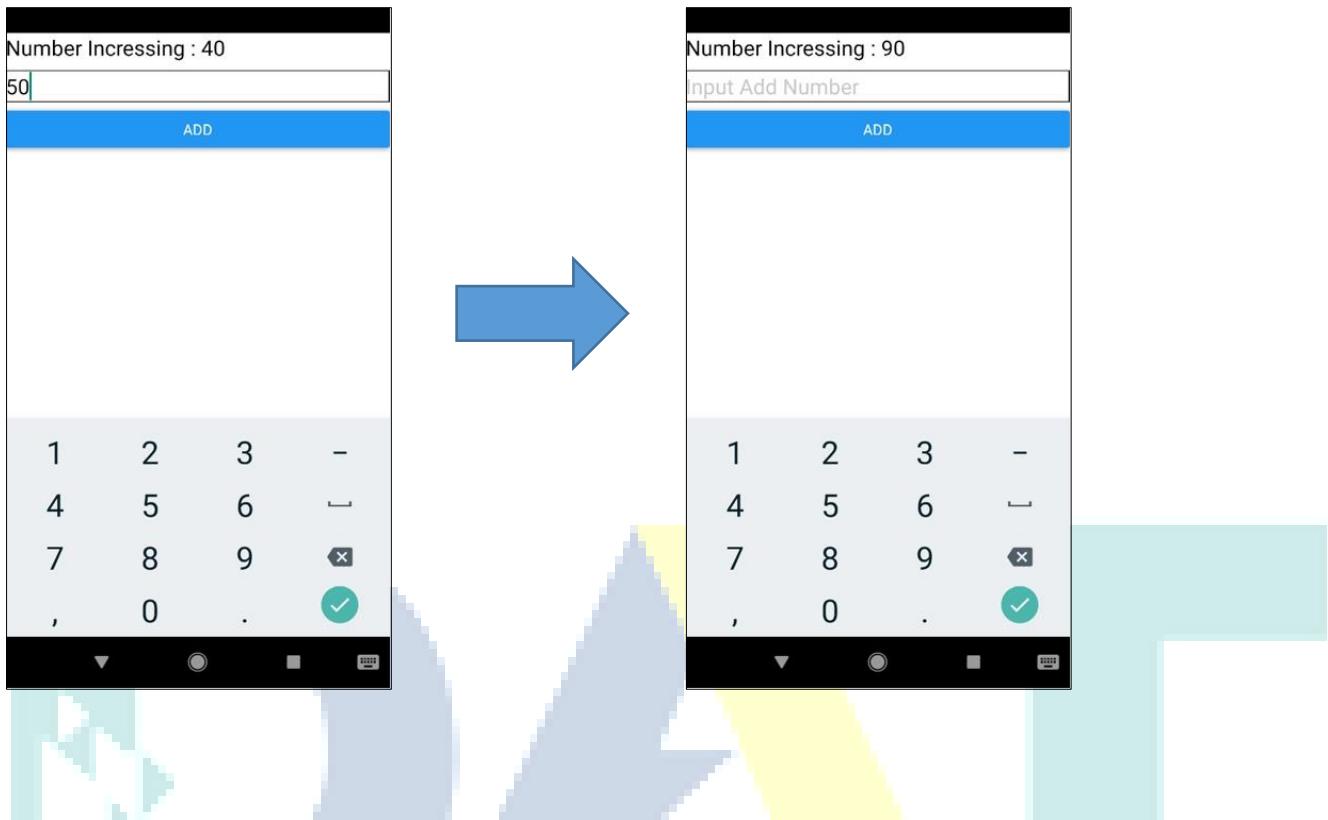
ตัวอย่าง Code

```
1. import React, {Component} from 'react';
2. import {View, Button, Image} from 'react-native';
3. export default class App extends Component {
4.   constructor() {
5.     super()
6.     this.state = {
7.       url: 'https://sv1.picz.in.th/images/2020/08/02/E9qb0Z.th.png'
8.     };
9.   }
10.  render() {
11.    return (
12.      <View style={{flex:1}}>
13.        <View style={{flex:1}}>
14.          <Image source={{uri:this.state.url}} style={{flex:1}}/>
15.        </View>
16.        <Button title="Change Image" onPress={()=>this.setState({url:'https://sv1.picz.in.th/images/2020/08/02/E9qA9e.th.jpg'})}/>
17.      </View>
18.    );
19.  }
}
```

จากตัวอย่างโปรแกรมด้านบนทำการสร้างแอพพลิเคชั่นง่ายๆโดยมีการทำงานเมื่อกดปุ่ม Change Image จะทำการเปลี่ยนภาพใหม่ โดยตัวอย่างมีการสร้าง constructor และมีการประกาศ State โดยมีพารามิเตอร์ชื่อว่า url มี Type เป็น String อัญญาณ และมีการกำหนดค่าเริ่มต้นคือคือลิงค์ URL ของภาพ โดยมีกระบวนการทำงานตาม Event ดังนี้

1. Tag <Button> มีการสร้าง Event onPress เมื่อทำการกดปุ่มจะทำงาน set State Parameter url ใหม่ด้วย String ที่เป็น URL ของภาพตัวใหม่

ตัวอย่างการใช้งาน State กับ TextInput



ตัวอย่าง Code

```
1. import React, {Component} from 'react';
2. import {View, Button, Image, TextInput, Text } from 'react-native';
3. export default class App extends Component {
4.   constructor() {
5.     super()
6.     this.state = {
7.       number:0,
8.       addNumber:0
9.     };
10.   }
11.
12.   calculate = ()=>{
13.     this.setState({number:(this.state.number+this.state.addNumber)})
14.   }
15.
16.   render() {
17.     return (
18.       <View style={{flex:1}}>
19.         <Text style={{fontSize:20,marginBottom:8}}>Number Increasing : {this.state.number}</Text>
20.         <TextInput style={{fontSize:20,borderWidth:1,marginBottom:8}} placeholder="Input Add Number" keyboardType="number-pad" onChangeText={text=>this.setState({addNumber:Number(text)})}/>
21.         <Button title="Change Image" onPress={this.calculate}/>
22.       </View>
23.     );
24.   }
}
```

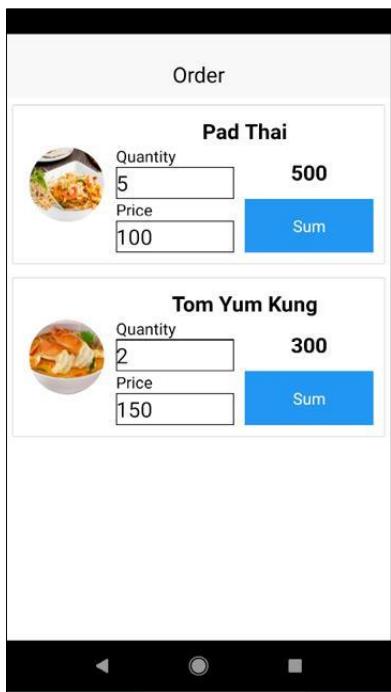
จากตัวอย่างโปรแกรมด้านบนทำการสร้างแอพพลิเคชันง่ายๆโดยมีการทำงานเมื่อ Input ตัวเลขลงใน TextInput และทำการกดปุ่มจะทำการบวกค่าเพิ่มขึ้นเรื่อยๆแล้วจึงทำการแสดงผลทางหน้าจอ โดยตัวอย่างมีการสร้าง constructor และ มีการประกาศ State โดยมีพารามิเตอร์ชื่อว่า number และ addNumber มี Type เป็น Number อยู่ภายใน การทำงานใน Tag <Text> มีการแสดงค่าของ number โดยมีกระบวนการทำงานตาม Event ดังนี้

1. Tag<TextInput> มีการสร้าง Event onChangeText เมื่อมีการป้อนตัวเลขลงไปจะทำการเซตค่า ไปยัง addNumber
2. Tag <Button> มีการสร้าง Event onPress เมื่อมีการกดปุ่มจะทำการเรียก Function calculate เพื่อทำการบวกค่าระหว่าง number และ addNumber

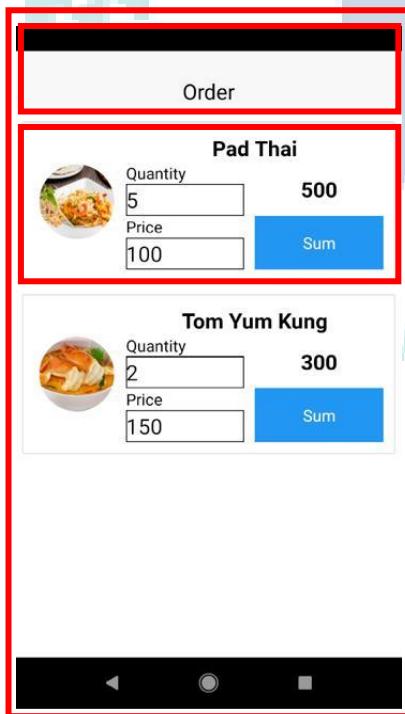


Digital Academy Thailand

ตัวอย่างการใช้งาน Custom component, State and Props

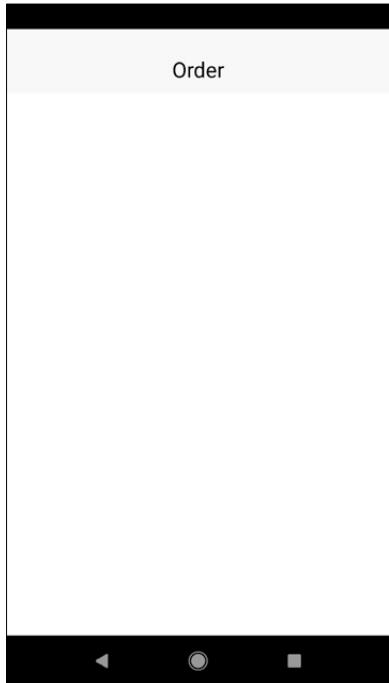


ตัวอย่างการใช้งานนี้จะไม่อธิบายวิธีการสร้าง ui กระบวนการและการและวิธีการในการใช้งานคอสตอม component Stage และProps ตัวอย่าง Application นี้จะทำการสร้าง List รายการอาหารเพื่อใช้สำหรับการสั่งอาหารและคำนวนราคารวมแต่ละรายการโดยผู้ใช้งานสามารถกรอกจำนวนอาหารที่ต้องการพร้อมราคา

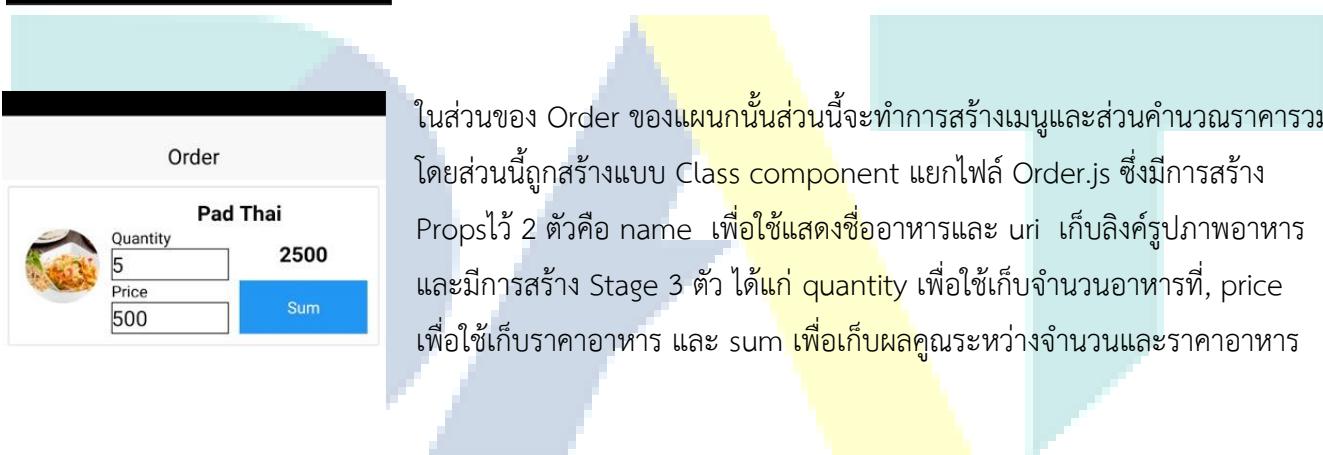


โดยการทำงานเบื้องต้นเราจะต้องทำการวางแผนแบ่งงานออกเป็นส่วนๆเรื่องง่ายต่อการพัฒนาโดย Application นี้เราจะทำการแบ่ง component ออกเป็น 3 ส่วนได้แก่

1. Main compartment ไฟล์งาน App.js ทำหน้าที่รวมทุกๆ component และแสดงผลโดยรวม
2. Header component ไฟล์งาน Header.js เป็น component ที่ใช้แสดง Header หรือ Title Bar นั่นเอง
3. Order component ไฟล์งาน Order.js เป็นส่วนที่ใช้ในการแสดง เม뉴อาหารและคำนวนผลคูณระหว่างจำนวนและราคา



ในส่วนของ Header component ส่วนนี้จะทำการสร้าง Title Bar โดยถูกสร้างแบบ Method Component แบบแยกไฟล์ Header.js โดยทำการสร้าง Props headerText เพื่อใช้แสดงข้อความใน Title Bar ในบรรทัดที่ 9 และถูกเรียกใช้งานจากไฟล์ App.js โดยมีการ Import Header ในบรรทัดที่ 5 มีการกำหนดค่า headerText="Order ในบรรทัดที่ 11



ในส่วนของ Order ของแผนกนั้นส่วนนี้จะทำการสร้างเมนูและส่วนคำนวณราคาร่วมโดยส่วนนี้ถูกสร้างแบบ Class component และไฟล์ Order.js ซึ่งมีการสร้าง Props ไว้ 2 ตัวคือ name เพื่อใช้แสดงชื่ออาหารและ uri เก็บลิงค์รูปภาพอาหาร และมีการสร้าง Stage 3 ตัว ได้แก่ quantity เพื่อใช้เก็บจำนวนอาหารที่, price เพื่อใช้เก็บราคาอาหาร และ sum เพื่อเก็บผลคูณระหว่างจำนวนและราคาอาหาร



Academy Thailand

ตัวอย่างโค้ด

```
1. //App.js
2. import * as React from 'react';
3. import { Text, View, StyleSheet } from 'react-native';
4. import Constants from 'expo-constants';
5. import Header from './Header';
6. import Order from './Order';
7.
8. export default function App() {
9.   return (
10.     <View style={styles.container}>
11.       <Header headerText="Order"/>
12.       <Order name='Pad Thai'
13.         uri='https://sv1.picz.in.th/images/2020/08/02/E9qb0Z.th.png' />
14.       <Order name='Tom Yum Kung'
15.         uri='https://sv1.picz.in.th/images/2020/08/02/E9qA9e.th.jpg' />
16.     );
17.   const styles = StyleSheet.create({
18.     container: {
19.       flex: 1,
20.       backgroundColor: '#ecf0f1',
21.     },
22.   });
}
```



Digital Academy Thailand

```
1. //Header.js
2. import React from 'react';
3. import PropTypes from 'prop-types'
4. import {Text, View, StyleSheet} from 'react-native';
5.
6. const Header = (props) =>{
7.     return(
8.         <View style={styles.viewStyle}>
9.             <Text style={styles.textStyle}>{props.headerText}</Text>
10.        </View>
11.    );
12. };
13.
14. Header.PropTypes = {
15.     headerText: PropTypes.string.isRequired
16. };
17.
18. const styles = StyleSheet.create({
19.     viewStyle:{
20.         backgroundColor: '#F8F8F8',
21.         justifyContent : 'center',
22.         alignItems:'center',
23.         height:60,
24.         paddingTop:15,
25.         shadowColor:'#000000',
26.         shadowOffset:{width:0 ,height:2},
27.         shadowOpacity:0.2,
28.         position:'relative'
29.     },
30.     textStyle :{
31.         fontSize : 20
32.     }
33. });
34. export default Header;
```

Digital Academy Thailand

```

1. //Order.js
2. import React, { Component } from 'react';
3. import PropTypes from 'prop-types';
4. import {
5.   Text,
6.   View,
7.   StyleSheet,
8.   Image,
9.   TextInput,
10.  TouchableOpacity,
11. } from 'react-native';
12.
13. export default class Order extends Component {
14.   state = {
15.     quantity: 1,
16.     price: 1,
17.     sum: 0,
18.   };
19.
20.   calculate = () => {
21.     this.setState({ sum: this.state.quantity * this.state.price });
22.   };
23.
24.   render(props) {
25.     return (
26.       <View style={styles.containerStyle}>
27.         <View style={{ flex: 1, justifyContent: 'center' }}>
28.           <Image style={styles.icon} source={{ uri: this.props.uri }} />
29.         </View>
30.         <View style={{ flex: 3, flexDirection: 'column', marginLeft: 6 }}>
31.           <View style={{ flexDirection: 'column' }}>
32.             <Text
33.               style={{ fontSize: 20, alignSelf: 'center', fontWeight: 'bold' }}
34.               >{this.props.name}</Text>
35.           </View>
36.           <View style={{ flexDirection: 'row' }}>
37.             <View style={{ flex: 1, flexDirection: 'column' }}>
38.               <Text style={{ fontSize: 15 }}>Quantity</Text>
39.               <TextInput
40.                 style={styles.input}
41.                 keyboardType="number-pad"
42.                 onChangeText={text =>
43.                   this.setState({ quantity: text })
44.                 }
45.               />
46.             </View>
47.             <Text style={{ fontSize: 15 }}>Price</Text>
48.             <TextInput
49.               style={styles.input}
50.               keyboardType="number-pad"
51.               onChangeText={text => this.setState({ price: text }) }
52.             />
53.           </View>
54.           <View style={{ flex: 1, flexDirection: 'column' }}>
55.             <View
56.               style={{
57.                 flex: 1,
58.                 justifyContent: 'center',
59.                 alignItems: 'center',
60.               }}>
61.               <Text style={{ fontSize: 20, fontWeight: 'bold' }}>
62.                 {this.state.sum}
63.               </Text>

```

```
64.          </Text>
65.        </View>
66.
67.      <View style={{ flex: 1 }}>
68.        <TouchableOpacity
69.          style={{
70.            flex: 1,
71.            width: '100%',
72.            backgroundColor: '#2196F3',
73.            justifyContent: 'center',
74.            alignItems: 'center',
75.          }}
76.          onPress={this.calculate}>
77.          <Text style={{ fontSize: 15, color: 'white' }}>Sum</Text>
78.        </TouchableOpacity>
79.      </View>
80.    </View>
81.  </View>
82.  </View>
83. </View>
84. );
85. }
86. }
87.
88. Order.PropTypes = {
89.   name: PropTypes.string.isRequired,
90.   uri: PropTypes.string.isRequired,
91. };
92.
93. const styles = StyleSheet.create({
94.   containerStyle: {
95.     flexDirection: 'row',
96.     borderWidth: 1,
97.     borderRadius: 2,
98.     borderColor: '#DDDDDD',
99.     margin: 6,
100.    padding: 10,
101.  },
102.  icon: {
103.    height: 70,
104.    width: 70,
105.    borderRadius: 35,
106.    alignSelf: 'center',
107.  },
108.  input: {
109.    borderWidth: 1,
110.    marginEnd: 10,
111.    fontSize: 20,
112.  },
113. });

```

บทที่ 4 การเขียนโปรแกรมภาษา JavaScript

ประวัติความเป็นมา

จา瓦สคริปต์ (JavaScript) เป็นภาษาสคริปต์ ที่มีลักษณะการเขียนแบบโพโรโทไทป์ (Prototyped-based Programming) ส่วนมากใช้ในหน้าเว็บเพื่อประมวลผลข้อมูลที่ผ่านของผู้ใช้งาน แต่ก็ยังมีใช้เพื่อเพิ่มเติมความสามารถในการเขียนสคริปต์โดยฝีมือในโปรแกรมอื่น ๆ พัฒนาโดย Brendan Eich พนักงานบริษัทเน็ตสเคป โดยขณะนั้น JavaScript ใช้ชื่อว่า Mocha และภายหลังได้เปลี่ยนชื่อมาเป็น ไลท์สคริปต์ และเป็น JavaScript ในปัจจุบัน JavaScript เริ่มจากเป็นภาษาสคริปเล็กๆ ที่ทำงานเพื่อเพิ่มลูกเล่นนิดหน่อยบนเบราว์เซอร์จนกระทั่งได้รับความนิยมอย่างแพร่หลายได้ถูกนำไปใช้งานในหลายงานอย่างกว้างขวาง แต่ตัวภาษาดังเดิมนั้นออกแบบมาอย่างไม่ดีมากนักเมื่อนำไปสร้างโปรแกรมขนาดใหญ่ ที่ซับซ้อน มักจะเกิดปัญหาจึงได้มีการปรับปรุงพัฒนาเพิ่มฟีเจอร์ที่ภาษาอยู่คู่ใหม่ควรจะทำได้ลงไป โดยถ้าเราไปอ่านรายละเอียดจะพบว่าชื่อรุ่นของ JavaScript นั้นจะใช้เป็น ES5, ES2015, ES6 เป็นต้น โดยตอนนี้ JavaScript มีบริษัท Oracle เป็นเจ้าของและได้รับการรับรองมาตรฐานจาก ECMA ซึ่งเป็นองค์กรกำหนดมาตรฐานของภาษา Programming ต่างๆ ส่วนของภาษา JavaScript จะใช้ชื่อว่า ECMAScript

JavaScript กับ React-Native

เป้าหมายแรกในการสร้าง JavaScript ขึ้นมาคือใช้ในการจัดการและควบคุม HTML ในเว็บเพจ ซึ่งใน React-Native นี้เราจะมีกลักษณะคล้ายคลึงกันคือใช้ในการควบคุมโลจิกของโปรแกรมแต่มีได้ใช้ร่วมกับ HTML แต่อย่างใด เพราะ Component ที่ถูกสร้างใน React-Native จะถูกสร้างด้วย Stylesheet ที่มีลักษณะเหมือนกับ CSS จึงอาจกล่าวได้ว่าเราใช้ภาษา JavaScript เพื่อจัดการ, ควบคุม Component และกำหนดโลจิกกระบวนการทำงานต่างใน Application ของเรา

Variable ตัวแปร

การทำงานของโปรแกรมคอมพิวเตอร์ไม่ว่าจะเป็นโปรแกรมชนิดใดก็ตาม สรุนใหญ่แล้วมักมีขั้นตอนและกลไกการทำงานเป็นแบบอย่างเฉพาะตัว โดยเฉพาะเรื่องเกี่ยวกับรูปแบบของภาษา การใช้ประโยชน์ข้อความสั่งการใช้ข้อมูลและตัวแปร ฯลฯ ภาษา JavaScript ก็เช่นกัน โดย JavaScript มีระบบตัวแปรแบบ Dynamic Typing หรือชนิดตัวแปรจะเป็นอะไรก็ได้เปลี่ยนแปลงได้ตลอดตามค่าที่ตัวมันเก็บเหมือนกับภาษา Python หรือ Ruby ซึ่งแตกต่างจากภาษาที่มีลักษณะที่เป็นแบบ Static Typing เช่นภาษา C หรือ Java ที่ต้องประกาศและระบุชนิดข้อมูลของตัวแปลบทันทีและไม่สามารถเปลี่ยนแปลง Type ของตัวแปลนั้นๆได้ ซึ่งตัวแปรในภาษา JavaScript เองสามารถแบ่งได้ 2 ประเภทใหญ่ๆคือ ชนิดตัวแปรมาตรฐาน(Primitive Type) และตัวแปรแบบวัตถุ(Object)

1. ตัวแปรมาตรฐาน(Primitive Type)

ตัวแปรมาตรฐานคือตัวแปรมูลฐานสุดในภาษา JavaScript ซึ่งประกอบด้วยตัวแปร 3 แบบดังนี้

ชนิดข้อมูล	การเก็บข้อมูล
number	ข้อมูลชนิดตัวเลข ประกอบด้วย เลขจำนวนเต็ม (Integer) และเลขจำนวนจริง (Floating)
logical	ข้อมูลทางตรรกะ มี 2 สถานะ คือจริง (True) และเท็จ (False)
String	ข้อมูลที่เป็นข้อความซึ่งจะต้องกำหนดไว้ในเครื่องหมาย ("...") หรือ ('...')

1.1. Number ชนิดตัวแปรแบบตัวเลขของ JavaScript นั้นใช้ระบบแบบทศนิยมขนาด 64-bit หรือเท่ากับ double ของภาษาตระกูลซี ดังนั้นการเก็บจะมีขนาดที่จำกัดเช่นเดียวกัน

ตัวอย่างการใช้งาน Number

```
var x;  
x = 18;  
x = 18.75;  
x = 245.2e-50;  
x = 0xff;  
x = Math.random();
```

1.2. String คือตัวแปรแบบสายอักษรและเนื่องด้วยภาษานี้ไม่มี char การประกาศ string ขึ้นมาใช้เลยเลือกได้เลยว่าจะใช้ ' (single quote) หรือ " (double quote)

ตัวอย่างการใช้งาน String

```
var a = 'i have an apple';  
var b = "It's gonna be OK.";  
var c = 'นี่คือ \'JS\''
```

1.3. Boolean คือตัวแปรที่เก็บค่าความจริงที่มีค่า true หรือ false เท่านั้น แต่เนื่องจากตัวภาษาเป็น dynamic type การเช็คว่าตัวแปรเก็บค่าเท่ากันหรือไม่อาจจะมีไม่เพียงพอเนื่องด้วยอาจจะมี Type คนละแบบกันจึงมี == และ != เพื่อใช้เช็คค่าและ type พร้อมๆกันเพิ่มเติมจากเดิม == และ != ที่ใช้เช็คเฉพาะค่าเท่านั้น

ตัวอย่างการใช้งาน Boolean

```

var a = true;
var b = false;
a==b // false
12 == '12' // true
12 === '12' // false
12 != '12' // false
12 !== '12' // true

```

2. ตัวแปรแบบวัตถุ(Object)

ตัวแปรแบบวัตถุคือตัวแปรที่มีการเก็บข้อมูลในลักษณะเป็นกลุ่มข้ออาจจะมีความสัมพันธ์กันหรือไม่สัมพันธ์กันก็ได้โดยแบ่งได้เป็น 2 แบบ

ชนิดข้อมูล	การเก็บข้อมูล
array	ชุดข้อมูลที่มี Type เดียวกัน
object	ชุดข้อมูลที่มีความสัมพันธ์กันมีการเรียกใช้งานในลักษณะ key-value

2.1. ตัวแปรชุด (Array) คือตัวแปรซึ่งมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงเพียงชื่อเดียว ด้วยการใช้หมายเลขอffset เป็นตัวจำแนกความแตกต่างของค่าตั้งแปรแต่ละตัว ถ้าเราจะเรียกตัวแปรชนิดนี้ว่า "ตัวแปรชุด" ก็เห็นจะไม่ผิดนัก ตัวแปรชนิดนี้มีประโยชน์มาก ลองคิดถึงค่าข้อมูลจำนวน 100 ค่า ที่ต้องการเก็บไว้ในตัวแปรจำนวน 100 ตัว อาจทำให้ต้องกำหนดตัวแปรที่แตกต่างกันมากถึง 100 ชื่อ กรณีอย่างนี้ควรจะทำอย่างไรดี แต่ด้วยการใช้คุณสมบัติ array เราสามารถนำตัวแปรหลาย ๆ ตัวมาอยู่ร่วมเป็นชุดเดียวกันได้ และสามารถเรียกใช้ตัวแปรทั้งหมดโดยระบุผ่านชื่อเพียงชื่อเดียวเท่านั้น ด้วยการระบุหมายเลขอffset หรือ ดัชนี(index) กำกับตามหลังชื่อตัวแปร ตัวแปรเพียงชื่อเดียว จึงมีความสามารถเทียบได้กับตัวแปรนับร้อยตัว พันตัว (ตัวที่ 1) ในตัวแปรแบบอาร์เรย์มีดัชนีเป็น 0 ส่วนตัวแปรต่อๆ ไปก็จะมีดัชนีเป็น 1,2,3,... ไปตามลำดับ เมื่อต้องการระบุชื่อตัวแปรแบบอาร์เรย์แต่ละตัว ก็จะใช้รูปแบบ name[0], name[1],... เรียงต่อกันไปเรื่อยๆ เราสามารถสร้างตัวแปรอาร์เรย์ใหม่ด้วยได้หลายวิธี

ตัวอย่างการใช้งาน Array

```

var arr = new Array();
arr[0] = 'a';
arr[1] = 'b';
// ["a", "b"]

// กำหนดขนาดเริ่มต้นก่อนก็ได้
arr = new Array(3); // [undefined, undefined, undefined]

// แต่ถ้าใส่ไปมากกว่า 1 ตัวจะเป็นการกำหนดสมาชิกเริ่มต้นแทน
arr = new Array(1, 2); // [1, 2]

// หรืออปชัน
arr = [1, 2]; // [1, 2]

console.log(typeof arr); // object
console.log(arr instanceof Array); // true

```

การเติมหรือลบข้อมูลเข้าไปใน array นอกจากกำหนด index เมื่อนั่งภาษาอื่นๆ เราสามารถใช้คำสั่งใช้คำสั่งพิเศษ push-pop / shift-unshift เอาไว้เพิ่ม/ลดข้อมูลใน array โดย push-pop จะใช้ในการเพิ่มและลบข้อมูลด้านท้าย(tail) ใน Array ส่วน shift-unshift จะใช้เพิ่มลบข้อมูลด้านหน้า (head)

ตัวอย่างการใช้งานชุดคำสั่ง Array

```
var arr = [1, 2, 3, 4];  
  
x = arr.pop(); // x = 4, arr = [1, 2, 3]  
arr.push(5); // arr = [1, 2, 3, 5]  
  
x = arr.shift(); // x = 1, arr = [2, 3, 5]  
arr.unshift(0); // arr = [0, 2, 3, 5]
```

2.2. ตัวแปรรัตตุ(Object) ในภาษา JavaScript ตัวแปร object จะมีลักษณะคล้ายคลึงกับตัวแบบ Structure ในภาษาระดับสูงอื่นๆ โดยมากมักจะทำการเก็บชุดข้อมูลที่มีความสัมพันธ์ซึ่งจะมีการเก็บข้อมูลในรูปของ key-value และไม่จำเป็นต้องสร้างขึ้นมาจากการเก็บข้อมูลของรถยังต้องโดยมีรายละเอียด

คุณสมบัติของรถตามตารางด้านล่าง



คุณสมบัติ
ยี่ห้อ : Fiat
รุ่น : s500
น้ำหนัก : 850kg
สี : ขาว

ตัวอย่างการใช้งานชุดคำสั่ง Object

```
// การประกาศใช้งาน  
var car1 = {type:"Fiat", model:"s500", color:"white", weight:850};  
  
// การเรียกใช้  
console.log(car1.model) // s500  
  
console.log(type of car1) //object  
  
console.log(type of car1.weight) //number  
  
// การประกาศใช้งาน  
var car2 = new Object();  
car2.type = "Fiat";  
car2.model = "s500";  
car2.color = "white";  
car2.weight = 850;  
  
console.log(type of car2) //object  
  
console.log(type of car2.weight) //number
```

3. ตัวแปรแบบพิเศษ (Special Type)

ตัวแปรแบบพิเศษนี้เป็นตัวแปรที่ถูกเพิ่มขึ้นมาเพื่อใช้ระบุว่าตัวแปรนั้นไม่ได้มีการกำหนดค่าไว้ โดยสามารถแบ่งได้ 2 ประเภท

ชนิดข้อมูล	การเก็บข้อมูล
null	ไม่มีค่าข้อมูลใดๆซึ่งค่า null ใช้สำหรับการยกเลิกพื้นที่เก็บค่าของตัวแปรออกจากหน่วยความจำ
undefined	ยังไม่ได้ระบุชนิดข้อมูล

3.1. Null คือ type ที่เราจะเป็นคนกำหนดเองเมื่อไม่มีค่าข้อมูลใดๆ

3.2. Undefined คือ type ที่ตัวภาษาที่กำหนดค่าเริ่มต้นมาให้ในกรณีที่ไม่มีการกำหนดค่าให้ซึ่งยังไม่ได้ระบุชนิดข้อมูล
ตัวอย่างการใช้งานตัวแปรแบบพิเศษ

```
var x;
console.log(typeof x); // undefined
var y = null;
console.log(typeof y); // null
console.log(undefined == null); // true
console.log(undefined === null); // false
console.log(typeof z === 'undefined');
console.log(z); // ERROR! ReferenceError: z is not defined
```

Digital Academy Thailand

การประกาศตัวแปร

การประกาศตัวแปร (Declarations) ภาษา JavaScript นี้จะต้องใช้คีย์เวิร์ด var, let, const หรือไม่ระบุคีย์เวิร์ดก็ได้ซึ่งการประกาศซึ่งแต่เดิมมีเฉพาะ var เพียงอย่างเดียวแต่ในเวอร์ชันปัจจุบันได้มีการเพิ่มคีย์เวิร์ดอื่นเพื่อเป็นการระบุสโคปของตัวแปร

1. var จะเป็นตัวแปรในลักษณะ function scope คือเมื่อตัวแปรถูกสร้างภายใน function เราจะสามารถเข้าถึงตัวแปรนั้นได้แต่จะไม่สามารถเข้าถึงเมื่อถูกเรียกวายนอก function

ตัวอย่างการใช้งาน var

```
function test() {  
    var temp = "OK";  
    console.log(temp); // ผลลัพธ์ OK  
}  
console.log(temp); // is not defined
```

2. let จะเป็นตัวแปรในลักษณะ Block scope คือจะมีลักษณะคล้ายกับ var แต่จะไม่สามารถเข้าถึงก่อนที่มันจะถูก assign ค่าได้และไม่สามารถประกาศตัวแปรซ้ำใน scope เดียวกันได้

ตัวอย่างการใช้งาน let

```
function test() {  
    let temp = "OK";  
    if(true){  
        let temp = "Not OK"  
        console.log(temp); // "Not OK"  
    }  
    console.log(temp); // OK  
}
```

3. const จะเป็นตัวแปรในลักษณะ Block scope เช่นเดียวกับ let แต่จะไม่สามารถเข้าถึงได้ก่อนถูก assign ค่าและ

ไม่สามารถ assign ซ้ำได้และไม่สามารถประกาศซ้ำได้ใน scope เดียวกัน

ตัวอย่างการใช้งาน const

```
const person = 'Boss';  
person = 'gmr'; // ข้อผิด Error  
console.log(person);
```

กฎในการประกาศตัวแปร

ในการประกาศสร้างตัวแปรต้องมีการกำหนดชื่อ ซึ่งชื่อนั้นไม่ใช่ว่าจะตั้งให้สื่อความหมายถึงข้อมูลที่เก็บอย่างเดียว โดยไม่คำนึงถึงอย่างอื่น เนื่องจากภาษา Javascript มีข้อกำหนดในการตั้งชื่อตัวแปรเอาไว้ แล้วถ้าตั้งชื่อผิดหลักการเหล่านี้ โปรแกรมจะไม่สามารถทำงานได้ หลักการตั้งชื่อตัวแปรในภาษา JavaScript แสดงไว้ดังนี้

- ชื่อตัวแปรสามารถประกอบด้วยตัวอักษร, ตัวเลข, เครื่องหมาย “_” และ “\$”
- ชื่อตัวแปรต้องขึ้นต้นด้วยตัวอักษร
- ชื่อตัวแปรสามารถขึ้นต้นด้วย “_” หรือ “\$” ได้
- อักษรตัวพิมพ์เล็ก (a) และ ตัวพิมพ์ใหญ่ (A) มีความหมายต่างกันเนื่องจาก JavaScript เป็นภาษาที่มีลักษณะเป็น Case Sensitive
- ห้ามใช้คำส่วน (Reserved Word)

```
abstract, arguments, await, boolean, break, byte, case, catch, char, class, const, continue,  
debugger, default, delete , do, double, else, enum, eval, export, extends, false, final,  
finally, float, for, function, goto, if, implements, import, in, instanceof, int, interface, let, long,  
native, new, null, package, private, protected, public, return, short, static, super, switch,  
synchronized, this, throw, throws, transient, true, try, typeof, var, void, volatile, while, with,  
yield
```

ตัวดำเนินการ(Operator)

ตัวดำเนินการ (Operator) หมายถึง เครื่องหมายกำหนดกรอบวิธีทางคณิตศาสตร์, พีชคณิต, บัญลี, การเปรียบเทียบ ระหว่างข้อมูล 2 ตัว ซึ่งเรียกว่า โอปරเรนด์(Operand) โดยอาจมีค่าเป็นตัวเลข ข้อความ ค่าคงที่ หรือตัวแปรต่าง ๆ ซึ่งชนิดของ Operator นั้นสามารถแบ่งได้ดังนี้

- ตัวดำเนินการคณิตศาสตร์ (Arithmetic operator) หมายถึง ใช้สำหรับคำนวณ Operand ที่เป็นค่าคงที่หรือตัวแปรก็ได้โดยให้ค่าผลลัพธ์เป็นตัวเลขค่าเดียว Operator เชิงคณิตศาสตร์ที่คนส่วนใหญ่รู้จักคุ้นเคยกันมากที่สุดได้แก่

ตัวดำเนินการ	ความหมาย
+	เครื่องหมายการบวก
-	เครื่องหมายการลบ
*	เครื่องหมายการคูณ
/	เครื่องหมายการหาร
%	เครื่องหมายหาเศษที่ได้จากการหารที่เรียกว่า Modulus
++	เครื่องหมายการเพิ่มค่าที่เรียกว่า Increment โดยจะเพิ่มค่าครั้งละ 1

--	เครื่องหมายการลดค่าที่เรียกว่า Decrement โดยจะลดค่าครั้งละ 1
(-)	เครื่องหมายแปลงค่าให้กลายเป็นค่าตรงกันข้ามกับค่าเดิมที่เรียกว่า Unary Negation

2. ตัวดำเนินการเชิงเปรียบเทียบ (Comparison operator) หมายถึง เครื่องหมายในการเปรียบเทียบข้อมูลผลลัพธ์ที่ได้จะมีค่าตຽรุกบูลีนเป็น จริง (True) และ เท็จ (False) ได้แก่

ตัวดำเนินการ	ความหมาย
==	เครื่องหมายเท่ากับ
!=	เครื่องหมายไม่เท่ากับ
>	เครื่องหมายมากกว่า
>=	เครื่องหมายมากกว่าหรือเท่ากับ
<	เครื่องหมายน้อยกว่า
<=	เครื่องหมายน้อยกว่าหรือเท่ากับ
====	เครื่องหมายเท่ากับทั้งจำนวนและชนิดข้อมูล

3. ตัวดำเนินการกำหนดค่า (Assignment operator) หมายถึง เครื่องหมายในการกำหนดให้ตัวแปรที่อยู่ทางฝั่งซ้าย มีค่าเท่ากับค่าเดิมในตัวแปรนั้น "กระทำ" (บวก, ลบ, คูณ, หาร) กับอีกด้วยตัวแปรหนึ่งที่อยู่ทางฝั่งขวา ได้แก่

ตัวดำเนินการ	ความหมาย
x=y	กำหนดค่า y ให้กับตัวแปร x
x+=y	เพิ่มค่า y ให้กับตัวแปร x ($x = x + y$)
x-=y	ลบค่า y ออกจากตัวแปร x ($x = x - y$)
x*=y	กำหนดค่า x คูณกับค่า y ให้กับตัวแปร x ($x = x * y$)
x/=y	กำหนดค่า x หารกับค่า y ให้กับตัวแปร x ($x = x / y$)
x% =y	กำหนดเศษที่ได้จากการหารค่า x ด้วยค่า y ให้กับตัวแปร x ($x = x \% y$)
x&=y	เก็บค่า x AND y ในตัวแปร x ($x = x \& y$)
x^=y	เก็บค่า x XOR y ในตัวแปร x ($x = x ^ y$)
x =y	เก็บค่า x OR y ในตัวแปร x ($x = x y$)

4. ตัวดำเนินการเชิงตรรก (Logical operator) เป็นเครื่องหมายที่ให้ค่าจริง (True) และ เท็จ (False) ในการเปรียบเทียบ ประกอบด้วยเครื่องหมาย

ตัวดำเนินการ	ความหมาย
&&	และ(AND) จะเป็นจริงเมื่อค่าที่ใช้เปรียบเทียบทั้ง 2 ค่าเป็นจริงทั้งคู่
	หรือ(OR) จะเป็นจริงเมื่อค่าที่ใช้เปรียบเทียบทั้ง 2 ค่าเป็นจริงทั้งคู่หรือจริงเพียงค่าใด ค่าหนึ่ง
!	ปฏิเสธ(NOT) เป็นการแปลงค่าตรงกันข้าม จากจริงจะเป็นเท็จ และ จากเท็จจะเป็นจริง

5. ตัวดำเนินการเชิงข้อความ (String operator) เป็นการเข้ามายังข้อความเข้าด้วยกัน (concatenation) โดยใช้เครื่องหมายบวก (+) เป็นตัวกระทำ

การเลือก(Selection)

การเขียนแอพพลิเคชั่นส่วนใหญ่มักจำเป็นต้องตัดสินใจบางอย่างและขึ้นอยู่กับเงื่อนไขบางอย่าง ซึ่งเหตุการณ์ในลักษณะนี้เราจะเรียกว่า การเลือก(Selection) ซึ่งการใช้งาน Selection ในภาษา JavaScript มีคำสั่งที่สำคัญอยู่ 2 แบบคือ if-else statement และ switch-case statement

1. **If-else statement** คำสั่ง if-else เป็นคำสั่งที่ใช้ในการสร้างประโยคเงื่อนไข โดยเป็นประโยคคำสั่งเลือกทำแบบสองทาง การทำงานจะตรวจสอบเงื่อนไขว่าเป็นจริงโดยมีรูปแบบดังนี้

การใช้งาน if statement

```
if(condition==ture) ถ้า (เงื่อนไขเป็นจริง) ..  
{  
    //statement  
}
```

ตัวอย่างการใช้งาน if-statement

```
var d = new Date(); //เอาวันที่ปัจจุบันขึ้นมา  
var time = d.getHours(); //เอาแต่เลขชั่วโมงมาใช้  
if(time < 10) // ถ้าเลขชั่วโมงที่ได้น้อยกว่า 10 จริง  
{  
    Console.log("Good morning"); //แสดงผลคำนี้  
}
```

การใช้งาน if-else statement

```
if(condition==ture) ถ้า (เงื่อนไขเป็นจริง) ..  
{  
    //statement  
}  
else ไม่ถ้า  
{  
    //statement  
}
```

ตัวอย่างการใช้งาน if-else statement

```
var d = new Date(); // เอาวันที่ปัจจุบันขึ้นมา
var time = d.getHours(); // เอาแต่เลขชั่วโมงมาใช้
if(time < 12) // ถ้าเลขชั่วโมงที่ได้น้อยกว่า 12 จริง
{
    Console.log("Good morning"); // แสดงผลคำว่า
}
else // ถ้าไม่จริง
{
    Console.log("Good afternoon"); // แสดงผลคำว่า
}
```

การใช้งาน if-else if statement

```
if(condition1==true) ถ้าเงื่อนไขที่ 1 เป็นจริง
{
    //statement
}
else if(condition2==ture) ถ้าเงื่อนไขที่ 2 เป็นจริง
{
    //statement
}
else ไม่ถูก (ไม่จริง)
{
    //statement
}
```

ตัวอย่างการใช้งาน if-else if statement

```
var d = new Date(); // เอาวันที่ปัจจุบันขึ้นมา
var time = d.getHours(); // เอาแต่เลขชั่วโมงมาใช้
if(time < 12) // ถ้าเลขชั่วโมงที่ได้น้อยกว่า 12 จริง
{
    console.log("Good morning"); // แสดงผลคำว่า
}
else if(time<18) // ถ้าเลขชั่วโมงที่ได้น้อยกว่า 19 จริง
{
    console.log("Good afternoon"); // แสดงผลคำว่า
}
else // ถ้าไม่จริง
{
    console.log("Good evening"); // แสดงผลคำว่า
}
```

2. **Switch-case statement** งานที่มีทางเลือกหลายๆทางบางครั้งทางเลือกจะไม่ใช่แค่จริงหรือเท็จ แต่เป็นตัวเลือก (choice) ให้เลือก เช่นการสร้างเมนูให้ผู้ใช้เลือก งานที่มีทางเลือกหลายๆ ตัวเลือกสามารถใช้คำสั่ง switch case ได้ การใช้งาน switch-case

```
n เป็นตัวแปรที่เก็บค่าเลขจำนวนเต็ม  
switch (n)  
{  
case 1: // สำหรับ n เป็นหนึ่ง  
    // statement ชุดที่ 1  
break; // จบการทำงาน  
case 2: // สำหรับ n เป็นสอง  
    // statement ชุดที่ 2  
break; // จบการทำงาน  
default: // สำหรับ n ไม่ตรงกับที่กำหนดไว้  
    // code อีกชุดหนึ่ง  
}
```

ตัวอย่างการใช้งาน switch-case

```
var d=new Date(); // เรียกวันที่ปัจจุบัน  
theDay=d.getDay(); // เล่าผลวันอุ่นมา (วันอาทิตย์เป็น 0 วันที่ร์เป็น 1 ไปเรื่อยๆ ถึง 6 )  
switch (theDay) // เลือกชุดคำสั่งโดยใช้ค่าของ theDay  
{  
case 5: // สำหรับ 5 (วันศุกร์)  
    console.log("Finally Friday");  
break;  
case 6: // สำหรับ 6 (วันเสาร์)  
    console.log("Super Saturday");  
break;  
case 0: // สำหรับ 0 (วันอาทิตย์)  
    console.log("Sleepy Sunday");  
break;  
default: // สำหรับที่ไม่ใช่ศุกร์ เสาร์ อาทิตย์  
    console.log("I'm looking forward to this weekend!");  
}
```

การทำซ้ำ(Iterative)

การเขียนโปรแกรมแบบทำซ้ำหรือวนรอบในที่นี้เราจะเรียกว่า loop ความหมายก็คือ การวนซ้ำหรือทำซ้ำไปเรื่อยๆ จนกว่าเงื่อนไขจะไม่เป็นจริง และประโยชน์ของการ loop ก็คือ ทำให้ผู้เขียนโปรแกรมเขียนโปรแกรมได้ง่าย กระชับ และยืดหยุ่นมากขึ้น โปรแกรมจะทำซ้ำไปเรื่อยๆ จนกว่าเงื่อนไขจะไม่เป็นจริง นั่นหมายความว่าผู้เขียนโปรแกรมไม่ต้องเขียนคำสั่ง เองหลายบรรทัด ตัวอย่างเช่น หากเราต้องการแสดงผลข้อความว่า "สวัสดี" ทั้งหมด 100 คำ นั่นหมายความว่าเราจะต้องเขียนคำสั่งแสดงผลทั้งหมด 100 คำสั่ง ซึ่งไม่คุ้มเสียเวลาเลย แต่ถ้าหากว่าเราเขียนวน loop เราอาจเขียนคำสั่งเพียงแค่ 3-4 บรรทัด เราก็จะได้ผลลัพธ์แสดงข้อความ "Hello" ทั้งหมด 100 คำ โดยที่ผู้เขียนไม่เสียเวลามากจนเกินไป และยังมีความยืดหยุ่นมากกว่าด้วยโดยการใช้งาน Loop มีการใช้งานที่หลากหลายดังนี้

- การวนรอบด้วยคำสั่ง do while เป็นการสั่งงานวนรอบแบบ test at bottom คือให้ทำงานใน loop ก่อนจนเสร็จ 1 รอบ แล้วจึงตรวจสอบเงื่อนไข หากเงื่อนไขเป็นจริงจึงจะวนกลับไปทำงานใน loop ใหม่ ดังนั้น loop ประเภทนี้จะต้องมีการทำงานอย่างน้อย 1 ครั้ง

การใช้งาน loop do while

```
do {  
    statement;  
} while (เงื่อนไข)
```

ตัวอย่างการใช้งาน loop do-while

```
// ทำการปริ้นตัวเลขตั้งแต่ 0-5  
var count=0; // ประกาศตัวแปรชื่อ count ให้มีค่าเป็น 0 เพื่อเป็นค่าเริ่มต้น  
do {  
    console.log(count);  
    count++;  
} while (count<5)
```

- การวนรอบด้วยคำสั่ง while เป็นการสั่งงานวนรอบแบบ test at top คือ ตรวจสอบเงื่อนไขก่อน หากเงื่อนไขเป็นจริง จึงจะทำงานใน loop โดย while จะทำงานโดย ทดสอบเงื่อนไขในวงเล็บ(condition) ก่อน ถ้าพบว่าเงื่อนไขเป็นจริงจะเข้าไปทำงานใต้ loop ตรงส่วน statement กรณีใน loop มี statement เดียวจะมีวงเล็บครอบหรือไม่ก็ได้ แต่กรณีที่ภายในloop มี statement หลาย statement จะต้องใส่เครื่องหมายวงเล็บครอบให้เป็น compound statement

การใช้งาน loop while

```
while (เงื่อนไข) {  
    statement;  
}
```

ตัวอย่างการใช้งาน loop while

```
// ทำการบริ่นด้วยตั้งแต่ 0-5
var count=0; // ประการตัวแปรชื่อ count ให้มีค่าเป็น 0 เพื่อเป็นค่าเริ่มต้น
while(count<6) {
    console.log(count);
    count++; // เพิ่มค่าของ count อีก 1
}
```

3. การวนรอบด้วยคำสั่ง for loop ที่พบเห็นได้บ่อยที่สุดคือ loop for มีความแตกต่างจาก while และ do while คือในส่วนของ for ภายในวงเล็บจะมีทั้งส่วนของการให้ค่าเริ่มต้น การเช็คเงื่อนไข และการเพิ่มจำนวนรอบให้กับ loop อุ่งภายในวงเล็บเดียว

การใช้งาน loop for

```
for (ค่าตั้งต้น;เงื่อนไข;นับ) {
    statement;
}
```

ตัวอย่างการใช้งาน loop for

```
// ทำการบริ่นด้วยตั้งแต่ 0-5
for(count=0;count<=5;count++) {
    console.log(count);
}
```

Digital Academy Thailand

ฟังก์ชัน(Function)

ฟังก์ชัน (Function) โดยเนื้อแท้แล้วก็คือโปรแกรมย่อย เป็นส่วนของโปรแกรมที่ถูกกำหนดให้ทำงานใดงานหนึ่ง จนสำเร็จ มีประโยชน์ตรงที่ช่วยเหลือการทำงาน หรือตอบสนองการทำงานต่อโปรแกรมหลัก เมื่อมีการเรียกใช้งาน และมีประโยชน์ทำให้โครงสร้างของโปรแกรม มีขนาดเล็กลง เมื่อมีการเรียกใช้งานเดิมซ้ำๆ หลายครั้ง แทนที่จะเขียนคำสั่งเดิมเพิ่มขึ้นใหม่ซ้ำๆ หลายครั้ง ทำให้ดูสิ้นเปลืองเนื้อที่ ข้ามตอนและเสียเวลาการทำงาน นอกจากนี้ฟังก์ชันยังทำให้โปรแกรมอ่านได้ง่ายขึ้น สะดวกในการหาจุดที่ผิดและง่ายต่อการปรับปรุงแก้ไขพัฒนาโปรแกรมให้ดียิ่งขึ้น โดยเราจะต้องทำการสร้างฟังก์ชันสร้างขึ้นเอง (User-defined Function) เป็นแบบชื่อของฟังก์ชันที่ผู้ใช้สร้างขึ้นมาใช้งาน เพื่อกำหนดให้ทำงานใดงานหนึ่งจนสำเร็จ เราอาจจะเรียกฟังก์ชันสร้างขึ้นเองนี้สั้นๆ ว่าฟังก์ชัน (Function) ซึ่งการเขียนด้วย React Native นี้เราจะใช้ Function ในรูปแบบ Arrow function และจะไม่กล่าวถึง function เนื่องจาก function แบบปกติในบางครั้งอาจจะใช้งานไม่ได้ทันทีอาจจะต้องมีการ Bind ฟังก์ชันในการใช้งานบางอย่างซึ่ง Arrow function จะไม่พบปัญหานี้

- การใช้งาน Arrow function แบบทั่วไป เป็นการใช้งานแบบไม่มีการ Pass parameter และ ไม่มีการ Return ค่ากับ การใช้งาน Arrow function แบบทั่วไป

```
functionName = () => {
    // statement
}
```

ตัวอย่างการใช้งาน Arrow function แบบทั่วไป

```
// การใช้งานในฟังก์ก
showText= () =>{
    console.log ("Pressed Button");
}
// In render fucntion
<Button title="OK" onPress={this.showText} />
```

- การใช้งาน Arrow function แบบ Pass parameter เป็นการใช้งานแบบมีการ Pass parameter ไปยัง function การใช้งาน Arrow function แบบ pass parameter

```
functionName = (parameter1,parameter2,...) => {
    // statement
}
```

ตัวอย่างการใช้งาน Arrow function แบบ pass parameter

```
// การใช้งานใน TextInput
showText=(txt)=>{
    console.log(txt);
}
// In render fucntion
<TextInput onChangeText={this.showText} />
```

3. การใช้งาน Arrow function แบบ Return เป็นการใช้งานแบบมีหรือไม่มีการ Pass parameter และ ไม่มีการ Return ค่ากับ

การใช้งาน Arrow function แบบ return parameter

```
functionName = (parameter1, parameter2, ...) => {  
    // statement  
    return value;  
}
```

ตัวอย่างการใช้งาน Arrow function แบบ pass parameter

```
multiply=(num1, num2)=>{  
    return num1*num2;  
}
```



การเรียกใช้งาน Function ระหว่าง Component

การใช้งาน function ส่วนใหญ่มักจะมีการใช้งานใน Component เดี่ยวกันหรืออยู่ใน Class เดี่ยวกันซึ่งการใช้งานจะเหมือนดังเช่นการใช้งาน function ก่อนหน้านี้แต่ในบางครั้งเราอาจจะมีการใช้งาน function ในลักษณะข้าม component ได้ 2 ลักษณะคือ children เรียกใช้งาน function ใน parent และ parent เรียกใช้งาน function ใน children

ผลลัพธ์การใช้งาน function แบบ children-parent และ parent-children



- การใช้งาน function แบบ children เรียกใช้งาน function ใน parent คือ การที่เราสร้าง function ไว้ใน parent และถูกเรียกใช้งานจาก children ซึ่งเราจะใช้ props ช่วยในการสร้างส่วนเรียกใช้งานใน children เพื่อให้ parent สามารถเรียกใช้งานได้

ตัวอย่างการใช้งาน function แบบ children-parent

ตัวอย่างเราสร้าง TextInput อยู่ใน Child component และ Text อยู่ใน App component(Parent) เมื่อทำการพิมพ์ข้อความจะแสดงข้อความที่พิมพ์ใน Text ด้วยโดย Child component จะทำการเรียกฟังก์ชันที่อยู่ใน Parent ผ่าน Props

```
1. import React, { Component } from 'react';
2. import {
3.   Text,
4.   View,
5.   TextInput,
6. } from 'react-native';
7.
8. class Child extends Component {
9.   render(props) {
10.     return (
11.       <TextInput style={{borderWidth:1, fontSize:16}}
12.         onChangeText={text=>{this.props.showText(text)}}
13.       );
14.   }
15. }
```

```

14. }
15.
16. export default class App extends Component {
17.   constructor(props) {
18.     super(props);
19.     this.state={
20.       text:"non"
21.     }
22.   }
23.
24.   showText=(txt)=>{
25.     this.setState({text:txt});
26.   }
27.
28.   render(props) {
29.     return (
30.       <View>
31.         <Child showText={this.showText}/>
32.           <Text>{this.state.text}</Text>
33.         </View>
34.       );
35.     }
36.   }

```

2. การใช้งาน function แบบ parent เรียกใช้งาน function ใน children คือ การที่เราสร้าง function ไว้ใน children แล้วกูเรียกใช้งานจาก parent ซึ่งเราจะใช้ ref ซึ่งเป็นฟังก์ชันเรียกกลับ Call ของ React ช่วยในการสร้าง ส่วนเรียกใช้งานใน Parent เพื่อเชื่อมต่อกับ Children ให้สามารถเรียกใช้งานได้

ตัวอย่างการใช้งาน function แบบ parent-children

ตัวอย่างเราสร้าง TextInput อยู่ใน App component(Parent) และ Text อยู่ใน Child component เมื่อทำการพิมพ์ ข้อความจะแสดงข้อความที่พิมพ์ใน Text ด้วยโดย App component จะทำการเรียกฟังก์ชันที่อยู่ใน Child ผ่าน Ref หรือ Callback function

```

1. import React, { Component } from 'react';
2. import {
3.   Text,
4.   View,
5.   TextInput,
6. } from 'react-native';
7.
8. class Child extends Component {
9.   constructor(props) {
10.     super(props);
11.     this.state={
12.       text:"non"
13.     }
14.     this.showText = this.showText.bind(this);
15.   }
16.
17.   showText=(txt)=>{
18.     this.setState({text:txt});
19.   }
20.

```

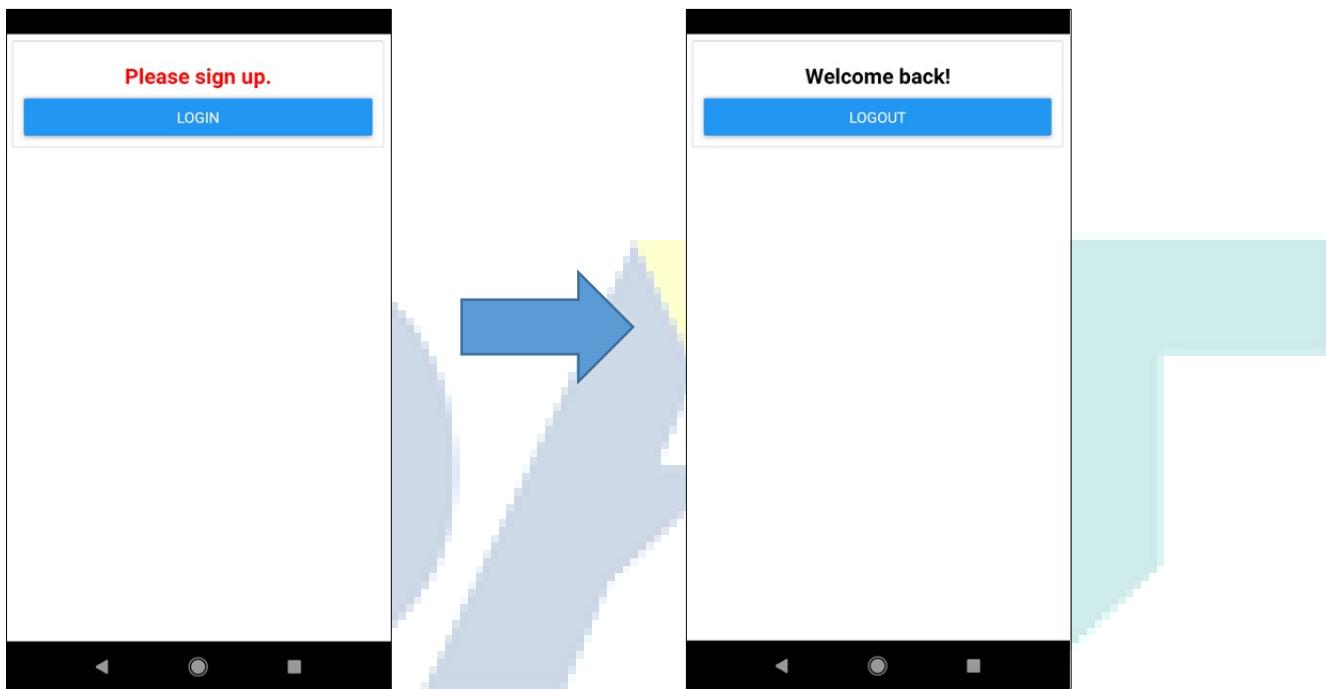
```
21.   render(props) {
22.     return (
23.       <Text>{this.state.text}</Text>
24.     );
25.   }
26. }
27.
28. export default class App extends Component {
29.   constructor(props) {
30.     super(props);
31.     this.myRef = React.createRef();
32.   }
33.
34.   textIn=(txt)=>{
35.     this.myRef.current.showText(txt);
36.   }
37.
38.   render(props) {
39.     return (
40.       <View>
41.         <TextInput style={{borderWidth:1,fontSize:16}}
42.           onChangeText={this.textIn}/>
43.         <Child ref={this.myRef}/>
44.       );
45.     }
46. }
```



Digital Academy Thailand

ตัวอย่างการใช้งาน Selection เพื่อควบคุม UI

ตัวอย่างโปรแกรมโดยมีการใช้งานคำสั่ง If-else เพื่อควบคุม UI ซึ่งจะมีโดยจะมีลักษณะที่แตกต่างจากการใช้งานในฟังก์ชันทั่วไปกล่าวคือเราจะใช้ If-else ใน render function ซึ่งจากตัวอย่างเราทำการ Check status state ว่ามีค่าเป็น 1 หรือ 0 หากมีค่าเป็น จะทำการกำหนดค่า UI text และ button ตามลักษณะด้านล่างซ้ายและ หากเป็น 1 จะทำการกำหนดค่า UI text และ button ตามลักษณะด้านล่างขวา ซึ่งในการเปลี่ยnlักษณะของ UI นี้ จะทำการควบคุมผ่าน Event onPress แล้วทำการ เปลี่ยน state status



Digital Academy Thailand

```
1. import React, { Component } from 'react';
2. import PropTypes from 'prop-types';
3. import {
4.   Text,
5.   View,
6.   StyleSheet,
7.   Button,
8. } from 'react-native';
9.
10. export default class Login extends Component {
11.   constructor() {
12.     super();
13.     this.state = {
14.       status: 0,
15.       txtStatus:"Login",
16.       items:[]
17.     };
18.     this.myRef = React.createRef();
19.   }
20. }
```

```
21.     render(props) {
22.       let text;
23.       let button;
24.
25.       if(this.state.status==0){
26.         text = <Text style={styles.textLogin}>Please sign up.</Text>
27.         button = <Button title="Login" onPress={()=>{this.setState({status:1})}}>
28.       } />
29.       else{
30.         text = <Text style={styles.textLogout}>Welcome back!</Text>
31.         button = <Button title="Logout" onPress={()=>{this.setState({status:0})}}>
32.       } } />
33.     }
34.     return (
35.       <View>
36.         <View style={styles.containerStyle}>
37.           {text}
38.           {button}
39.         </View>
40.       );
41.   }
42.
43.   const styles = StyleSheet.create({
44.     containerStyle: {
45.       borderWidth: 1,
46.       borderRadius: 2,
47.       borderColor: '#DDDDDD',
48.       margin: 6,
49.       padding: 10,
50.     },
51.     textLogin: {
52.       color:'red',
53.       fontWeight:"bold",
54.       fontSize:20,
55.       padding:8,
56.       alignSelf:"center"
57.     },
58.     textLogout: {
59.       fontWeight:"bold",
60.       fontSize:20,
61.       padding:8,
62.       alignSelf:"center"
63.     }
64.   });
65.
```

ตัวอย่างการเพิ่ม Object แบบ Static

ตัวอย่างโปรแกรมจะทำการเพิ่ม Object ที่มีจำนวนแน่นอนโดยจะใช้ Loop เพื่อควบคุม UI โดยทำการสร้าง Item Component ที่มี Props Color, ID, Name เพื่อแสดงสี ไอดี และ ชื่อ ในการเรียกใช้งาน Item Component ถูกเรียกใช้งานจาก App Component เพื่อทำการเพิ่ม Item Component ลงใน App Component ตามข้อมูลที่อยู่ภายใน initialArr(Array of Object) โดยใช้การวนลูป for ตามจำนวนข้อมูลที่อยู่ภายใน initialArr และสร้าง Object ของ Item Component ที่มีค่า Props ตาม Object เก็บไว้ใน arr ที่มีการแสดงผลใน Render function



ตัวอย่างโค้ด

```
1. // App.js
2. import React, { Component } from 'react';
3. import PropTypes from 'prop-types';
4. import {
5.   View,
6.   StyleSheet,
7. } from 'react-native';
8.
9. import Item from './Item'
10.
11. let initialArr = [{ id: 1, color: "blue", name: "text1"}, {id: 2,color: "red",name: "text2"}];
12.
13. export default class App extends Component {
14.   constructor() {
15.     super();
16.   }
17. }
```

```
17.
18.     render(props) {
19.       let arr = [];
20.       for(let i=0;i<initialArr.length;i++) {
21.         arr.push(<Item name={initialArr[i].name} id={initialArr[i].id} color={i
 initialArr[i].color} />)
22.       }
23.       return (
24.         <View style={styles.containerStyle}>
25.           {arr}
26.         </View>
27.       );
28.     }
29.   }
30. const styles = StyleSheet.create({
31.   containerStyle: {
32.     borderWidth: 1,
33.     borderRadius: 2,
34.     borderColor: '#DDDDDD',
35.     margin: 6,
36.     padding: 10,
37.   }
38. });


```

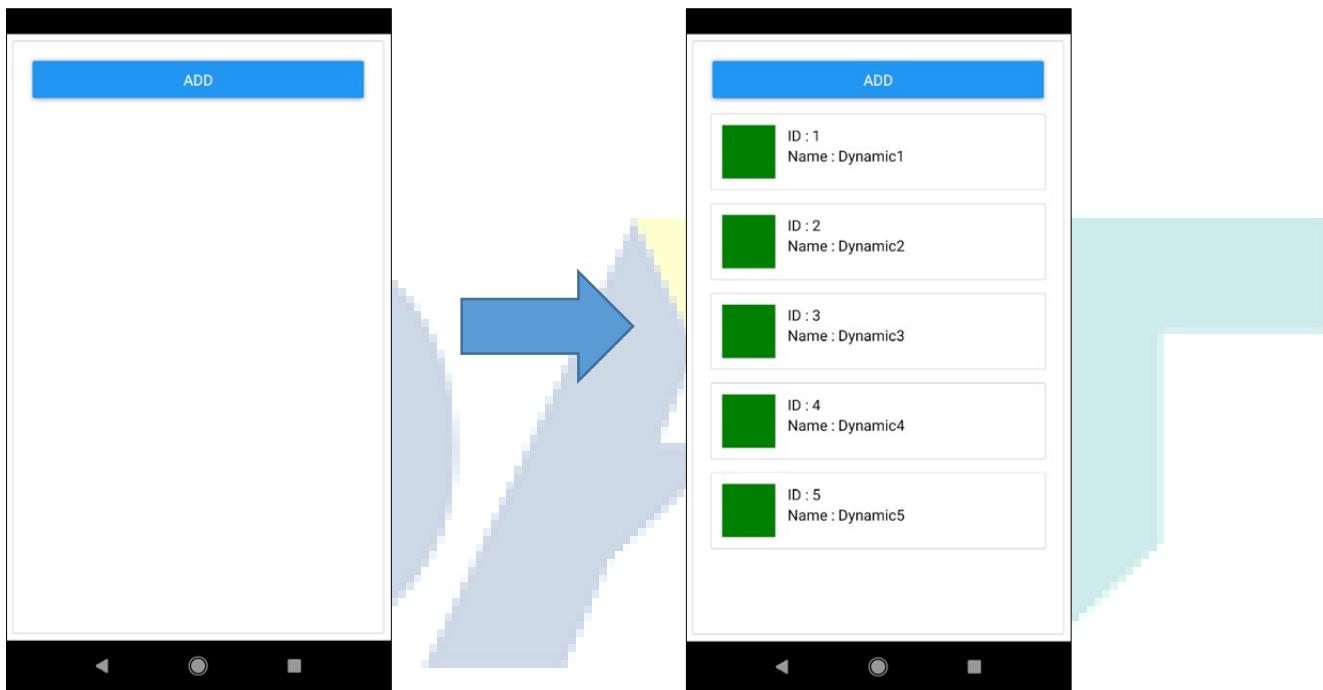


Digital Academy Thailand

```
1. // Item.js
2. import React, { Component } from 'react';
3. import PropTypes from 'prop-types';
4. import {
5.   Text,
6.   View,
7.   StyleSheet,
8.   Image,
9.   TextInput,
10.    TouchableOpacity,
11. } from 'react-native';
12.
13. export default class Item extends Component {
14.
15.   render(props) {
16.     return (
17.       <View style={styles.containerStyle}>
18.         <View style={{flexDirection:"row"} }>
19.           <View style={{width:50,height:50,backgroundColor:this.props.color}}/>
20.           <View style={{flexDirection:"column",paddingStart:8 }}>
21.             <Text> ID : {this.props.id} </Text>
22.             <Text> Name : {this.props.name} </Text>
23.           </View>
24.         </View>
25.       </View>
26.     );
27.   }
28. }
29.
30. Item.PropTypes = {
31.   id: PropTypes.string.isRequired,
32.   name: PropTypes.string.isRequired,
33.   color: PropTypes.string.isRequired,
34. };
35.
36. const styles = StyleSheet.create({
37.   containerStyle: {
38.     borderWidth: 1,
39.     borderRadius: 2,
40.     borderColor: '#DDDDDD',
41.     margin: 6,
42.     padding: 10,
43.   }
44. });
```

ตัวอย่างการเพิ่ม Object แบบ Dynamic

ตัวอย่างโปรแกรมจะทำการเพิ่ม Object ที่มีจำนวนไม่แน่นอนโดยโดยทำการสร้าง Item Component ที่มี Props Color, ID, Name เพื่อแสดงสี ไอดี และ ชื่อ ในการเรียกใช้งาน Item Component ถูกเรียกใช้งานจาก App Component เพื่อทำการเพิ่ม Item Component ลงใน App Component ตามจำนวนครั้งที่กดปุ่ม ADD โดยใช้การเพิ่ม Object of Item Component ค่าลงใน items array ที่ถูกสร้างแบบ state ที่มีการแสดงผลใน Render function



Digital Academy Thailand

```
1. //App.js
2. import React, { Component } from 'react';
3. import PropTypes from 'prop-types';
4. import {
5.   View,
6.   StyleSheet,
7.   Button,
8.   ScrollView,
9. } from 'react-native';
10.  import Item from './Item'
11.
12.  export default class App extends Component {
13.    constructor() {
14.      super();
15.      this.state = {
16.        items: []
17.      };
18.      this.myRef = React.createRef();
19.    }
20. }
```

```
21.     Add= ()=>{
22.         var len = (this.state.items.length+1).toString()
23.         let item = <Item name={"Dynamic"+len} id={len} color="green" />
24.         this.setState({items:this.state.items.concat(item)})
25.     }
26.
27.     render(props) {
28.         return (
29.             <View style={styles.containerStyle}>
30.
31.                 <View style={{padding:8}}>
32.                     <Button title="Add" onPress={this.Add} />
33.                 </View>
34.
35.                 <View style={{flex:1}}>
36.                     <ScrollView style={{flex:1}}>
37.                         {this.state.items}
38.                     </ScrollView>
39.                 </View>
40.
41.             </View>
42.         );
43.     }
44. }
45.
46. const styles = StyleSheet.create({
47.     containerStyle: {
48.         flex:1,
49.         borderWidth: 1,
50.         borderRadius: 2,
51.         borderColor: '#DDDDDD',
52.         margin: 6,
53.         padding: 10,
54.     }
55. });

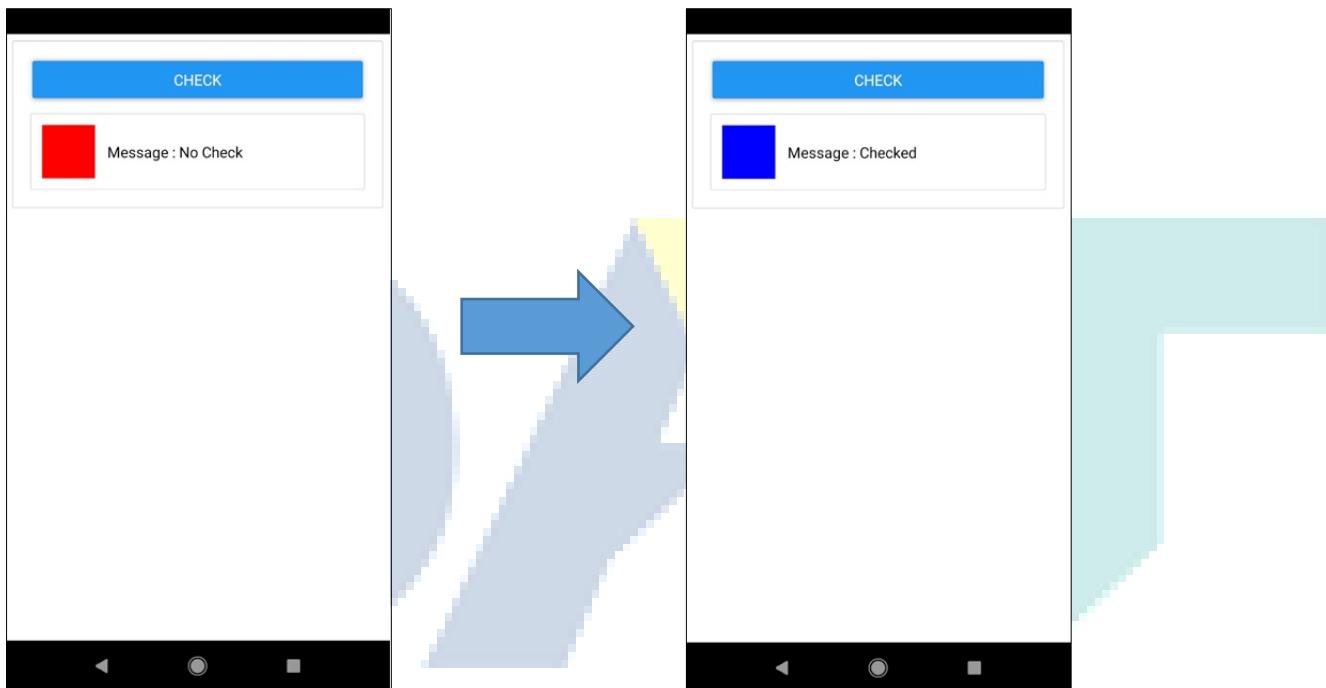
```

Digital Academy Thailand

```
1. //Item.js
2. import React, { Component } from 'react';
3. import PropTypes from 'prop-types';
4. import {
5.   Text,
6.   View,
7.   StyleSheet,
8. } from 'react-native';
9.
10.  export default class Item extends Component {
11.
12.    render(props) {
13.      return (
14.        <View style={styles.containerStyle}>
15.          <View style={{flexDirection:"row"} }>
16.            <View style={{width:50,height:50,backgroundColor:this.props.color }} />
17.            <View style={{flexDirection:"column",paddingStart:8 }}>
18.              <Text> ID : {this.props.id} </Text>
19.              <Text> Name : {this.props.name} </Text>
20.            </View>
21.          </View>
22.        </View>
23.      );
24.    }
25.  }
26.
27. Item.PropTypes = {
28.   id: PropTypes.string.isRequired,
29.   name: PropTypes.string.isRequired,
30.   color: PropTypes.string.isRequired,
31. };
32.
33. const styles = StyleSheet.create({
34.   containerStyle: {
35.     borderWidth: 1,
36.     borderRadius: 2,
37.     borderColor: '#DDDDDD',
38.     margin: 6,
39.     padding: 10,
40.   }
41.});
```

ตัวอย่างการเรียกฟังชั่น Parent-Child

ตัวอย่างโปรแกรมการสร้าง Function และการเรียกใช้งานแบบ Parent-Children คือจะทำการเรียกฟังก์ชันที่อยู่ใน Item Component ที่เป็น Children component ถูกเรียกจาก App Component ซึ่งเป็น Parent โดยแอพพลิเคชันจะทำงานโดยเมื่อกดปุ่ม CHECK ที่อยู่ใน App Component แล้วจึงเปลี่ยนสี View จากแดงเป็นสีเขียวและจากข้อความจาก No Check เป็น Checked ซึ่งใน Item Component



Digital Academy Thailand

```
1. //App.js
2. import React, { Component } from 'react';
3. import PropTypes from 'prop-types';
4. import {
5.   Text,
6.   View,
7.   StyleSheet,
8.   Button,
9. } from 'react-native';
10.
11. import Item from './Item'
12.
13. export default class App extends Component {
14.   constructor() {
15.     super();
16.     this.myRef = React.createRef();
17.   }
18.
19.   Check=()=>{
20.     this.myRef.current.check();
21.   }
22. }
```

```
23.     render(props) {
24.
25.       return (
26.         <View style={styles.containerStyle}>
27.           <View style={{ padding:8 }}>
28.             <Button title="Check" onPress={this.Check}/>
29.           </View>
30.           {<Item ref={this.myRef}/>}
31.         </View>
32.       );
33.     }
34.   }
35.
36. const styles = StyleSheet.create({
37.   containerStyle: {
38.     borderWidth: 1,
39.     borderRadius: 2,
40.     borderColor: '#DDDDDD',
41.     margin: 6,
42.     padding: 10,
43.   }
44. });
});
```



Digital Academy Thailand

```

1. //Item.js
2. import React, { Component } from 'react';
3. import PropTypes from 'prop-types';
4. import {
5.   Text,
6.   View,
7.   StyleSheet,
8.   TextInput,
9. } from 'react-native';
10.
11. export default class Item extends Component {
12.   constructor() {
13.     super();
14.     this.state = {
15.       color: "red",
16.       text: "No Check",
17.       isCheck: true
18.     }
19.     this.myRef = React.createRef();
20.     this.check = this.check.bind(this)
21.   }
22.
23.   check() {
24.     this.setState({isCheck: !this.state.isCheck});
25.     if(this.state.isCheck) {
26.       this.setState({color: "blue"});
27.       this.setState({text: "Checked"});
28.     } else{
29.       this.setState({color: "red"});
30.       this.setState({text: "No Check"});
31.     }
32.   }
33.
34.   render(props) {
35.     return (
36.       <View ref={this.myRef} style={styles.containerStyle}>
37.         <View style={{flexDirection: "row"} }>
38.           <View style={{width:50,height:50,backgroundColor: this.state.color}}/>
39.           <View style={{flexDirection: "column",paddingStart:8,justifyContent: "center"} }>
40.             <Text> Message : {this.state.text} </Text>
41.           </View>
42.         </View>
43.       </View>
44.     );
45.   }
46. }
47. const styles = StyleSheet.create({
48.   containerStyle: {
49.     borderWidth: 1,
50.     borderRadius: 2,
51.     borderColor: '#DDDDDD',
52.     margin: 6,
53.     padding: 10,
54.   }
55. });

```

บทที่ 5 การสร้างลิสข้อมูล

ลิสข้อมูล(List)

การพัฒนาแอปพลิเคชันบนสมาร์ตโฟนส่วนใหญ่มักจะมีการแสดงรายการข้อมูลไม่ว่าจะอยู่ในรูปแบบข้อความ, รูปภาพ, วิดีโอ หรือสื่อประสมต่างๆขึ้นอยู่กับงานที่ใช้ ในอดีตการสร้างรายการข้อมูลมักจะเป็นการแสดงในรูปแบบข้อความ หรือรูปภาพอย่างโดยย่างหนึ่งเท่านั้น เพราะเครื่องมือที่ใช้พัฒนาและสมาร์ตโฟนยังด้อยประสิทธิภาพแต่ในปัจจุบัน เทคโนโลยีต่างๆได้ถูกพัฒนาอย่างก้าวกระโดดทำให้สามารถแสดงข้อมูลสื่อประสมต่างๆได้อย่างราบลื่น ตัวอย่างแอปพลิเคชัน Facebook ที่เน้นการแสดงข้อมูลแบบลีส หากลองสักเกตุ Feature ของ Facebook จะทำการแสดงข้อมูลในรูปแบบลีสไม่ ว่าจะเป็นลีสของโพสต์ต่างๆ, Story, วิดีโอ หรือแม้กระทั้ง message โดยข้อดีของการแสดงผลด้วยลีสมีดังนี้ สามารถแสดง ข้อมูลจำนวนมากไม่มีจำกัด, ใช้งานง่าย, ประหยัดพื้นที่ในการแสดงผลและง่ายต่อการพัฒนา การสร้างลิสข้อมูลสิ่งแรกที่เรา จะต้องทำความเข้าใจคือประเภทของลีสโดยทั่วไปความสามารถแบ่งได้เป็น 2 ประเภทดังนี้

- ลีสที่แสดงข้อมูลอย่างเดียว คือลีสที่มีการแสดงข้อมูลเพียงอย่างเดียวโดยที่ผู้ใช้งานไม่สามารถมีปฏิสัมพันธ์กับลีส นั้นได้ ยกตัวอย่างเช่น ลีสแสดงประวัติการโกรเข้ารับสายในโทรศัพท์ ลีสแสดงรูปภาพในเครื่องเป็นต้น
- ลีสที่มีปฏิสัมพันธ์กับผู้ใช้ คือลีสที่ผู้ใช้งานสามารถกระทำการใดๆกับลีสนั้นได้ ยกตัวอย่างเช่น ลีสโพสต์ต่างๆในเพส Facebook ที่ผู้ใช้งานสามารถกดไลค์หรือคอมเม้นต์ในโพสนั้นได้เป็นต้น

ในการพัฒนาแอปพลิเคชันที่มีลีสอยู่ด้วย React-Native เราจำเป็นจะต้องรู้และเข้าใจถึงเครื่องมือที่จำเป็นในการสร้าง ลีสและเครื่องมือในการตกแต่งดังนี้

Digital Academy Thailand

องค์ประกอบของลีสข้อมูล

การสร้าง List ข้อมูลสามารถได้โดยใช้ Scroll View และ FlatList โดยในบทนี้จะกล่าวเฉพาะการสร้างด้วย FlatList เท่านั้น เพราะ Scroll View ได้ลองทำไปบ้างแล้ว ซึ่งการสร้าง List ไม่ว่าจะใช้ Component ใดล้วนแล้วจะต้องประกอบไปด้วย 3 ส่วนหลักดังนี้

1. ข้อมูล คือ ข้อมูลต่างๆที่ต้องการแสดงในลีสโดยข้อมูลเหล่านี้อาจได้มาจากการอ่านข้อมูลหรือเว็บ API ต่างๆ ตัวอย่างข้อมูลที่ใช้งานบ่อยอาทิเช่น

- 1.1. Object variable คือ ตัวแปรที่มีลักษณะการเก็บข้อมูลแบบ Array of object ซึ่ง object จะมีลักษณะคล้ายคลึงกับตัวแปรแบบ Structure ในภาษาต้นต้นๆ โดยมากมักจะทำการเก็บชุดข้อมูลที่มีความสัมพันธ์ซึ่งจะมีการเก็บข้อมูลในรูปของ key-value และจะถูกรวบไว้ใน Array เป็นชุดข้อมูล โดยตัวอย่างข้อมูลด้านล่าง

```
const data = [
  {
    id: 1,
    title: 'First Item',
  },
  {
    id: 2,
    title: 'Second Item',
  },
  {
    id: 3,
    title: 'Third Item',
  },
];
```

- 1.2. JSON ย่อมาจาก JavaScript Object Notation คือ Standard format อย่างหนึ่งที่เป็นแบบ Clear text สามารถอ่านออกได้โดยตาเปล่า ใช้ในการสร้าง object ขึ้นมาเพื่อส่งข้อมูลระหว่าง application ในรูปแบบ File หรือ Applications Program Interface (API) โดย format จะมีรูปแบบเป็นคู่ Key-Value หรือเป็นแบบ Array โดย Type ของ Value ใน JSON จะมีดังนี้

1. Number: ตัวเลขเท่านั้น
2. String: Unicode ใช้เครื่องหมาย double-quote ("") เป็นตัวบ่งบอก
3. Boolean: True or False
4. Array: ชุดข้อมูล ซึ่งจะเป็นชนิดได้ ใช้สัญลักษณ์ square bracket [var1,var2] เป็นตัวแสดง และคั้นด้วย comma แต่คล้ายใน array
5. Object: ชุดข้อมูลที่เป็นคู่ Key-Value แบบ strings ใช้สัญลักษณ์ปีกกา {key1:value1,key2:value2} ใช้ comma เป็นตัวแบ่งแต่ละคู่ และใช้ colon เป็นตัวแบ่งระหว่าง key และ value
6. Null: ค่าว่าง

โดยตัวอย่างการเก็บข้อมูลในรูปแบบของ JSON ตามตัวอย่างด้านล่างจะประกอบไปด้วย Key - car ทำการเก็บข้อมูลในรูปแบบ Array ที่มีสมาชิกเป็น Object ที่ประกอบไปด้วย Key – id, url, name ที่มี Type เป็น String และ Key – price มี Type เป็น Number

ตัวอย่างไฟล์ข้อมูล JSON

```
//Data.json
{
  "car" : [
    {"id" : 1, "url":"EvMoVn.png", "name":"Altis", "price":839000},
    {"id" : 2, "url":"EvmLES.jpg", "name":"Camry", "price":1455000},
    {"id" : 3, "url":"EvmAYg.jpg", "name":"CH-R", "price":979000},
    {"id" : 4, "url":"EvmPRW.jpg", "name":"Cross", "price":1019000},
    {"id" : 5, "url":"EvmSu2.jpg", "name":"Passo", "price":487000},
    {"id" : 6, "url":"Evm2M1.jpg", "name":"Vios", "price":609000}
  ]
}
```

2. Render Item คือ Custom Component ที่จะใช้ในการบรรจุข้อมูลที่มีการตกแต่งอย่างสวยงามสามารถแบ่งตามวิธีการสร้างได้ 2 ประเภท Method Component และ Class Component โดยการสร้างแต่ละแบบนั้นจะมีข้อดีข้อเสียที่ต่างกันดังนี้

ข้อดีของการใช้งาน Method Component

ในการสร้างแบบ Method Component เราจะสามารถจัดการ Event ที่เกิดขึ้นได้อย่างง่ายกว่าแบบ Class Component เมื่อจากอาจจะมีการเรียกใช้งาน State เดียวกัน เช่น การกำหนด SelectedID ที่เป็น State หากเราใช้ Method Component เราสามารถ set State ได้ทันทีแต่หากเราใช้ Class Component อาจจะต้องทำการการส่งค่าระหว่าง Component หรือทำ state management เพื่อส่งค่าไปยัง Parent

ข้อดีของการใช้งาน Class Component

ในการสร้างแบบ Class Component จะสร้างง่ายกว่าและสามารถแบ่งงานกันทำที่ง่ายในกรณีมีการทำงานเป็นทีม

- 2.1. Method Component คือ Component ที่ถูกสร้างในลักษณะของ Function ที่มีการ ส่งพารามิเตอร์ไปยัง Function(อ่านเพิ่มเติมบทที่ 3) โดยพารามิเตอร์ที่ส่งไปคือ Object ที่เราได้จากการอ่านไฟล์ JSON หรือ API ต่างๆ ทั้ง ตัวอย่างส่วนของโปรแกรมแบบ renderItem แบบ method แสดงด้านล่าง

```
renderItem={({item})=>{
  return(
    <View style={styles.item}>
      <Text style={styles.title}>{item.name}</Text>
    </View>
  );
}}
```

2.2. Class Component คือ Component ที่ถูกสร้างในลักษณะแบบ Class(อ่านเพิ่มเติมบทที่ 3) ซึ่งการสร้างจำเป็นจะต้องมีการส่งค่าพารามิเตอร์ไปยัง Class Component เราจำเป็นจะต้องสร้าง Props เพื่อรองรับค่าที่ถูกส่งมา ตัวอย่างส่วนของโปรแกรมแบบ renderItem แบบ class แสดงด้านล่าง

```
//App.js
renderItem={({item})=>{
  return (
    <Item name={item.name} />
  );
}}
```

```
//Item.js
render(props) {
  return (
    <View style={styles.item}>
      <Text style={styles.title}>{this.props.name}</Text>
    </View>
  );
}

Item.propTypes={
  name:PropTypes.string
}
```

3. List คือ ส่วนที่แสดงรายการข้อมูลสามารถปรับแนวการแสดงผลได้ทั้งแนวตั้งและแนวนอนในการแสดงรายงานเราสามารถใช้ Component ได้ 2 แบบคือ ScrollView และ ListView โดยแต่ละแบบจะมีจุดเด่นจุดด้อยดังนี้

การทำงาน	ListView	ScrollView
1. การแสดงผลแนวตั้งและแนวนอน	ได้	ได้
2. การเพิ่มข้อมูลใน列ส	Object, Array	ผ่าน Array
3. พัฒนสนับสนุนการทำงาน	มี	ไม่มี(เขียนเอง)
4. การควบคุม Event ที่เกิดขึ้น	ง่าย	ยาก

การสร้างลีสข้อมูลด้วย FlatList

ในการสร้าง List ด้วย FlatList เราจำเป็นจะต้องเข้าใจลักษณะ Component, Feature และ การใช้งาน Callback function ที่จำเป็น

การติดตั้ง

ในการใช้งาน FlatList จะต้องทำการ Import Component FlatList ที่อยู่ในไลบรารี 'react-native' เสียก่อนตามตัวอย่างด้านล่าง

```
import { FlatList } from 'react-native';
```

การเรียกใช้งาน

การใช้งาน FlatList โดยส่วนมากจะกำหนด Props ที่สำคัญอย่างน้อยสุด 3 ค่าคือ

Props	หน้าที่และการใช้งาน
renderItem	Props ที่ทำหน้าที่ Render Item ในแต่ละรายการที่แสดงโดยมีลักษณะการใช้งาน renderItem({ item, index, separators }); item (Object): คือ Object ของ Custom Component ที่ต้องการ Render index (number): คือ ตำแหน่งของ Index ที่อยู่ใน Array separators (Object) คือ Object ของ Custom Component ที่ใช้ในการแบ่งแต่ละ item
data	ข้อมูลที่ต้องการนำมาระดับใน List ที่มีลักษณะเป็น Array
keyExtractor	คีย์ที่มีลักษณะเป็นเอกภาพ(ไม่ซ้ำกัน) เพื่อใช้ในการอ้างอิงภายใต้ในกรณีที่มีการจัดเรียงแบบไม่เป็นลำดับ

การเรียกใช้งานสามารถทำได้ตามลักษณะด้านล่าง

```
<FlatList  
  Data={Your_Data}  
  renderItem={Your_RenderItem_Function}  
  keyExtractor={Your_Identifier_Key}  
 />
```

Props อื่นๆที่จะใช้หรือไม่ใช้ก็ได้โดย <https://reactnative.dev/docs/flatlist.html>

ตัวอย่างการใช้ FlatList ในรูปแบบต่างๆ

- การสร้าง List อย่างง่ายด้วย FlatList เพื่อแสดงข้อมูล โดยปกติแล้วเราสามารถสร้างได้ 2 แบบคือ Method และ Class component ซึ่งในตัวอย่างจะใช้วิธีการสร้างแบบ Method component โดยตัวอย่างจะทำการแสดง รายการรุ่นของรถต่างๆของรถยี่ห้อ Toyota ที่ทำการดึงข้อมูลจาก JSON ไฟล์ และนำมาแสดงผลโดยใช้ UI โดยใช้ Props พื้นฐานและยังไม่ได้มีการควบคุม หรือตกแต่ง Component ซึ่งตัวแอพพลิเคชันจะมีลักษณะดังภาพ



ตัวอย่างข้อมูล JSON

```
{
  "car" : [
    {"id" :1, "url":"https://www.img.live/images/2020/08/24/Altis.th.jpg", "name":"Altis", "price":839000, "description":"ประเภทเกียร์: Continuous variable transmission\nเก้าอี้นั่ง: 1.5 ลิตร\bgาน้ำสูงสุด: 100 แรงม้า"}, 
    {"id" :2, "url":"https://www.img.live/images/2020/08/24/camry.th.jpg", "name":"Camry", "price":1455000, "description":"ประเภทเกียร์: Continuous variable transmission\nเก้าอี้นั่ง: 1.8 ลิตร\bgาน้ำสูงสุด: 125 แรงม้า"}, 
    {"id" :3, "url":"https://www.img.live/images/2020/08/24/chr.th.jpg", "name":"CHR", "price":979000, "description":"ประเภทเกียร์: Continuous variable transmission, เก้าอี้นั่ง: 1.8 ลิตร, กำลังสูงสุด: 125 แรงม้า"}, 
    {"id" :4,"url":"https://www.img.live/images/2020/08/24/cross.th.jpg", "name":"Cross", "price":1019000, "description":"ประเภทเกียร์: Continuous variable transmission\nเก้าอี้นั่ง: 1.8 ลิตร\bgาน้ำสูงสุด: 120 แรงม้า"}, 
    {"id" :5, "url":"https://www.img.live/images/2020/08/24/passo.th.jpg", "name":"Passo", "price":487000, "description":"ประเภทเกียร์: Continuous variable transmission\nเก้าอี้นั่ง: 1.0 ลิตร\bgาน้ำสูงสุด: 68 แรงม้า"}, 
    {"id" :6, "url":"https://www.img.live/images/2020/08/24/vios.th.jpg", "name":"Vios", "price":609000, "description":"ประเภทเกียร์: Continuous variable transmission\nเก้าอี้นั่ง: 1.5 ลิตร\bgาน้ำสูงสุด: 108 แรงม้า"}]
```

ตัวอย่าง Code

```
1. import React, { Component } from 'react';
2. import {View,FlatList,TouchableOpacity,StyleSheet,Text} from 'react-native';
3. import * as data from './data.json'
4.
5. export default class App extends Component {
6.   constructor(){
7.     super();
8.     this.state = {
9.       };
10.   }
11.
12.   renderItem={({item})=>{
13.
14.     return(
15.       <View style={[styles.itemStyle]}>
16.         <Text style={styles.title}>{item.name}</Text>
17.       </View>
18.     );
19.   }
20.
21.   render(props) {
22.     return (
23.       <View style={{flex:1}}>
24.         <FlatList
25.           data={data.car}
26.           renderItem={this.renderItem}
27.           keyExtractor={item => item.id}
28.         />
29.       </View>
30.     );
31.   }
32. }
33. const styles = StyleSheet.create({
34.   itemStyle: {
35.     padding: 10,
36.     marginVertical: 8,
37.   },
38.   title: {
39.     fontSize: 24,
40.   });

```

2. การตกแต่ง List ด้วย Custom Component ในตัวอย่างนี้จะเมื่อกับตัวอย่างก่อนหน้าแต่จะมีการตกแต่งรายละเอียดต่างๆให้มากขึ้นโดยใช้ Core Components และ Pops ของ FlatList

```
<FlatList  
    data={data.car}  
    renderItem={this.renderItem}  
    keyExtractor={item => item.id}  
    ItemSeparatorComponent={this.renderSeparator}  
    ListHeaderComponent={this.renderHeader}  
/>
```

เราได้ทำการเรียกใช้ ItemSeparatorComponent เพื่อสร้างเส้นขีดแบ่งของแต่ละ Item ผ่าน Function Component ชื่อ renderSeparator

```
renderSeparator = () => {  
  return (  
    <View  
      style={{  
        height: 1,  
        backgroundColor: "#dddddd",  
      }}  
    />  
  );  
};
```

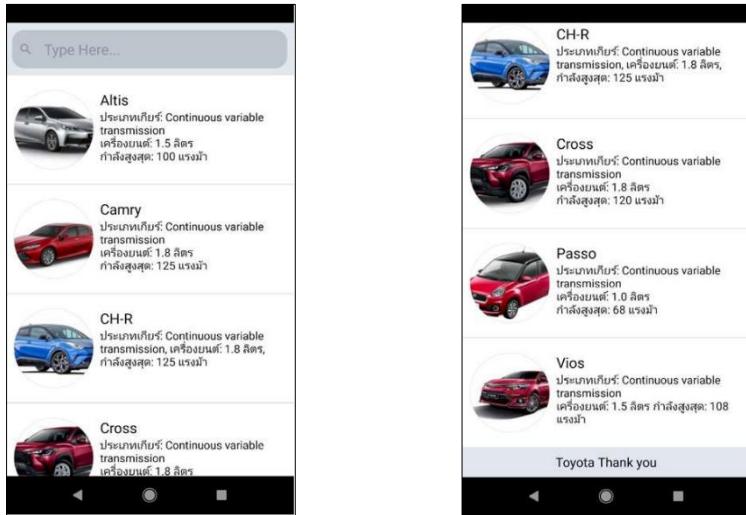
ในส่วนของ header เราได้ทำการ ListHeaderComponent เพื่อสร้างส่วนหัวของ List ผ่าน Function Component ชื่อ renderHeader

```
renderHeader = () => {  
  return <SearchBar placeholder="Type Here..." lightTheme round />;  
};
```

ในส่วนของ Footer เราได้ทำการ ListFooterComponent เพื่อสร้างส่วนท้ายของ List ผ่าน Function Component ชื่อ renderFooter

```
renderFooter = () => {  
  return (  
    <View style={styles.footer}>  
      <Text style={{textAlign:"center",fontSize:16}}> Toyota Thank you </Text>  
    </View>  
  );  
};
```

ตัวอย่างผลลัพธ์การทำงาน



ตัวอย่างโค้ดทั้งหมด

```
1. import React, { Component } from 'react';
2. import {
3.   View, FlatList, TouchableOpacity, StyleSheet, Text, Alert, Image
4. } from 'react-native';
5. import * as data from './data.json'
6. import { List, ListItem, SearchBar } from "react-native-elements";
7.
8. export default class App extends Component {
9.   constructor() {
10.     super();
11.     this.state = {
12.       };
13.   }
14. }
15.
16. renderHeader = () => {
17.   return <SearchBar placeholder="Type Here..." lightTheme round />;
18. };
19.
20. renderFooter = () => {
21.   return (
22.     <View style={styles.footer}>
23.       <Text style={{textAlign:"center",fontSize:16}}> Toyota Thank you </Te
xt>
24.     </View>
25.   );
26. };
27.
28. renderSeparator = () => {
29.   return (
30.     <View
31.       style={{
32.         height: 1,
33.         backgroundColor: "#dddddd",
34.       } }
35.     />
36.   );
37. };
38.
39. renderItem={({item})=>{
```

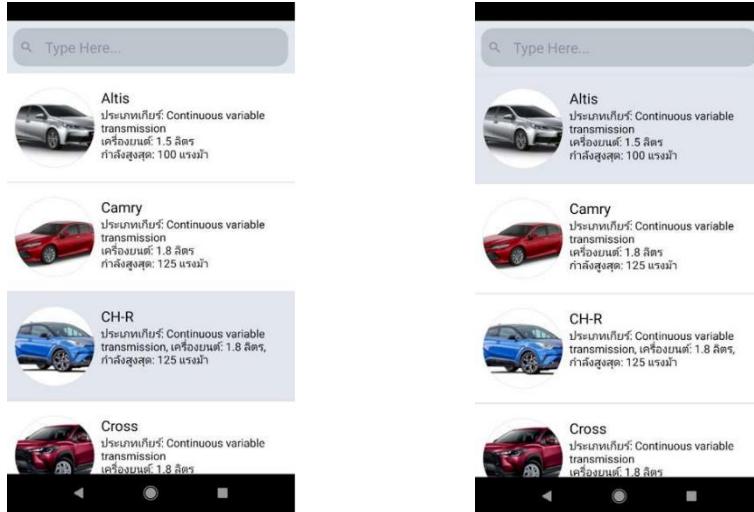
```

40.      return (
41.        <View>
42.          <View style={styles.itemStyle}>
43.            <Image style={styles.image} source={{uri:item.url}}/>
44.            <View style={{paddingLeft:8, flex:1}}>
45.              <Text style={styles.title}>{item.name}</Text>
46.              <Text>{item.description}</Text>
47.            </View>
48.          </View>
49.        </View>
50.      );
51.    }
52.
53.    render(props) {
54.      return (
55.        <View style={{flex:1}}>
56.          <FlatList
57.            data={data.car}
58.            renderItem={this.renderItem}
59.            keyExtractor={item => item.id}
60.            ItemSeparatorComponent={this.renderSeparator}
61.            ListHeaderComponent={this.renderHeader}
62.            ListFooterComponent={this.renderFooter}
63.          />
64.        </View>
65.      );
66.    }
67.  }
68. const styles = StyleSheet.create({
69.   itemStyle: {
70.     flexDirection:'row',
71.     padding: 10,
72.     marginVertical: 8,
73.   },title: {
74.     fontSize: 18,
75.   },image: {
76.     height:100,
77.     width:100,
78.     borderRadius:50,
79.     borderWidth:0.5,
80.     borderColor:"#dddddd"
81.   },footer: {
82.     padding:8,
83.     backgroundColor:"#e2e6ef",
84.     justifyContent:"center",
85.     alignContent:"center"
86.   }
87. });
88.

```

- การควบคุม Event เมื่อมีการกดเลือก ในการควบคุม Event ที่เกิดขึ้นภายใน renderItem function จะทำเหมือนดังเช่นการควบคุม Event ใน Custom component ซึ่งมีหลายวิธีตามที่ได้กล่าวมาแล้วในบทก่อนหน้านี้โดยในที่นี้ผู้เขียนจะแนะนำให้ใช้วิธีการเปลี่ยน state เพื่อควบคุม UI เพื่อความสะดวกในการเขียนโปรแกรม โดยตัวอย่างจะทำการเปลี่ยนสีพื้นหลังของ item เมื่อถูกเลือก

ตัวอย่างผลลัพธ์การทำงาน



ตัวอย่างโค้ดส่วนของ renderItem

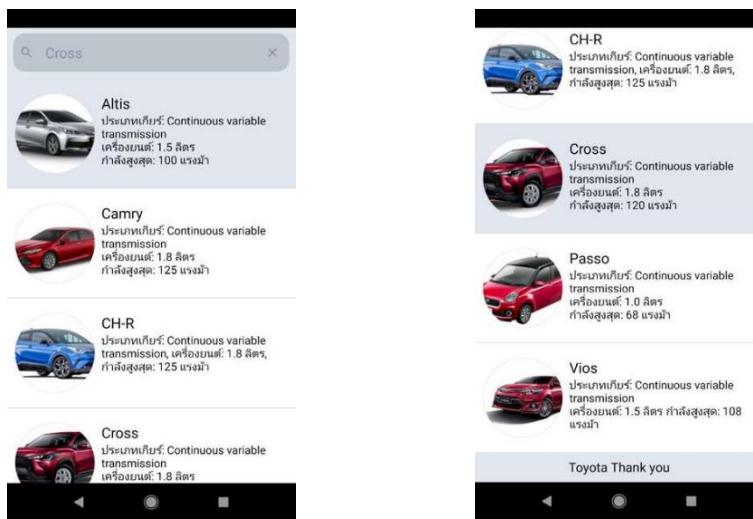
```

renderItem={({item})=>{
  let backgroundColor = null
  if(this.state.selectedID==item.id){
    backgroundColor="#e2e6ef"
  }else{
    backgroundColor="white"
  }
  return(
    <View>
      <TouchableOpacity style={{backgroundColor:backgroundColor}} onPress={()=>this.setState({selectedID:item.id})}>
        <View style={styles.itemStyle}>
          <Image style={styles.image} source={{uri:item.url}}/>
          <View style={{paddingLeft:8, flex:1}}>
            <Text style={styles.title}>{item.name}</Text>
            <Text>{item.description}</Text>
          </View>
        </View>
      </TouchableOpacity>
    </View>
  );
}
}

```

- การใช้งาน Callback Function ใน FlatList โดยทั่วไป FlatList function สำเร็จรูปให้เราใช้งานได้โดยไม่จำเป็นต้องเขียนเอง เช่น Function ที่ใช้ scrollToItem() และ scrollToIndex() โดยการใช้งาน Function เหล่านี้จะไม่ได้เรียกใช้ใน Tag <FlatList> โดยตรงโดยมากมักจะใช้งานเมื่อเกิด Event ต่างๆ ซึ่งเราจะต้องทำ Ref เพื่อให้สามารถใช้งาน Function เหล่านั้นนอก Tag ได้

ตัวอย่างผลลัพธ์การทำงาน



ตัวอย่าง Code ที่มีการสร้าง Ref ใน FlatList

```
<FlatList
  data={data.car}
  renderItem={this.renderItem}
  keyExtractor={item => item.id}
  ItemSeparatorComponent={this.renderSeparator}
  ListHeaderComponent={this.renderHeader}
  ListFooterComponent={this.renderFooter}
  ref={(ref) => { this.flatListRef = ref; }}
/>
```

ตัวอย่าง Code ที่มีเรียกใช้งาน Ref ใน FlatList

```
onSearch=(text)=>{
  this.setState({searchText:text});
  for(let i=0;i<data.car.length;i++){
    if(this.state.searchText==data.car[i].name){
      this.flatListRef.scrollToIndex({index:i});
      this.setState({selectedID:data.car[i].id})
    }
  }
}
```

ตัวอย่าง Code ทั้งหมด

```
1. import React, { Component } from 'react';
2. import {
3.   View,FlatList,TouchableOpacity,StyleSheet,Text,Alert,Image
4. } from 'react-native';
5. import * as data from './data.json'
6. import { List, ListItem, SearchBar } from "react-native-elements";
7.
8. export default class App extends Component {
9.   constructor() {
10.     super();
11.     this.state = {
```

```

12.         selectedID:null,
13.         searchText:null
14.     };
15.
16. }
17. onSearch=(text)=>{
18.     this.setState({searchText:text});
19.     for(let i=0;i<data.car.length;i++) {
20.         if(this.state.searchText==data.car[i].name) {
21.             this.flatListRef.scrollToIndex({index:i});
22.             this.setState({selectedID:data.car[i].id})
23.         }
24.     }
25. }
26.
27. renderHeader = () => {
28.     return <SearchBar placeholder="Type Here..." lightTheme round
29.             onChangeText={this.onSearch} value={this.state.searchText}/>;
30. };
31. renderFooter = () => {
32.     return(
33.         <View style={styles.footer}>
34.             <Text style={{textAlign:"center",fontSize:16}}> Toyota Thank you </Te
xt>
35.         </View>
36.     );
37. };
38.
39. renderSeparator = () => {
40.     return (
41.         <View
42.             style={{
43.                 height: 1,
44.                 backgroundColor: "#dddddd",
45.             }}
46.         />
47.     );
48. };
49.
50. renderItem={({item})=>{
51.     let backgroundColor = null
52.     if(this.state.selectedID==item.id){
53.         backgroundColor="#e2e6ef"
54.     }else{
55.         backgroundColor="white"
56.     }
57.     return(
58.         <View>
59.             <TouchableOpacity style={{backgroundColor:backgroundColor}} onPress={()=>this.setState({selectedID:item.id})}>
60.                 <View style={styles.itemStyle}>
61.                     <Image style={styles.image} source={{uri:item.url}}/>
62.                     <View style={{paddingLeft:8, flex:1}}>
63.                         <Text style={styles.title}>{item.name}</Text>
64.                         <Text>{item.description}</Text>
65.                     </View>
66.                 </View>
67.             </TouchableOpacity>
68.         </View>
69.     );
70. }
71. }
72. render(props) {
73.     return (

```

```
74.      <View style={{flex:1}}>
75.        <FlatList
76.          data={data.car}
77.          renderItem={this.renderItem}
78.          keyExtractor={item => item.id}
79.          ItemSeparatorComponent={this.renderSeparator}
80.          ListHeaderComponent={this.renderHeader}
81.          ListFooterComponent={this.renderFooter}
82.          ref={(ref) => { this.flatListRef = ref; }}
83.        />
84.      </View>
85.    );
86.  }
87. }
88. const styles = StyleSheet.create({
89.   itemStyle: {
90.     flexDirection:'row',
91.     padding: 10,
92.     marginVertical: 8,
93.   },title: {
94.     fontSize: 18,
95.   },image:{
96.     height:100,
97.     width:100,
98.     borderRadius:50,
99.     borderWidth:0.5,
100.     borderColor:"#dddddd"
101.   },footer:{
102.     padding:8,
103.     backgroundColor:"#e2e6ef",
104.     justifyContent:"center",
105.     alignContent:"center"
106.   }
107. });
108.
```

Digital Academy Thailand

การประยุกต์ใช้ Card เพื่อตกแต่ง FlatList

ในการตกแต่ง List นอกเหนือจากการตกแต่งด้วย Custom component แล้วเรายังสามารถใช้ Card เพื่อช่วยในการตกแต่งให้ง่ายขึ้นด้วยโดย Card นี้ถูกนำมาใช้ใน Application ชื่อดังอย่าง Facebook โดยมีจุดเด่นคือง่ายต่อการใช้งาน มีการตั้งค่า Style ของ Component เป็นองค์รวมให้แล้ว โดยการใช้งาน Card มีวิธีการดังนี้

การติดตั้ง

ในการใช้งาน Card จะต้องทำการ Install Component ให้เรียบร้อยเสียก่อนเนื่องจาก Component นี้ไม่ได้เป็นส่วนหนึ่งของ React-Native Component ที่ถูกติดตั้งมาตอนสร้างโปรเจคตามคำสั่งด้านล่าง

```
yarn add react-native-paper
```

//Or

```
npm install react-native-paper
```

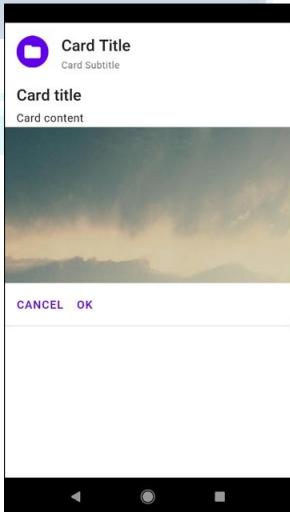
โดยการเรียกใช้งานเราจะต้องทำการ Import Component ต่างๆ ที่อยู่ในไลบรารี 'react-native-paper' เสียเสียก่อนตามตัวอย่างด้านล่าง

```
import { Avatar, Button, Card, Title, Paragraph } from 'react-native-paper';
```

ตัวอย่าง Component เพิ่มเติมสามารถดูได้ที่ <https://callstack.github.io/react-native-paper/>

การเรียกใช้งาน

การใช้งาน Card โดยส่วนมากจะใช้งานควบคู่กับ Component อื่นๆตามตัวอย่างด้านล่าง



```
<Card>
  <Card.Title title="Card Title"
             subtitle="Card Subtitle"
             left={()=>(<Avatar.Icon size={50}
                           icon="folder" />)} />

  <Card.Content>
    <Title>Card title</Title>
    <Paragraph>Card content</Paragraph>
  </Card.Content>

  <Card.Cover source={{ uri:
    'https://picsum.photos/700' }} />

  <Card.Actions>
    <Button>Cancel</Button>
    <Button>Ok</Button>
  </Card.Actions>
</Card>
```

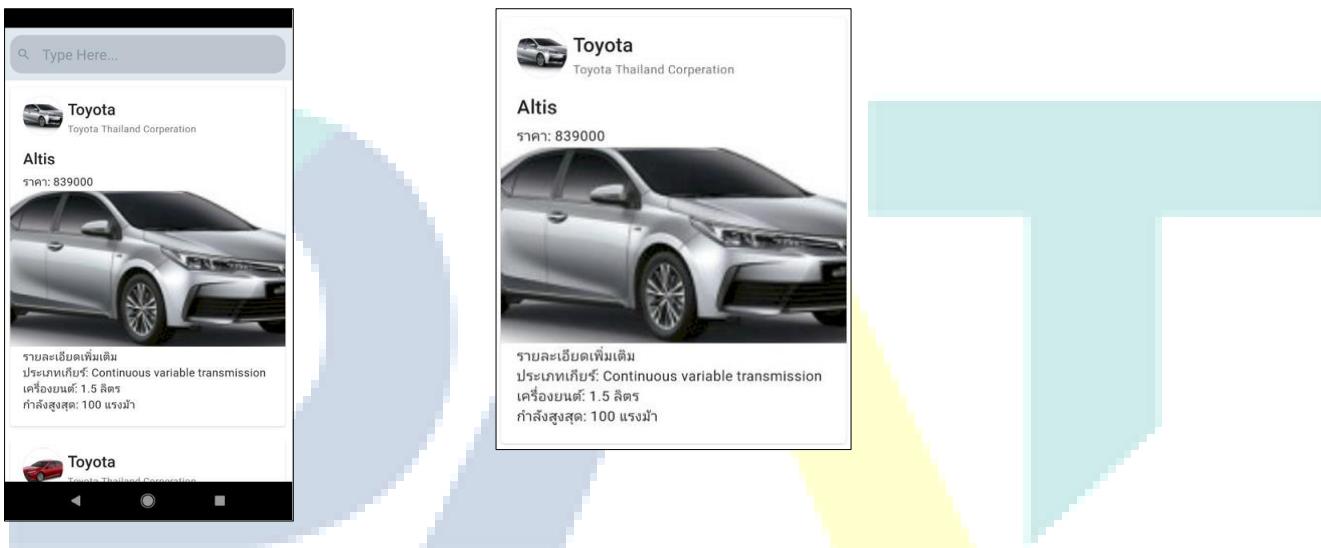
ตัวอย่างการใช้งาน Card ร่วมกับ FlatList

ในตัวอย่างนี้เราจะใช้ชุดข้อมูลเดิมก่อนหน้านี้แต่เราจะทำการเปลี่ยนแปลง Custom component ใน renderItem ใหม่ทั้งหมด โดยใช้ Card แทนโดยสามารถแบ่งออกได้ 3 ส่วนคือ

1. Card.Title เป็นส่วนแสดงข้อมูลในส่วนหัวซึ่งในนี้เราจะทำการแสดง Title, icon และ subtitle
2. Card.Content เป็นส่วนที่แสดงข้อมูลรายละเอียดต่างๆในรูปแบบข้อความ
3. Card.Cover ในส่วนนี้เราจะทำการแสดงรูปภาพขนาดใหญ่

ในการจัดเรียง ส่วนต่างๆของ Card เราสามารถจัดเรียงอย่างไรก็ได้ขึ้นกับวัตถุประสงค์ในการออกแบบ

ตัวอย่างผลลัพธ์การทำงาน



ตัวอย่าง Code

```
1. import React, { Component } from 'react';
2. import {
3.   View, FlatList, TouchableOpacity, StyleSheet, Text, Alert, Image
4. } from 'react-native';
5. import * as data from './data.json'
6. import { List, ListItem, SearchBar } from "react-native-elements";
7. import { Avatar, Button, Card, Title, Paragraph } from 'react-native-paper';
8.
9. export default class App extends Component {
10.   constructor() {
11.     super();
12.     this.state = {
13.       selectedID:null,
14.       searchText:null
15.     };
16.
17.   }
18.
19.   renderHeader = () => {
20.     return <SearchBar placeholder="Type Here..." lightTheme round />;
21.   };
22. }
```

```
23.     renderFooter = () => {
24.       return(
25.         <View style={styles.footer}>
26.           <Text style={{textAlign:"center",fontSize:16}}> Toyota Thank you </Te
xt>
27.         </View>
28.       );
29.     };
30.     renderItem=({item})=>{
31.
32.       return(
33.         <View style={{padding:8}}>
34.           <Card>
35.             <Card.Title title="Toyota" subtitle="Toyota Thailand Corporation" left={(
) =>(<Avatar.Image size={50} source={{uri:item.url}}/>) } />
36.             <Card.Content>
37.               <Title>{item.name}</Title>
38.               <Paragraph>{"เวลา: "+item.price}</Paragraph>
39.             </Card.Content>
40.             <Card.Cover source={{ uri: item.url}} />
41.             <Card.Content>
42.               <Paragraph>{"รายละเอียดเพิ่มเติม \n"+item.description}</Paragraph>
43.             </Card.Content>
44.           </Card>
45.         </View>
46.
47.       );
48.     }
49.     render(props) {
50.       return (
51.         <View style={{flex:1}}>
52.           <FlatList
53.             data={data.car}
54.             renderItem={this.renderItem}
55.             keyExtractor={item => item.id}
56.             ListHeaderComponent={this.renderHeader}
57.             ListFooterComponent={this.renderFooter}
58.             ref={(ref) => { this.flatListRef = ref; }}
59.           />
60.         </View>
61.       );
62.     }
63.   }
64.   const styles = StyleSheet.create({
65.     footer:{
66.       padding:8,
67.       backgroundColor:"#e2e6ef",
68.       justifyContent:"center",
69.       alignContent:"center"
70.     }
71.   });

```

บทที่ 6 การใช้งาน Navigation

การนำทาง Navigation

การนำทาง(Navigation) เราอาจจะไม่คุ้นเคยมากนักเท่ากับคำว่าการควบคุมการเปลี่ยน Page(Page Control) ซึ่งทั้งสองแบบนี้คือสิ่งเดียวกันในการพัฒนาโมบายแอพพลิเคชั่น การสร้างการนำทางเป็นสิ่งที่จำเป็นในทุกๆแอพพลิเคชั่น โดยจะมีลักษณะการสร้างที่แตกต่างกันออกไปขึ้นอยู่กับเครื่องมือที่ใช้งาน หากเราเคยเขียนโปรแกรมแบบ native มา ก่อน ใน Android studio เราอาจจะคุ้นเคยกับการควบคุม Activity และใน Xcode ก็จะใช้ Story Board ในการควบคุมการเปลี่ยน Page ของแอพพลิเคชั่น ใน React-Native นี้เราจะเรียกว่าการนำทาง Navigation ซึ่งการสร้าง Navigation ไม่ใช่เรื่องยาก ซึ่งเราจำเป็นจะต้องทำความเข้าใจการใช้งานเสียก่อน

การสร้าง Navigation ใน React-Native เราจะใช้ Library ที่มีชื่อว่า React Navigation ในการช่วยสร้างโดยเจ้า React Navigation Lib นี้มีหลายเวอร์ชันซึ่ง ในปัจจุบันคือเวอร์ชัน 5.x ซึ่งเวอร์ชันนี้เป็นเวอร์ชันที่มีความเสถียรสุด(Stable Version) ซึ่งจะมีลักษณะต่างจากเวอร์ชันเก่าอย่างเห็นได้ชัดคือ มีการแยก ต่างๆออกจากกันไม่ได้รวมไว้อยู่ใน Packageเดียวเหมือนอย่างแต่ก่อนซึ่งจะมีข้อดีคือ เราสามารถติดตั้งเฉพาะส่วนที่ใช้งานเท่านั้น ทำให้แอพพลิเคชั่นเรามีขนาดไม่เกินความจริง การใช้งาน React Navigation เราจะต้องสร้างตัวนำทาง(Navigator) ที่มีหน้าที่ควบคุมการเปลี่ยน Page ในรูปแบบต่างแบ่งตามลักษณะ UI ที่แสดงผลได้ 3 ประเภทใหญ่ๆดังนี้ Stack Navigator, Drawer Navigator, Tab Navigator การใช้งานแต่ละแบบจะกล่าวในลำดับถัดไป

Digital Academy Thailand

การติดตั้งเครื่องมือ

ก่อนที่เราจะใช้งาน React Navigation เราจะต้องทำการติดตั้ง Package หลักต่างๆให้เรียบร้อยเสียก่อนที่จะติดตั้ง Navigator อีกที ซึ่ง Package หลักจะประกอบไปด้วย

Package	หน้าที่
react-navigation/native	เป็นยูทิลิตี้หลักที่ทุก Navigator เรียกใช้งานในการสร้างการนำทาง
react-native-gesture-handler	เป็นตัวจัดการและตรวจสอบการสัมผัสที่เกิดขึ้นในแอพพลิเคชัน
react-native-reanimated	เป็นตัวกลางในการใช้งาน Animated library ที่มีความยืดหยุ่นและสะดวกกว่าการใช้งานแบบตรงๆ
react-native-screens	เป็นตัว Container สำหรับตัวนำทาง Navagation
react-native-safe-area-context	เป็นตัวจัดการตำแหน่งของการแสดงผลและการสัมผัสที่มีความยืดหยุ่นสูง
react-native-community/masked-view	เป็นตัวสร้างหน้าจอเพื่อตกแต่ง Component ให้มีความสวยงาม

การติดตั้ง Package หลักจะมีขั้นตอนที่แตกต่างกันขึ้นอยู่กับลักษณะโปรเจคที่สร้างสามารถแบ่งออกได้ 2 แบบดังนี้

1. การติดตั้งในโปรเจคแบบ Expo managed project ในการติดตั้งแบบนี้ไม่มีอะไรซับซ้อนมากเราสามารถใช้คำสั่งด้านล่างนี้

```
npm install @react-navigation/native
```

```
expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view
```

2. การติดตั้งในโปรเจคแบบ React Native project ในการติดตั้งในโปรเจคแบบนี้จะมีความซับซ้อนกว่าแบบแรก เเละน้อยกว่าคือ ในตัวโปรดเจคจะมี Native Package อยู่ซึ่งใน iOS Package เราจะต้องทำการ Linking Library เสียก่อนจึงจะใช้งานได้ โดยพิมพ์คำสั่งตามดังนี้

```
npm install @react-navigation/native
```

```
npm install react-native-reanimated react-native-gesture-handler react-native-screens react-native-safe-area-context @react-native-community/masked-view
```

```
npx pod-install ios
```

หลังจากที่เราติดตั้ง Package หลักเรียบร้อยแล้ว ขั้นถัดมาจะเป็นการติดตั้ง Navigator และเรียกใช้งานตัวนำทาง ซึ่งวิธีการสร้างตัวนำทางและการเรียกใช้งานจะมีวิธีที่แตกต่างกันดังต่อไปนี้

การใช้งานตัวนำทางแบบ Stack Navigator

ตัวนำทางแบบ Stack Navigator คือตัวที่ควบคุมการเปลี่ยน Screen โดยจะมีการนำ Screen ใหม่มาแสดงผลโดยวางซ้อนทับกับ Screen เก่า ผู้อ่านอาจจะเกิดความสับสนให้ลองจินตนาการถึงการวางแผนซ้อนกันเป็นชั้นๆ เช่น Stack Navigator จะมีลักษณะเช่นเดียวกันคือเมื่อเราเปิด Screen ขึ้นมาใหม่ก็เปลี่ยนเหมือนกับการออกจากวงจรชั้นนอกและเมื่อเรากดปุ่ม Back ในสมาร์ทโฟนเพื่อกลับไปยัง Screen ก่อนหน้าก็เปลี่ยนเหมือนกับการที่เราหยิบจานใบบนสุดออกแล้วมองดูจานที่เหลือ โดยทั่วไปตัวนำทางแบบ Stack Navigator จะไม่มีส่วนควบคุม(Control Screen UI)แบบตัวนำทางแบบอื่นๆ เวลาใช้งานเราจะจำเป็นต้องสร้างปุ่มหรือตัวควบคุมคุณภาพเปลี่ยน Screen ขึ้นมา ตัวอย่างส่วนของโมบายแอพพลิเคชันที่มีการใช้งานตัวนำทางแบบ Stack Navigator เช่น การเปลี่ยน Screen จากหน้าแรกของแอพพลิเคชัน(Splash Screen)ไปยังหน้าหลัก/Home Screen), การเปลี่ยน Screen จากหน้าล็อกอิน(Login Screen)ไปยังหน้าสมัครสมาชิก(Registration Screen) เป็นต้น

การติดตั้งเครื่องมือ

ก่อนใช้งานตัวนำทางแบบ Stack Navigator เราจะต้องทำการติดตั้ง Package ย่อของตัวนำทาง Stack Navigator นี้ให้เรียบร้อยโดยเสียก่อนโดยมีคำสั่งดังนี้

```
npm install @react-navigation/stack
```

การใช้งาน Stack Navigator

การสร้าง Stack Navigator เราจะต้อง Import Component จาก `@react-navigation/stack` โดยมีตัวอย่างด้านล่าง

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

การเรียกใช้งาน Stack Navigator เราจะต้องสร้าง Object Component ผ่าน Function `createStackNavigator` โดยมีตัวอย่างด้านล่าง

```
const Stack = createStackNavigator();
<Stack.Navigator>....</Stack.Navigator>
```

การสร้าง Screen ใน Stack Navigator เราจำเป็นจะต้องทำการเพิ่ม Screen โดยเราจะส่งค่าพารามิเตอร์ด้วย Props ดังนี้ `name` คือชื่อของ Screen และ `component` คือ screen component ที่มีลักษณะเป็น Function Component หรือ Class Component ทั้งนี้ในการใช้งาน Navigation version 5.x นักพัฒนาได้แนะนำให้สร้าง Screen Component ในรูปแบบ Function แต่หากเราต้องการสร้างเป็น Class Component นักพัฒนาได้แนะนำให้เรียกใช้งานผ่าน Function อีกที

```
<Stack.Screen name="Screen Name" component={Screen_Component_Function} />
```

ตัวอย่างการสร้าง Screen ใน Stack Navigator

```
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();
function MyStack() {
  return (
    <Stack.Navigator>
      <Stack.Screen name="Home" component={Home} />
      <Stack.Screen name="Notifications" component={Notifications} />
      <Stack.Screen name="Profile" component={Profile} />
      <Stack.Screen name="Settings" component={Settings} />
    </Stack.Navigator>
  );
}
```

การตกแต่ง Stack Navigator

การตกแต่งและตั้งค่าใน Stack Navigator จะกระทําภายในหน้า(Screen)ของแต่ละหน้าโดยใช้ Props Options ตามตัวอย่างโค้ดด้านล่าง

```
<Stack.Screen options={{ }} name="Screen_Name" component={Component} />
```

ตัวอย่างการตกแต่งที่ใช้บ่อย

Options	การใช้งาน
title	แสดงชื่อบนໄຕเต็ลแท็บ options={{title: "Title Name"}}
headerShown	เปิดและปิดการแสดงໄຕเต็ลแท็บ true : แสดง(Defualt), false : ซ่อน options={{ headerShown: false}}
headerTitleAlign	จัดตำแหน่งของ Title left (Defualt), center options={{ headerTitleAlign: "center"}}
headerTintColor	เปลี่ยนสีตัวอักษรของ Title options={{ headerTintColor: "center"}}
headerBackground	เปลี่ยนพื้นหลังของໄຕเต็ลแท็บ function background(){ return(<View style={{flex:1, backgroundColor:"red"}>); } options={{headerBackground :background}}
headerRight	เพิ่ม Component ลงในໄຕเต็ลแท็บฝั่งขวาพื้นหลังของໄຕเต็ลแท็บ function rightHead (){ return(<Button title="OK"/>);}

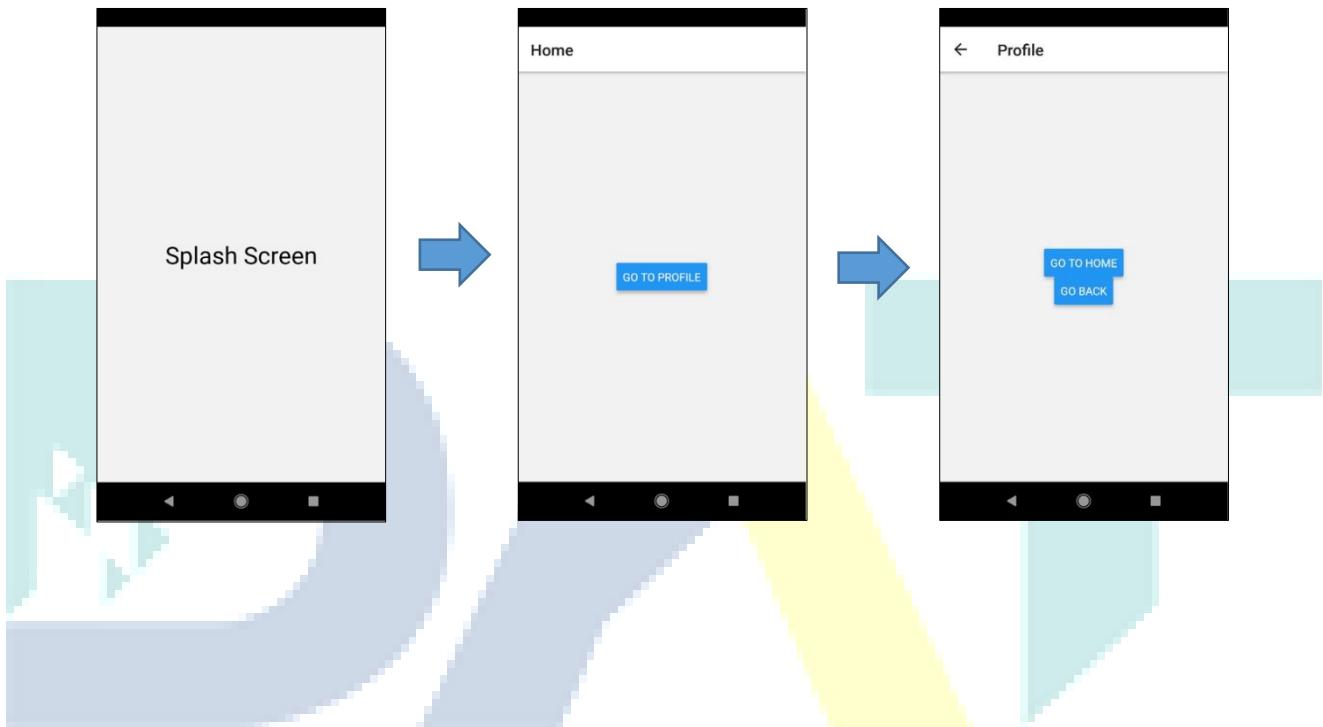
	options={{headerBackground :rightHead}}
headerLeft	เพิ่ม Component ลงในไฟเติลแท็บฝั่งซ้ายพื้นหลังของไฟเติลแท็บ function rightHead (){ return(<Button title="OK"/>);} options={{headerBackground :rightHead}}

ตัวอย่าง <https://reactnavigation.org/docs/stack-navigator>



ตัวอย่างการใช้งาน Stack Navigator

ในตัวอย่างนี้เราจะสร้างการนำทางโดยใช้ตัวนำทางแบบ Stack Navigation โดยเมื่อทำการเปิด App ขึ้นมาจะแสดงหน้าแรก(Splash Screen) จากนั้นรอประมาณ 2.5 วินาทีจะทำการเปลี่ยนหน้าไปยังหน้าหลัก(Home Screen) เมื่อยื่นในหน้า Home เราสามารถกดเปลี่ยนหน้าไปยังหน้าโปรไฟล์(Profile Screen) ด้วยการกดปุ่ม “GO TO PROFILE” หากเราอยู่ในหน้าโปรไฟล์แล้วเราก็จะกลับมายังหน้าหลักได้ด้วยการกดปุ่มกลับ(Back) ที่อยู่ด้านบนสุดหรือกดปุ่ม “GO TO HOME” หรือ “GO BACK” ได้เช่นกัน ตัวอย่าง User Interface แสดงด้านล่าง



Digital Academy Thailand

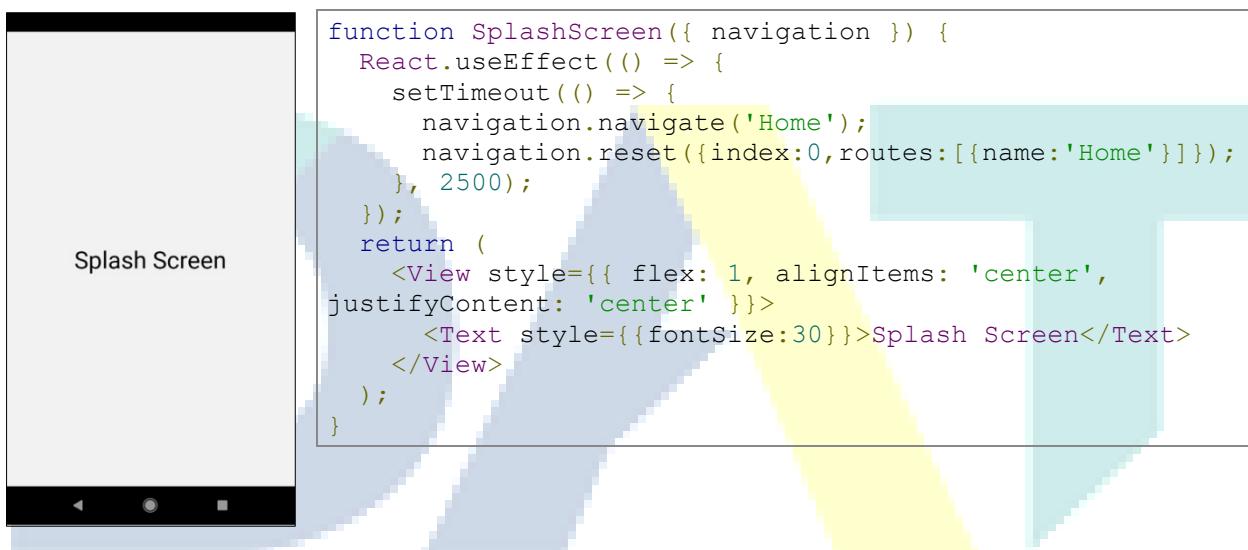
หน้าแรก(Splash Screen)

การควบคุมการนำทางแบบ Stack Navigator ในหน้าแรกนี้ เราจะทำการตัวค่า Timmer ให้หน่วงเวลา 2.5 วินาทีแล้วจึงทำการเปลี่ยนหน้าไปยังหน้าหลัก(Home Screen) ด้วยใช้ Navigation Hook(อธิบายเพิ่มเติมด้านล่าง) ด้วยคำสั่ง

```
navigation.navigate('Home');
```

เนื่องจากหน้าแรกของทุกๆแอพพลิเคชันจะแสดงแค่ครั้งเดียวและไม่สามารถทำการกลับมายังหน้านี้ได้อีกจนกว่าจะมีการปิดและเปิดมันขึ้นมาใหม่ด้วยเหตุนี้เราจึงต้องทำรีเซ็ตสเตรค(Stack) ให้ตำแหน่งแรกสุดของเป็นหน้าหลัก(Home Screen)แทนที่หน้าเดิมคือหน้าแรก(Splash Screen) ด้วยคำสั่ง

```
navigation.reset({index:0,routes:[{name:'Home'}]});
```



Digital Academy Thailand

หน้าหลัก(Home Screen)

การควบคุมการนำทางแบบ Stack Navigator ในหน้าหลักนี้จะแตกต่างจากหน้าแรกที่มีการควบคุมด้วย Timer แต่ในหน้านี้จะใช้ปุ่มในการควบคุมการเปลี่ยนหน้าจากหน้าแรก(Home Screen) ไปยังหน้าโปรไฟล์(Profile Screen) ด้วยคำสั่งด้านล่าง

```
navigation.navigate('Profile')
```



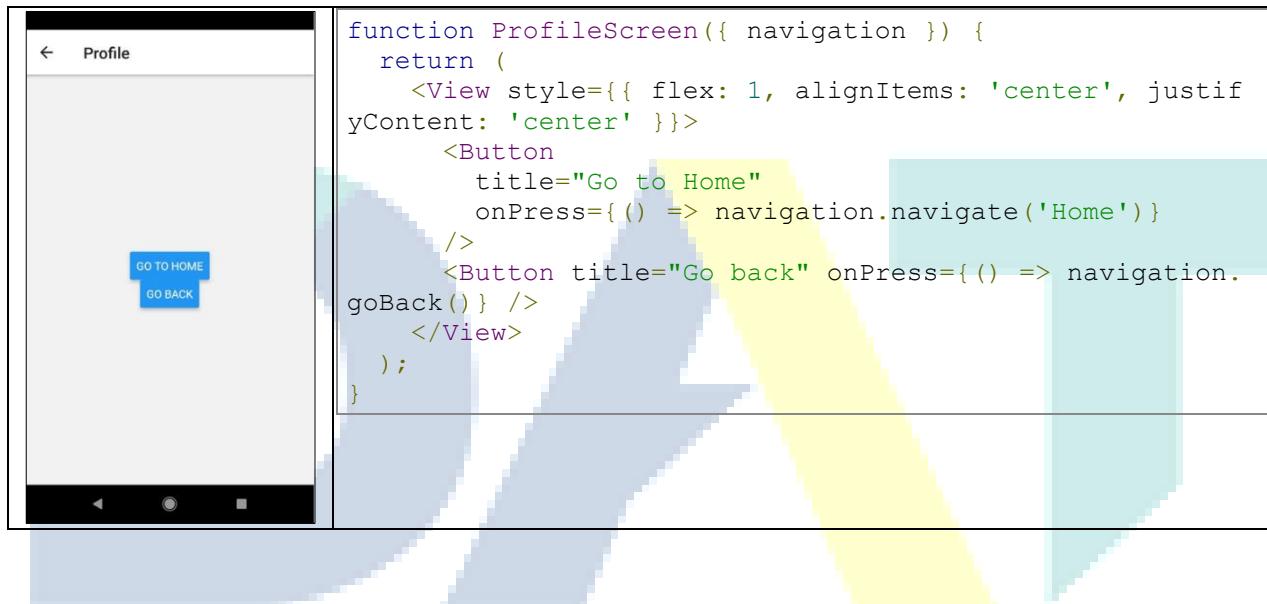
Digital Academy Thailand

หน้าโปรไฟล์(Profile Screen)

การควบคุมการนำทางแบบ Stack Navigator ในหน้าโปรไฟล์นี้จะเหมือนดังเช่นการควบคุมให้หน้าหลัก (Home Screen) คือจะใช้ปุ่มในการควบคุมการเปลี่ยนหน้าจากหน้าโปรไฟล์(Profile Screen) ไปยังหน้าหลัก(Home Screen) ด้วยปุ่ม “Go to Home” หรือหากเราต้องการกลับไปยังหน้าก่อนหน้าด้วยการกดปุ่ม “GO BACK” ก็สามารถทำได้เช่นกันโดยใช้คำสั่ง

```
navigation.goBack()
```

ทั้งนี้ในการสร้างตัวนำทางแบบ Stack Navigator เมื่อมีการเปลี่ยนหน้าไปยังหน้าอื่นจะมีการสร้างปุ่มกลับ(Back)ให้อัตโนมัติหากเราทำการแสดงตัวเต็ลาร์(Title bar) โดยที่เรามิ่งเป็นจะต้องเขียนโค้ดเพิ่มเติมก็ได้



Digital Academy Thailand

โค้ดทั้งหมดในการใช้งานตัวนำทางแบบ Stack Navigator

```
1. import * as React from 'react';
2. import { Button, View, Text } from 'react-native';
3. import { NavigationContainer } from '@react-navigation/native';
4. import { createStackNavigator } from '@react-navigation/stack';
5. function SplashScreen({ navigation }) {
6.   React.useEffect(() => {
7.     setTimeout(() => {
8.       navigation.navigate('Home');
9.       navigation.reset({index:0, routes:[{name:'Home'}]}));
10.      }, 2500);
11.    });
12.    return (
13.      <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
14.        <Text style={{fontSize:30}}>Splash Screen</Text>
15.      </View>);
16.    function HomeScreen({ navigation }) {
17.      return (
18.        <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
19.          <Button
20.            title="Go to Profile"
21.            onPress={() => navigation.navigate('Profile')}
22.          />
23.        </View>
24.      );
25.    function ProfileScreen({ navigation }) {
26.      return (
27.        <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
28.          <Button
29.            title="Go to Home"
30.            onPress={() => navigation.navigate('Home')}
31.          />
32.          <Button title="Go back" onPress={() => navigation.goBack()} />
33.        </View>
34.      );
35.    const Stack = createStackNavigator();
36.    function MyStack() {
37.      return (
38.        <Stack.Navigator>
39.          <Stack.Screen options={{headerShown:false}} name="Splash"
40.            component={SplashScreen} />
41.          <Stack.Screen name="Home" component={HomeScreen} />
42.          <Stack.Screen name="Profile" component={ProfileScreen} />
43.        </Stack.Navigator>
44.      );
45.    export default function App() {
46.      return (
47.        <NavigationContainer>
48.          <MyStack />
49.        </NavigationContainer>);
```

การใช้งานตัวนำทางแบบ Tab Navigation

ตัวนำทางแบบ Tab Navigator คือตัวที่ควบคุมการเปลี่ยน Screen ที่มีลักษณะคล้ายกับ Stack Navigator ก่อนหน้านี้แต่จะมีส่วนพิเศษที่เรียกว่าส่วนแท็บควบคุม(Tab Control Screen) เพิ่มเข้ามาเพื่อควบคุมการเปลี่ยนหน้าแอปพลิเคชัน โดยตัวนำทางแบบ Tab Navigator สามารถแบ่งออกเป็น 2 ประเภทตามลักษณะการแสดงผลของแท็บควบคุมคือ Bottom Tab Navigator และ Top Tab Navigator

การใช้งาน Bottom Tab Navigator

ตัวนำทางแบบ Bottom Tab Navigator จะมีแท็บควบคุมอยู่ด้านล่างสุดของหน้าจอเราสามารถกดเปลี่ยนหน้าจอได้จากแท็บด้านล่างนี้

การติดตั้งเครื่องมือเพื่อใช้งาน Bottom Tab Navigator

ก่อนใช้งานตัวนำทางแบบ Bottom Navigator เราจะต้องทำการติดตั้ง Third Party Library ของตัวนำทาง Bottom Navigator นี้ให้เรียบร้อยเสียก่อนโดยมีคำสั่งดังนี้

```
npm install @react-navigation/bottom-tabs
```

การใช้งาน Bottom Tab Navigator

การสร้าง Bottom Tab Navigator เราจะต้อง Import Component จาก @react-navigation/bottom-tabs โดยมีตัวอย่างด้านล่าง

```
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
```

Digital Academy Thailand

การเรียกใช้งาน Bottom Navigator เราจะต้องสร้าง Object Component ผ่าน Function createStackNavigator โดยมีตัวอย่างด้านล่าง

```
const Tab = createBottomTabNavigator();
<Tab.Navigator>....</Tab.Navigator>
```

การสร้างหน้า(Screen) ใน Bottom Tab Navigator เราจำเป็นจะต้องทำการเพิ่ม Screen โดยเราจะส่งค่าพารามิเตอร์ด้วย Props ดังนี้ name คือชื่อของ Screen และ component คือ screen component ที่มีลักษณะเป็น Function Component หรือ Class Component ทั้งนี้ในการใช้งาน Navigation version 5.x นักพัฒนาได้แนะนำให้สร้าง Screen Component ในรูปแบบ Function แต่หากเราต้องการสร้างเป็น Class Component นักพัฒนาได้แนะนำให้เรียกใช้งานผ่าน Function อีกที

```
<Tab.Screen name="Screen Name" component={Screen_Component_Function} />
```

ตัวอย่างการสร้างหน้า(Screen) ใน Bottom Tab Navigator

```
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

const Tab = createBottomTabNavigator();

function MyTabs() {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Home" component={HomeScreen} />
      <Tab.Screen name="Settings" component={SettingsScreen} />
    </Tab.Navigator>
  );
}
```

การตกแต่ง Bottom Tab Navigator

การตกแต่งและตั้งค่าใน Bottom Tab Navigator สามารถแบ่งได้ 2 ส่วนคือส่วน Tab และส่วน Screen โดยการตั้งค่าในส่วน Tab จะต้องใช้ Props tabBarOptions ตามตัวอย่างโค้ดด้านล่าง

```
<Tab.Navigator tabBarOptions={{ . . . }}>
```

ตัวอย่างการตกแต่งที่ใช้บ่อย

Options	การใช้งาน
activeTintColor	เปลี่ยนสีไอคอนและข้อความในแท็บเมื่อมีการกดเลือก tabBarOptions={{activeTintColor: '#e91e63'}}
inactiveTintColor	เปลี่ยนสีไอคอนและข้อความในแท็บเมื่อไม่มีการกดเลือก tabBarOptions={{inactiveTintColor: 'gray'}}
activeBackgroundColor	เปลี่ยนแท็บเมื่อมีการกดเลือก tabBarOptions={{ activeBackgroundColor: '#e91e63' }}
inactiveBackgroundColor	เปลี่ยนแท็บเมื่อไม่มีการกดเลือก tabBarOptions={{ inactiveBackgroundColor: '#e91e63' }}

การตกแต่งในส่วน Screen จะต้องใช้ Props options ตามตัวอย่างโค้ดด้านล่าง

```
<Tab.Screen options={{...}} name="Screen_Name" component={Component} />
```

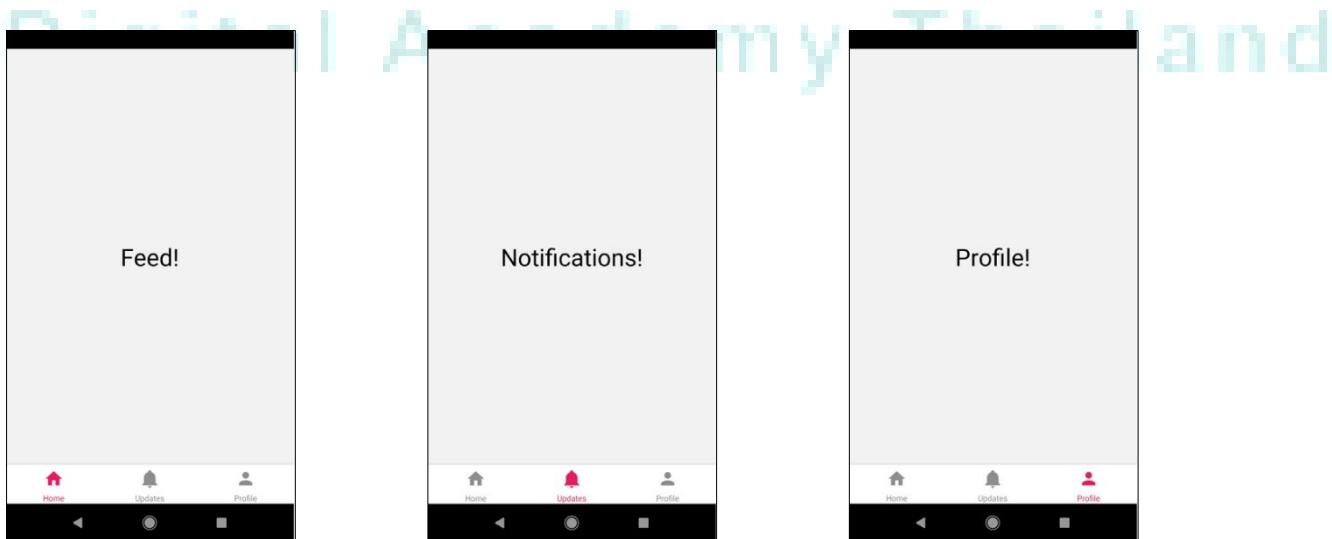
ตัวอย่างการตกแต่งที่ใช้บ่อย

Options	การใช้งาน
title	แสดงชื่อหน้าต่างแบบแท็บ options={{title: "Title Name"}}
tabBarIcon	กำหนดไอคอนในแท็บบาร์ options={{tabBarIcon: ({ color, size }) => (<MaterialCommunityIcons name="home" color={color} size={size} />)}}
tabBarLabel	กำหนดไอคอนในแท็บบาร์ options={{ tabBarLabel: 'Home' }}

ตัวอย่าง <https://reactnavigation.org/docs/stack-navigator>

ตัวอย่างการใช้งาน Bottom Tab Navigator

ในตัวอย่างนี้เราจะสร้างการนำทางโดยใช้ตัวนำทางแบบ Bottom Tab Navigator โดยตัวอย่างจะประกอบไปด้วยหน้าจำนวน 3 หน้าได้แก่ Feed, Profile, Notification เมื่อมีการเปิดแอปพลิเคชันขึ้นมาจะแสดงหน้า Feed เป็นหน้าแรกเนื่องจากเราสร้างหน้านี้เป็นลำดับแรกและเรามาสามารถกดเปลี่ยนหน้าได้ด้วยการกดไปที่ไอコンบริเวณแท็บควบคุมตัวอย่าง User Interface แสดงด้านล่าง



โค้ดทั้งหมดในการใช้งานตัวนำทางแบบ Bottom Tab Navigator

```
1. import * as React from 'react';
2. import { Text, View } from 'react-native';
3. import { NavigationContainer } from '@react-navigation/native';
4. import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
5. import { MaterialCommunityIcons } from '@expo/vector-icons';
6.
7. function Feed() {
8.   return (
9.     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
10.       <Text style={{fontSize:30}}>Feed!</Text>
11.     </View>
12.   );
13. }
14.
15. function Profile() {
16.   return (
17.     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
18.       <Text style={{fontSize:30}}>Profile!</Text>
19.     </View>
20.   );
21. }
22.
23. function Notifications() {
24.   return (
25.     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
26.       <Text style={{fontSize:30}}>Notifications!</Text>
27.     </View>
28.   );
29. }
30.
31. const Tab = createBottomTabNavigator();
32.
33. function MyTabs() {
34.   return (
35.     <Tab.Navigator
36.       initialRouteName="Feed"
37.       tabBarOptions={{
38.         activeTintColor: '#e91e63',
39.       }}
40.     >
41.       <Tab.Screen
42.         name="Feed"
43.         component={Feed}
44.         options={{
45.           tabBarLabel: 'Home',
46.           tabBarIcon: ({ color, size }) => (
47.             <MaterialCommunityIcons name="home" color={color} size={size} />
48.           ),
49.         }}
50.       />
51.       <Tab.Screen
52.         name="Notifications"
53.         component={Notifications}
54.         options={{
55.           tabBarLabel: 'Updates',
56.           tabBarIcon: ({ color, size }) => (
57.             <MaterialCommunityIcons name="bell" color={color} size={size} />
58.           ),
59.         }}
60.     
```

```
60.      />
61.      <Tab.Screen
62.        name="Profile"
63.        component={Profile}
64.        options={{
65.          tabBarLabel: 'Profile',
66.          tabBarIcon: ({ color, size }) => (
67.            <MaterialCommunityIcons name="account" color={color} size={size}>
68.              ,
69.            ) )
70.          />
71.        </Tab.Navigator>
72.      );
73.    }
74.
75.    export default function App() {
76.      return (
77.        <NavigationContainer>
78.          <MyTabs />
79.        </NavigationContainer>
80.      );
81.    }
82.
83.
```



Digital Academy Thailand

การใช้งาน Top Tab Navigator

ตัวนำทางแบบ Top Tab Navigator จะมีแท็บควบคุมอยู่ด้านบนสุดของหน้าจอ เราสามารถกดเปลี่ยนหน้าจอได้จากแท็บด้านบนนี้

การติดตั้งเครื่องมือเพื่อใช้งาน Top Tab Navigator

ก่อริใช้งานตัวนำทางแบบ Stack Navigator เราจะต้องทำการติดตั้ง Third Party Library ของตัวนำทาง Bottom Navigator นี้ให้เรียบร้อยเสียก่อนโดยมีคำสั่งดังนี้

```
npm install @react-navigation/material-top-tabs react-native-tab-view
```

การใช้งาน Top Tab Navigator

การสร้าง Top Tab Navigator เราจะต้อง Import Component จาก `@react-navigation/bottom-tabs` โดยมีตัวอย่างด้านล่าง

```
import { createMaterialTopTabNavigator } from '@react-navigation/material-top-tabs';
```

การเรียกใช้งาน Top Navigator เราจะต้องสร้าง Object Component ผ่าน Function

`createMaterialTopTabNavigator` โดยมีตัวอย่างด้านล่าง

```
const Tab = createMaterialTopTabNavigator();  
<Tab.Navigator>....</Tab.Navigator>
```

การสร้างหน้า(Screen) ใน Top Tab Navigator เราจำเป็นจะต้องทำการเพิ่ม Screen โดยเราจะส่งค่าพารามิเตอร์ด้วย Props ดังนี้ name คือชื่อของ Screen และ component คือ screen component ที่มีลักษณะเป็น Function Component หรือ Class Component ทั้งนี้ในการใช้งาน Navigation version 5.x นักพัฒนาได้แนะนำให้สร้าง Screen Component ในรูปแบบ Function แต่หากเราต้องการสร้างเป็น Class Component นักพัฒนาได้แนะนำให้เรียกใช้งานผ่าน Function อีกที

```
<Tab.Screen name="Screen Name" component={Screen_Component_Function} />
```

ตัวอย่างการสร้างหน้า(Screen) ใน Top Tab Navigator

```
import { createMaterialTopTabNavigator } from '@react-navigation/material-top-tabs';

const Tab = createMaterialTopTabNavigator();

function MyTabs() {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Home" component={HomeScreen} />
      <Tab.Screen name="Settings" component={SettingsScreen} />
    </Tab.Navigator>
  );
}
```

การตกแต่ง Top Tab Navigator

การตกแต่งและตั้งค่าใน Top Tab Navigator จะต้องใช้ Props tabBarOptions ตามตัวอย่างโค้ดด้านล่าง

```
<Tab.Navigator Options={{ . . . }}>
```

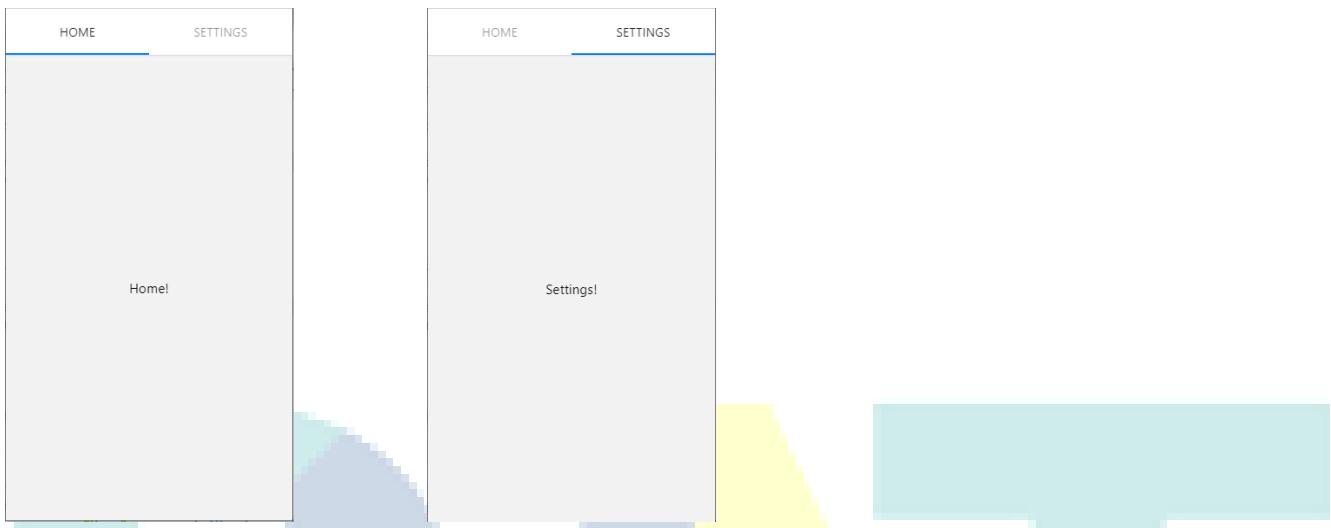
ตัวอย่างการตกแต่งที่ใช้บ่อย

Options	การใช้งาน
title	แสดงชื่อบนไตเติลแท็บ options={{title: "Title Name"}}
tabBarIcon	กำหนดไอคอนในแท็บบาร์ options={{tabBarIcon: ({ color, size }) => (<MaterialCommunityIcons name="home" color={color} size={size} />)}}
tabBarLabel	กำหนดไอคอนในแท็บบาร์ options={{ tabBarLabel: 'Home' }}

ตัวอย่าง <https://reactnavigation.org/docs/material-top-tab-navigator>

ตัวอย่างการใช้งาน Bottom Tab Navigator

ในตัวอย่างนี้เราจะสร้างการนำทางโดยใช้ตัวนำทางแบบ Bottom Tab Navigator โดยตัวอย่างจะประกอบไปด้วยหน้าจำนวน 2 หน้าได้แก่ Home, Settings เมื่อมีการเปิดแอปพลิเคชันขึ้นมาจะแสดงหน้า Home เป็นหน้าแรกเนื่องจากเราสร้างหน้านี้เป็นลำดับแรกและเรารสามารถกดเปลี่ยนหน้าได้ด้วยการกดไปที่ไอคอนบริเวณแท็บควบคุมตัวอย่าง User Interface แสดงด้านบน



โค้ดทั้งหมดในการใช้งานตัวนำทางแบบ Top Tab Navigator

```
1. import * as React from 'react';
2. import { Text, View } from 'react-native';
3. import { NavigationContainer } from '@react-navigation/native';
4. import { createMaterialTopTabNavigator } from '@react-navigation/material-top-tabs';
5.
6. function HomeScreen() {
7.   return (
8.     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
9.       <Text>Home!</Text>
10.      </View>
11.    );
12. }
13. function SettingsScreen() {
14.   return (
15.     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
16.       <Text>Settings!</Text>
17.     </View>
18.   );
19. }
20. const Tab = createMaterialTopTabNavigator();
21. export default function App() {
22.   return (
23.     <NavigationContainer>
24.       <Tab.Navigator>
25.         <Tab.Screen name="Home" component={HomeScreen} />
26.         <Tab.Screen name="Settings" component={SettingsScreen} />
27.       </Tab.Navigator>
28.     </NavigationContainer>
29.   );
30. }
```

การใช้งานตัวนำทางแบบ Drawer Navigation

ตัวนำทางแบบ Drawer Navigator คือตัวที่ควบคุมการเปลี่ยน Screen ที่มีลักษณะแตกต่าง Navigator ตัวอื่นๆ ก่อนหน้านี้ โดยจะมีส่วนพิเศษที่เรียกว่าส่วนสไตล์ควบคุม(Drawer Control Screen) เพิ่มเข้ามาเพื่อควบคุมการเปลี่ยนหน้าแอปพลิเคชัน โดยตัวนำทางแบบ Drawer Navigator สามารถใช้งานได้ดังนี้

การติดตั้งเครื่องมือ

ก่อนใช้งานตัวนำทางแบบ Drawer Navigator เราจะต้องทำการติดตั้ง Package ยอดของตัวนำทางประเภทนี้ให้เรียบร้อยเสียก่อนโดยมีคำสั่งดังนี้

```
npm install @react-navigation/drawer
```

การใช้งาน Drawer Navigator

การสร้าง Drawer Navigator เราจะต้อง Import Component จาก `@react-navigation/drawer` โดยมีตัวอย่างด้านล่าง

```
import { createDrawerNavigator } from '@react-navigation/drawer';
```

การเรียกใช้งาน Drawer Navigator เราจะต้องสร้าง Object Compoent ผ่าน Function `createDrawerNavigator` โดยมีตัวอย่างด้านล่าง

```
const Drawer = createDrawerNavigator();
<Drawer.Navigator>....</Drawer.Navigator>
```

การสร้าง Screen ใน Drawer Navigator เราจำเป็นจะต้องทำการเพิ่ม Screen โดยเราจะส่งค่าพารามิเตอร์ด้วย Props ดังนี้ `name` คือชื่อของ Screen และ `component` คือ screen component ที่มีลักษณะเป็น Function Component หรือ Class Component ทั้งนี้ในการใช้งาน Navigation version 5.x นักพัฒนาได้แนะนำให้สร้าง Screen Component ในรูปแบบ Function แต่หากเราต้องการสร้างเป็น Class Component นักพัฒนาได้แนะนำให้เรียกใช้งานผ่าน Function อีกที

```
<Drawer.Screen name="Screen Name" component={Screen_Component_Function} />
```

ตัวอย่างการสร้าง Screen ใน Drawer Navigator

```
function MyDrawer() {
  return (
    <Drawer.Navigator>
      <Drawer.Screen name="Feed" component={Feed} />
      <Drawer.Screen name="Article" component={Article} />
    </Drawer.Navigator>
  );
}
```

การตกแต่ง Drawer Navigator

การตกแต่งและตั้งค่าใน Drawer Navigator จะกระทำภายในหน้า(Screen)ของแต่ละหน้า และในส่วนของแท็บสไลด์(Drawer) โดยการตกแต่งภายในหน้าสามารถทำได้ด้วยคำสั่ง

```
<Drawer.Screen options={{...}} name="Screen_Name" component={Component} />
```

ตัวอย่างการตกแต่งที่ใช้บ่อย

Options	การใช้งาน
title	แสดงชื่อบนไตเติลแท็บ options={{title: "Title Name"}}
drawerLabel	กำหนดชื่อของสไลด์แท็บ options={{ drawerLabel: 'Profile' }}
drawerIcon	กำหนดไอคอนที่แสดงในแท็บสไลด์ options={{ drawerIcon: ({ color, size }) => (<MaterialCommunityIcons name="details" color={color} size={size} />)} }

ตัวอย่าง <https://reactnavigation.org/docs/drawer-navigator>

โดยการตกแต่งภายในสไลด์แท็บจะต้องมีการสร้าง เป็น Component Function

```
<Drawer.Navigator drawerContent={props => <CustomDrawerContent {...props} />}>
```

```
function CustomDrawerContent(props) {  
  return (  
    <DrawerContentScrollView {...props}>  
      <DrawerItemList {...props} />  
      <DrawerItem label="Help" onPress={() => alert('Link to help')} />  
    </DrawerContentScrollView>  
  );  
}
```

Options	การใช้งาน
<DrawerItemList {...props} />	ทำการแสดงเมนูตามรายชื่อ Screen
<DrawerItem />	เป็นการเพิ่ม Item ลงในแท็บเช่นเพิ่มปุ่ม หรือ ข้อความ
<DrawerContentScrollView {...props}>	ในสไลด์แท็บจะมีการใส่ ScrollView ในการแสดง Menu เพราะฉะนั้นควรใส่ Tag นี้ไว้เสมอ

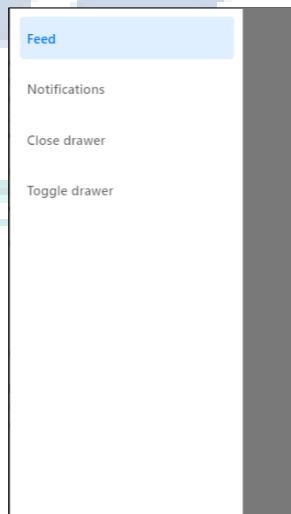
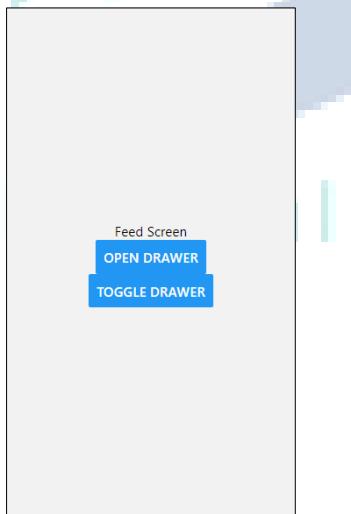
การควบคุม Drawer Navigator

ในการควบคุม Drawer Navigator จะแตกต่างจาก Navigation ตัวอื่นๆที่ไม่สามารถควบคุม Event ของตัว Navigation ได้มากนักแต่ Drawer Navigator สามารถควบคาระสไลด์ของตัวแท็บได้โดยใช้คำสั่งดังนี้

Options	การใช้งาน
navigation.openDrawer();	สไลด์แท็บเพื่อแสดงเมนู
navigation.closeDrawer();	สไลด์แท็บเพื่อปิดการแสดงเมนู
navigation.toggleDrawer();	สไลด์แท็บเพื่อแสดงเมนูหากเมนูปิดอยู่หรือสไลด์แท็บเพื่อปิดเมนู หากเมนูเปิดอยู่
navigation.jumpTo('Profile', { owner: 'Satya' });	เปลี่ยนหน้า Screen เป็นหน้าที่ต้องการ

ตัวอย่างการใช้งาน Drawer Navigator

ในตัวอย่างนี้เราจะสร้างการนำทางโดยใช้ตัวนำทางแบบ Drawer Navigation โดยเมื่อมีการเปิด App ขึ้นมาจะแสดงหน้า Feed(Feed Screen) เราสามารถกดปุ่มOpen Drawer เพื่อสไลด์เมนูด้านในและสามารถกดปุ่ม Toggle Drawer เพื่อแสดงเมนูหากถูกปิดอยู่และปิดเมนูหากเมนูถูกเปิดอยู่ตัวอย่าง User Interface แสดงด้านล่าง



my Thailand

ตัวอย่างโค้ดการใช้งานตัวนำทางแบบ Drawer Navigation

```
1. import * as React from 'react';
2. import { View, Text, Button } from 'react-native';
3. import { NavigationContainer } from '@react-navigation/native';
4. import {
5.   createDrawerNavigator,
6.   DrawerContentScrollView,
7.   DrawerItemList,
8.   DrawerItem,
9. } from '@react-navigation/drawer';
10.
11. function Feed({ navigation }) {
12.   return (
13.     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
14.       <Text>Feed Screen</Text>
15.       <Button title="Open drawer" onPress={() => navigation.openDrawer()} />
16.       <Button title="Toggle drawer" onPress={() => navigation.toggleDrawer()} />
17.     </View>
18.   );
19. }
20.
21. function Notifications() {
22.   return (
23.     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
24.       <Text>Notifications Screen</Text>
25.     </View>
26.   );
27. }
28.
29. function CustomDrawerContent(props) {
30.   return (
31.     <DrawerContentScrollView {...props}>
32.       <DrawerItemList {...props} />
33.       <DrawerItem
34.         label="Close drawer"
35.         onPress={() => props.navigation.closeDrawer()} />
36.       <DrawerItem
37.         label="Toggle drawer"
38.         onPress={() => props.navigation.toggleDrawer()} />
39.     </DrawerContentScrollView>
40.   );
41. }
42.
43. }
44.
45. const Drawer = createDrawerNavigator();
46.
47. function MyDrawer() {
48.   return (
49.     <Drawer.Navigator drawerContent={props => <CustomDrawerContent {...props} />}>
50.       <Drawer.Screen name="Feed" component={Feed} />
51.       <Drawer.Screen name="Notifications" component={Notifications} />
52.     </Drawer.Navigator>
53.   );
54. }
55.
56. export default function App() {
57.   return (
58.     <NavigationContainer>
59.       <MyDrawer />
```

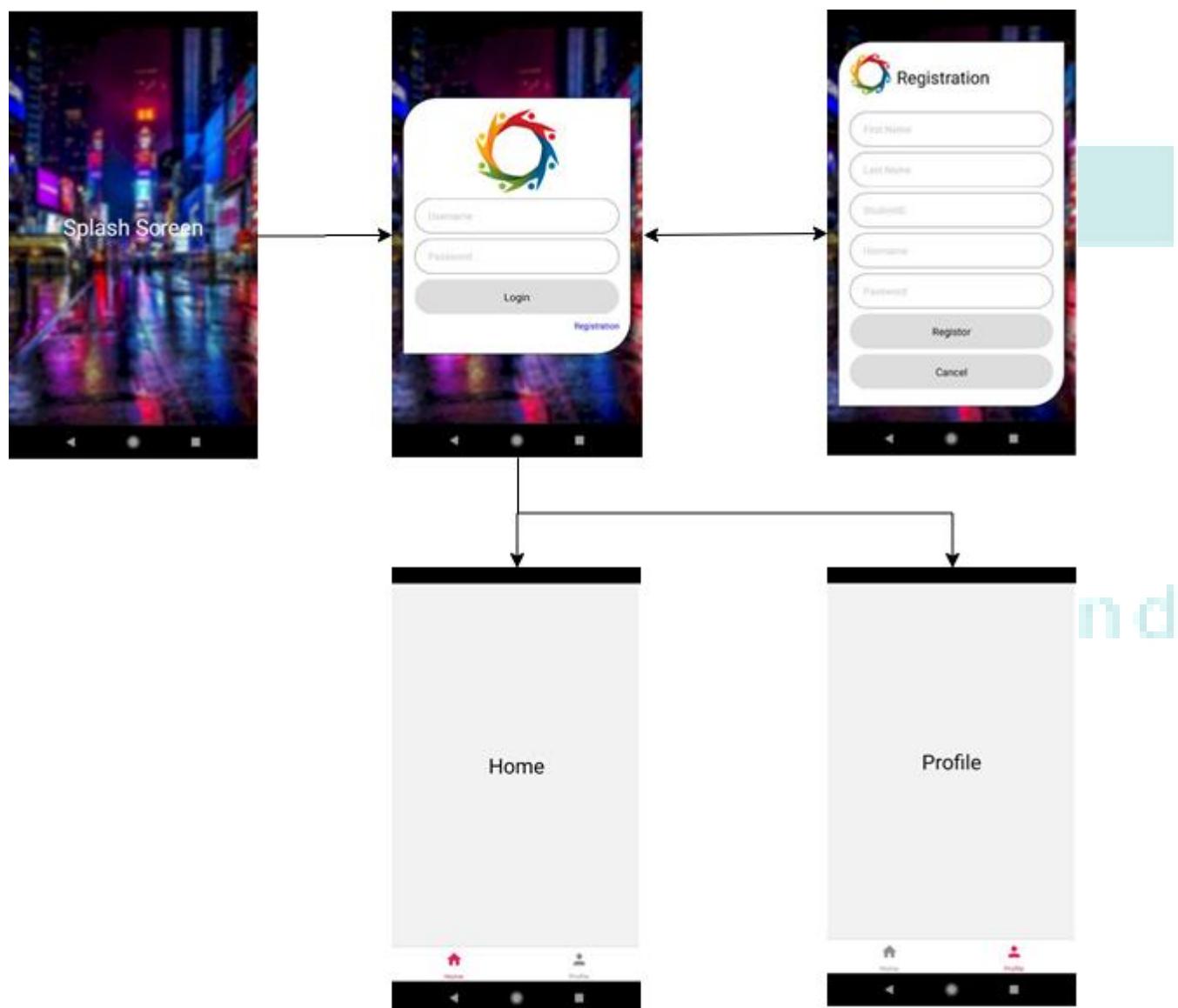
```
60.      </NavigationContainer>
61.    );
62.  }
```



Digital Academy Thailand

ตัวอย่างการใช้งาน Navigation

ตัวอย่างการแอพพลิเคชันนี้จะเป็นการผสมผสานการใช้งาน Stack Navigation และ Tab Bottom Navigation เพื่อความคุ้ม Screen ของแอพพลิเคชันจะมีทั้งหมด 5 หน้า เมื่อทำการเปิดแอพขึ้นมาจะแสดงหน้า Splash Screen เป็นเวลา 2.5 วินาที จากนั้นจะเปลี่ยนหน้ามายังหน้า Login Screen หากกดปุ่ม Login จะเปลี่ยนหน้าไปยังหน้า Home Screen และหากกด Registration จะเปลี่ยนหน้าไปยัง Registration Screen, ในหน้า Registration Screen หากกดปุ่ม Register หรือ Cancel จะย้อนกลับมายังหน้า Login Screen, ในหน้า Home Screen เราสามารถกด Menu Bottom Tab เพื่อเปลี่ยนหน้าไปยัง Profile Screen ได้ โดยตัวอย่างการเปลี่ยนหน้าแสดงดังแผนผังด้านล่าง



ตัวอย่างโค้ด

```
1. // App.js
2. import React, { Component } from 'react';
3. import { NavigationContainer } from '@react-navigation/native';
4. import { createStackNavigator } from '@react-navigation/stack';
5. import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
6. import { useNavigation } from '@react-navigation/native';
7. import { MaterialCommunityIcons } from '@expo/vector-icons';
8. import Splash from './Splash'
9. import Home from './Home'
10. import Profile from './Profile'
11. import Notifications from './Notifications'
12. import Login from './Login'
13. import Registration from './Registration'
14.
15. /*
16.
17.     Stack.Navigator
18.     SplashScreen
19.     login
20.     Tab.Navigator
21.         Home
22.         Profile
23.         Registration
24.
25. */
26.
27. const RegistrationScreen = ()=>{
28.     const navigation = useNavigation();
29.     return (
30.         <Registration navigation={navigation}>
31.     );
32. }
33.
34. const LoginScreen = ()=> {
35.     const navigation = useNavigation();
36.     return (
37.         <Login navigation={navigation}>
38.     );
39. }
40.
41. const HomeScreen = ()=> {
42.     const navigation = useNavigation();
43.     return (
44.         <Home navigation={navigation}>
45.     );
46. }
47.
48. const ProfileScreen=()=> {
49.     const navigation = useNavigation();
50.     return (
51.         <Profile navigation={navigation}>
52.     );
53. }
54.
55. const NotificationsScreen=()=> {
56.     const navigation = useNavigation();
57.     return (
58.         <Notifications navigation={navigation}>
59.     );
}
```

```

60.      }
61.
62.      const SplashScreen=()=> {
63.          const navigation = useNavigation();
64.          return (
65.              <Splash navigation={navigation}/>
66.          );
67.      }
68.
69.      const Tab = createBottomTabNavigator();
70.      const TabScreen=()=> {
71.          return (
72.              <Tab.Navigator
73.                  initialRouteName="Feed"
74.                  tabBarOptions={{
75.                      activeTintColor: '#e91e63',
76.                  }}>
77.
78.                  <Tab.Screen
79.                      name="Home"
80.                      component={HomeScreen}
81.                      options={{
82.                          tabBarLabel: 'Home',
83.                          tabBarIcon: ({ color, size }) => (
84.                              <MaterialCommunityIcons name="home" color={color} size={size} />
85.                          ),
86.                      })
87.                  />
88.                  <Tab.Screen
89.                      name="Profile"
90.                      component={ProfileScreen}
91.                      options={{
92.                          tabBarLabel: 'Profile',
93.                          tabBarIcon: ({ color, size }) => (
94.                              <MaterialCommunityIcons name="account" color={color} size={size} />
95.                          ),
96.                      })
97.                  />
98.              </Tab.Navigator>
99.          );
100.     }
101.
102.     const Stack = createStackNavigator();
103.     function MyStack() {
104.         return (
105.             <Stack.Navigator>
106.                 <Stack.Screen name="Splash"
107.                     component={SplashScreen}
108.                     options={{ headerShown: false }}/>
109.
110.                 <Stack.Screen name="Registration"
111.                     component={RegistrationScreen}
112.                     options={{ headerShown: false }}/>
113.
114.                 <Stack.Screen name="Login"
115.                     component={LoginScreen}
116.                     options={{ headerShown: false }}/>
117.
118.                 <Stack.Screen name="Tab" component={TabScreen} />
119.             </Stack.Navigator>
120.         );
121.     }
122.

```

```
123.  export default class App extends Component {
124.    constructor(props) {
125.      super(props);
126.      this.state = {
127.
128.    };
129.
130.  }
131.  render(props) {
132.    return (
133.      <NavigationContainer>
134.        <MyStack />
135.      </NavigationContainer>
136.    );
137.  }
138.}
```



Digital Academy Thailand

```
1. //Home.js
2. import React, { Component } from 'react';
3. import {
4.   View,FlatList,TouchableOpacity,StyleSheet,Text,Alert,Image,Button
5. } from 'react-native';
6.
7. export default class Home extends Component {
8.   constructor(props) {
9.     super(props);
10.    this.state = {
11.
12.      };
13.
14.   }
15.   render(props) {
16.     const { navigation } = this.props;
17.     return (
18.       <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
19.         <Text style={{fontSize:30}}>Home</Text>
20.       </View>
21.     );
22.   }
23. }
```



Digital Academy Thailand

```

1. //Login.js
2. import React from "react";
3. import { View, StyleSheet, Image, TextInput, TouchableOpacity, Text, ImageBackground
} from "react-native";
4.
5. const Login = (props) => {
6.   return (
7.
8.     <ImageBackground
9.       style={styles.imageBackground}
10.      source={{uri:'https://sv1.picz.in.th/images/2020/07/28/EYFj0b.jpg'}}
11.      blurRadius={1}
12.
13.       <View style={styles.middle} >
14.         <Image
15.           style={styles.image}
16.           source={{uri:'https://sv1.picz.in.th/images/2020/07/28/Ez0iOl
 .png'}}}
17.         />
18.         <TextInput placeholder="Username" style={styles.textInput}/>
19.
20.         <TextInput placeholder="Password" style={styles.textInput}/>
21.
22.         <TouchableOpacity style={styles.buttonLogin} onPress={()=>{prop
 s.navigation.navigate('Tab');props.navigation.reset({index:0,routes:[{name:'Tab'}]
});}}>
23.           <Text style={{fontSize:15}}>Login</Text>
24.         </TouchableOpacity>
25.
26.           <TouchableOpacity style={styles.buttonRegister} onPress={()=>p
 rops.navigation.navigate('Registration')}>
27.             <Text style={styles.text}>Registration</Text>
28.           </TouchableOpacity>
29.         </View>
30.
31.       </ImageBackground>
32.     );
33.   }
34.
35. const styles = StyleSheet.create({
36.   middle: {
37.     backgroundColor: '#ffffff',
38.     borderWidth: 1,
39.     padding: 16,
40.     margin: 16,
41.     borderTopLeftRadius: 50,
42.     borderBottomRightRadius: 50,
43.   },
44.   image: {
45.     width: 120,
46.     height: 120,
47.     resizeMode: 'contain',
48.     alignSelf: 'center',
49.     marginBottom: 8
50.
51.   },
52.   imageBackground: {
53.     flex: 1,
54.     resizeMode: "cover",
55.     justifyContent: "center"
56.   },
57.   buttonLogin: {
58.     justifyContent: "center",
59.     alignItems: "center",

```

```
60.      backgroundColor: "#DDDDDD",
61.      borderRadius:25,
62.      height: 50,
63.      marginBottom:8
64.    },
65.    buttonRegister: {
66.      justifyContent:"center",
67.      alignItems: "flex-end",
68.    },
69.    textInput:{
70.      borderRadius:25,
71.      height: 50,
72.      borderColor: 'gray',
73.      borderWidth: 1,
74.      paddingStart:20,
75.      marginBottom:8
76.    },
77.    text:{ 
78.      fontSize:12,
79.      textDecorationLine:"underline",
80.      color:'blue',
81.      marginBottom:16
82.    }
83.  });
84.  );
85. export default Login;
```



Digital Academy Thailand

```
1. //Notifications
2. import React, { Component } from 'react';
3. import {
4.   View,FlatList,TouchableOpacity,StyleSheet,Text,Alert,Image,Button
5. } from 'react-native';
6.
7. export default class Notifications extends Component {
8.   constructor(props) {
9.     super(props);
10.    this.state = {
11.
12.      };
13.
14.    }
15.   render(props) {
16.     const { navigation } = this.props;
17.     return (
18.       <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }>
19.         <Text>Notifications</Text>
20.         <Button title="OK" onPress={()=>navigation.navigate('Feed')} />
21.       </View>
22.     );
23.   }
24. }
```



Digital Academy Thailand

```
1. //Profile.js
2. import React, { Component } from 'react';
3. import {
4.   View,FlatList,TouchableOpacity,StyleSheet,Text,Alert,Image,Button
5. } from 'react-native';
6.
7. export default class Profile extends Component {
8.   constructor(props) {
9.     super(props);
10.    this.state = {
11.      };
12.    }
13.   render(props) {
14.     const { navigation } = this.props;
15.     return (
16.       <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
17.         <Text style={{fontSize:30}}>Profile</Text>
18.       </View>
19.     );
20.   }
21. }
```



```

1. //Registration.js
2. import React, { Component } from 'react';
3. import {
4.   View,FlatList,TouchableOpacity,StyleSheet,Text,Alert,Image,Button, ImageBackground, TextInput
5. } from 'react-native';
6.
7. export default class Registration extends Component {
8.   constructor(props) {
9.     super(props);
10.    this.state = {
11.
12.      };
13.
14.    }
15.    render(props) {
16.      const { navigation } = this.props;
17.      return (
18.
19.        <ImageBackground
20.          style={styles.imageBackground}
21.          source={{uri:'https://sv1.picz.in.th/images/2020/07/28/EYFj0b.jpg'}}
22.          blurRadius={1}>
23.
24.          <View style={styles.middle} >
25.            <View style={{flexDirection:'row',marginBottom:16}}>
26.              <Image
27.                style={styles.image}
28.                source={{uri:'https://sv1.picz.in.th/images/2020/07/28/Ez0i
  Ol.png'}}>
29.              />
30.              <Text style={{fontSize:25,marginStart:8, alignSelf:"center"}}>
31.                Registration</Text>
32.              </View>
33.              <TextInput placeholder="First Name" style={styles.textInput}/>
34.              <TextInput placeholder="Last Name" style={styles.textInput}/>
35.              <TextInput placeholder="StudentID" style={styles.textInput}/>
36.              <TextInput placeholder="Username" style={styles.textInput}/>
37.              <TextInput placeholder="Password" style={styles.textInput}/>
38.
39.              <TouchableOpacity style={styles.buttonLogin} onPress={()=>this.
  props.navigation.navigate('Login')}>
40.                <Text style={{fontSize:15}}>Registor</Text>
41.              </TouchableOpacity>
42.
43.              <TouchableOpacity style={styles.buttonLogin} onPress={()=>this.
  props.navigation.navigate('Login')}>
44.                <Text style={{fontSize:15}}>Cancel</Text>
45.              </TouchableOpacity>
46.
47.            </View>
48.          </ImageBackground>
49.        );
50.
51.      }
52.    }
53.    const styles = StyleSheet.create({
54.      middle: {
55.        backgroundColor:'#ffffff',

```

```
60.      borderWidth: 1,
61.      padding: 16,
62.      margin: 16,
63.      borderTopLeftRadius: 50,
64.      borderBottomRightRadius: 50,
65.    },
66.    image: {
67.      width: 60,
68.      height: 60,
69.      resizeMode: 'contain',
70.      alignSelf: 'center',
71.      marginBottom: 8
72.
73.    },
74.    imageBackground: {
75.      flex: 1,
76.      resizeMode: "cover",
77.      justifyContent: "center"
78.    },
79.    buttonLogin: {
80.      justifyContent: "center",
81.      alignItems: "center",
82.      backgroundColor: "#DDDDDD",
83.      borderRadius: 25,
84.      height: 50,
85.      marginBottom: 8
86.    },
87.    textInput: {
88.      borderRadius: 25,
89.      height: 50,
90.      borderColor: 'gray',
91.      borderWidth: 1,
92.      paddingStart: 20,
93.      marginBottom: 8
94.    },
95.
96.
97.  });

```

Digital Academy Thailand

```
1. //Splash.js
2. import React, { Component } from 'react';
3. import {
4.   View,FlatList,TouchableOpacity,StyleSheet,Text,Alert,Image,Button,ImageBackground
5. } from 'react-native';
6.
7. export default class Splash extends Component {
8.   constructor(props){
9.     super(props);
10.    this.state = {
11.
12.      };
13.    }
14.
15.   componentDidMount() {
16.     setTimeout(() => {
17.       this.props.navigation.navigate('Login');
18.       this.props.navigation.reset({index:0,routes:[{name:'Login'}]} );
19.     }, 2500)
20.
21.   }
22.
23.   render(props) {
24.     const { navigation } = this.props;
25.     return (
26.       <ImageBackground
27.         style={styles.imageBackground}
28.         source={{uri:'https://sv1.picz.in.th/images/2020/07/28/EYFj0b.jpg'}}
29.         blurRadius={1}>
30.         <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
31.           <Text style={{color:"white", fontSize:32}}>Splash Screen</Text>
32.         </View>
33.       </ImageBackground>
34.     );
35.   }
36. }
37.
38. const styles = StyleSheet.create({
39.   imageBackground: {
40.     flex: 1,
41.     resizeMode: "cover",
42.     justifyContent: "center"
43.   }
44. });


```

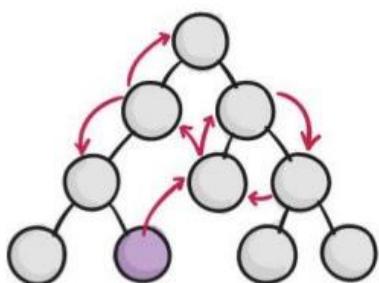
บทที่ 7 การจัดการ State ด้วย Redux

เหตุใดต้องมีการจัดการ State

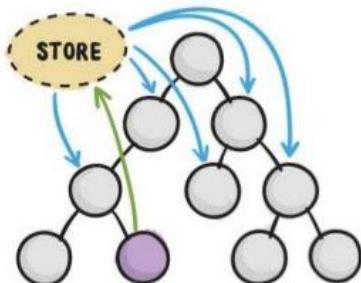
State คือตัวแปรแบบหนึ่งที่มีการเก็บข้อมูลในลักษณะ Object ที่ใช้ในการควบคุมการแสดงผล หากมีการเปลี่ยนแปลงค่าต่างๆภายใน State ก็จะมีการอัพเดตค่าเพื่อแสดงผลด้วย ในการเขียนแอพฯ ด้วย React-native มักจะมีการสร้างหลายๆ Component ซึ่งแต่ละ Component ก็จะมีการเก็บค่าใน State จำนวนมาก เช่นกัน ในบางครั้งหลาย Component จะมีการเปลี่ยนแปลงค่าภายใน State ซึ่ง Component อื่นๆอาจจะมีการเก็บค่าแบบเดียวกันทำให้จะต้องมีการเปลี่ยนแปลง State ภายใน Component อื่นๆด้วย เราลองจินตนาการดูถ้าเรามี Class Component ทั้งหมด 10 Components และแต่ละ Component จะมีการเก็บค่าภายใน State เหมือนกัน หากเราทำการเปลี่ยนค่าใน State ใน Component หนึ่งและต้องการทำให้แอพฯเป็นปัจจุบันเราก็จะต้องทำการไล่แก้ทุก State ในทุกๆ Component ซึ่งเป็นเรื่องที่ยากมากหากไม่มีการจัดการที่ดี

ตัวอย่างในแอพฯ จริงที่มีการเรียกใช้งาน State ข้ามอนกัน ในแอพฯ ส่วนใหญ่มักจะให้ผู้ใช้งานสมัครสมาชิกก่อนใช้งาน ซึ่งผู้ใช้งานจะสามารถดูประวัติข้อมูลและแก้ไขข้อมูลส่วนตัวได้ โดยหน้า(Screen) ที่ทำหน้าที่แก้ไขข้อมูลกับหน้าที่แสดงข้อมูล มักจะเป็นคนละหน้ากันซึ่งแต่ละหน้าเองก็จะมีการเก็บ State ที่บรรจุข้อมูลส่วนบุคคลเหมือนกัน ถ้าหากมีการแก้ไขข้อมูล เปลี่ยนแปลงค่า State ในหน้าแก้ไขข้อมูลแรกจะต้องมีการแก้ไขข้อมูลในหน้าแสดงข้อมูลเช่นเดียวกัน

ด้วยปัญหานี้จึงได้มีการคิดค้นวิธีการจัดการ State (State management) ให้ง่ายต่อการใช้งานโดยจะทำการรวม State ที่มีการใช้งานเข้าช้อนกันไว้ในที่เดียวกันและให้เรียกใช้งานหรือแก้ไขจากที่นี่ที่เดียวเท่านั้น ตัวอย่างจากภาพด้านข้างมีอีกลักษณะหนึ่งที่มีการสร้าง State ภายใน Component โครง Component มันซึ่งยุ่งยากต่อการใช้แก้ไขค่าที่เข้าช้อนกันกับภาพด้านขวา อีกหนึ่งอย่างที่มีการบริหารจัดการ State โดยมีการสร้างที่เก็บ State(Store) ไว้ที่เดียวกันหากมีการแก้ไขจาก Component ใด Component หนึ่ง Component อื่นๆ ก็สามารถอัพเดตค่าได้โดยง่าย



ไม่มีการทำ State management



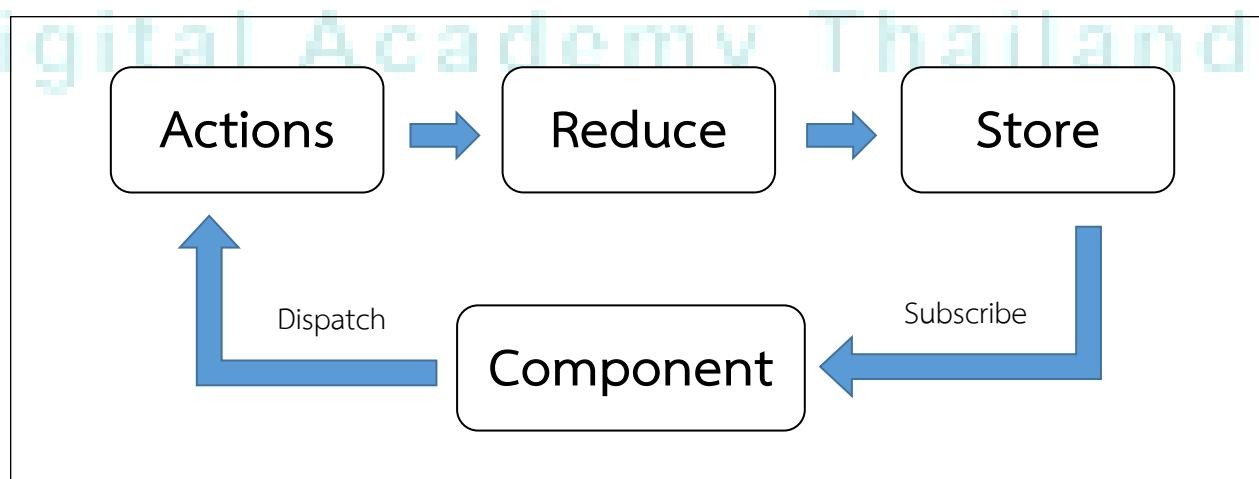
มีการทำ State management

Redux คืออะไร

Redux เป็น JavaScript Library แบบโօเพนซอร์สสำหรับจัดการ State โดยเฉพาะ ซึ่งได้รับความนิยมใช้งานอย่างแพร่หลายยกตัวอย่าง framework ที่มีการใช้งาน Redux เช่น React.js, Angular, Swift และ React-Native ซึ่ง framework เหล่านี้มักเป็น Font-end framework ที่ใช้ในการสร้าง User Interface ในแอพฯต่างๆ ซึ่ง Redux ถูกพัฒนาโดย Dan Abramov และ Andrew Clark ในปี ค.ศ. 2015 และได้รับความนิยมเรื่อยมาจนลึกลงไปจนถึง React-Native โดยการจัดการ State ด้วย Redux นี้จะมีหลักการสำคัญอยู่ 3 ข้อได้แก่

1. Single source of truth คือ การรวมและเก็บข้อมูล State ที่เข้าช้อนกันเอาไว้ที่เดียว โดยที่ไม่จำเป็นต้อง State ไว้ที่ Component ใด Component ซึ่งแต่ละ Component อาจจะมีการสร้าง State ที่ทำหน้าที่เหมือนกัน แต่เมื่อใช้ Redux เราจะต้องรวม State เหล่านี้เอาไว้ที่เดียวกันซึ่งเรียกว่า Store
2. State is read-only คือ State ที่เรารวมไว้เราจะสามารถอ่านได้อย่างเดียวเท่านั้น เราสามารถอ่าน State เหล่านี้ได้จากทุกๆ Component แต่เราจะไม่สามารถแก้ไข State ได้โดยตรงแต่เราจะใช้วิธีการสร้าง State ขึ้นมาใหม่แทนที่ State เดิมโดยใช้ Dispatcher ที่มีการกำหนด Action ต่างๆเอาไว้เท่านั้น
3. Changes are made with pure functions คือ พังก์ชันที่จะเปลี่ยน State จะต้องมีลักษณะเป็น pure functions กล่าวคือจะต้องสร้างพังก์ชันที่มีลักษณะไม่ขึ้นอยู่กับค่าตัวแปรหรือไม่เปลี่ยนแปลงค่าตัวแปร (Variables) ที่อยู่นอกสโคปของตัวมัน ในที่นี่เราจะทำการส่ง State ผ่านพังก์ชันนี้แล้วได้ State ใหม่อกมาโดยไม่มีการแก้ไข State เดิม ซึ่งเราจะต้องสร้างพังก์ชันนี้ใหม่ Action ต่างๆรวมกันไว้ที่เดียวกัน เราจะเรียกพังก์ชันนี้ว่า Reducer

โดยภาพรวมการทำงานของ Redux จะประกอบด้วย 5 ส่วนหลักตามแผนผังด้านล่าง



1. Store คือ ที่เก็บ State โดย State จะถูกเก็บไว้ในที่เดียวกันและเข้าถึงได้ผ่าน Store เท่านั้น
2. Component คือ Custom Component ที่ทำการเรียกแสดงค่า State ที่เก็บไว้ใน Store โดยจะทำเรียกใช้งานผ่าน Props หรือทำการ Map StateToProps

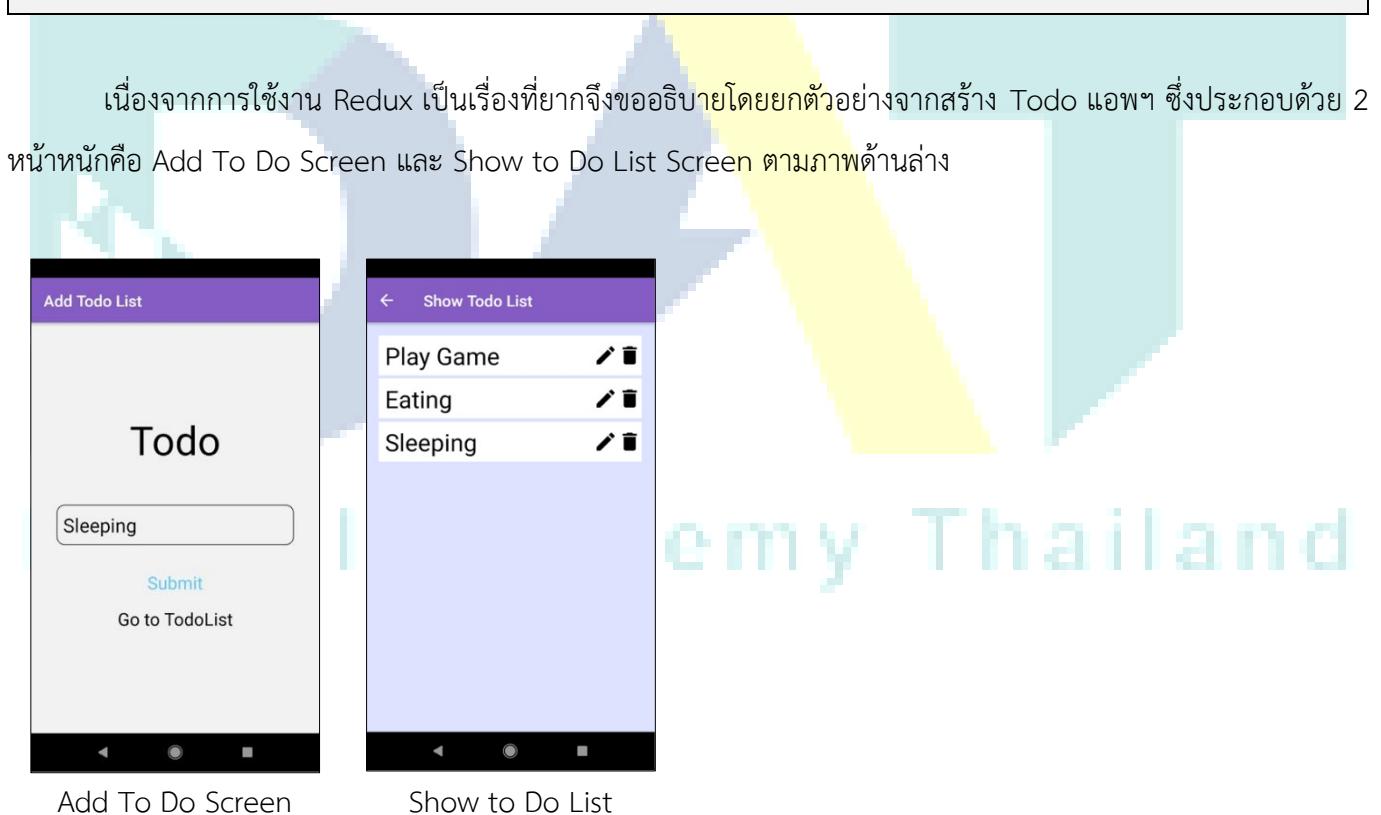
- Action คือ ตัวแปรประเภท object ที่ใช้ในการส่งค่า�ัง Reducer ซึ่งจะมีการระบุการกระทำต่อ State(Type) และค่าพารามิเตอร์ที่มีการแก้ไข
- Reducer คือ ฟังก์ชันที่ใช้สำหรับแก้ไขอัพเดตค่าต่างๆของ State

การติดตั้งไลบรารี Redux ใน React-native Project

ก่อนใช้งาน Redux ได้เราจะต้องทำการติดตั้งไลบรารีของ Redux ประกอบด้วยชุดไลบรารี 2 คือ redux core และ react-redux โดยใช้คำสั่งด้านล่าง

```
npm install redux
npm install react-redux
```

การจัดการ State(State management) ด้วย Redux



ตัวอย่างเราทำการเขียนชุดคำสั่งให้สามารถ Event ต่างๆได้ด้วยการสร้าง State ในหน้า To Do List Screen ประกอบด้วย todo เอาไว้ใช้ในการรับข้อมูลจาก TextInput และ todoList ที่เป็นอาร์เรย์เอาไว้สำหรับเก็บข้อมูลอีเวนต์ที่ต้องการจะทำเมื่อมีการกดปุ่ม Submit

```
this.state = {
  todo: null,
  todoList: []
}
```

เมื่อเราทำการกดเปลี่ยนหน้าไปยังหน้า Show List Screen เราจะทำการส่ง todoList, Functions deleteTodo และ editTodo ผ่าน Navigation(Route Hook Navigation)

```
deleteTodo=(key)=>{
  this.setState({todoList:[...this.state.todoList.filter((item)=>item.key!==key)]})
}
```

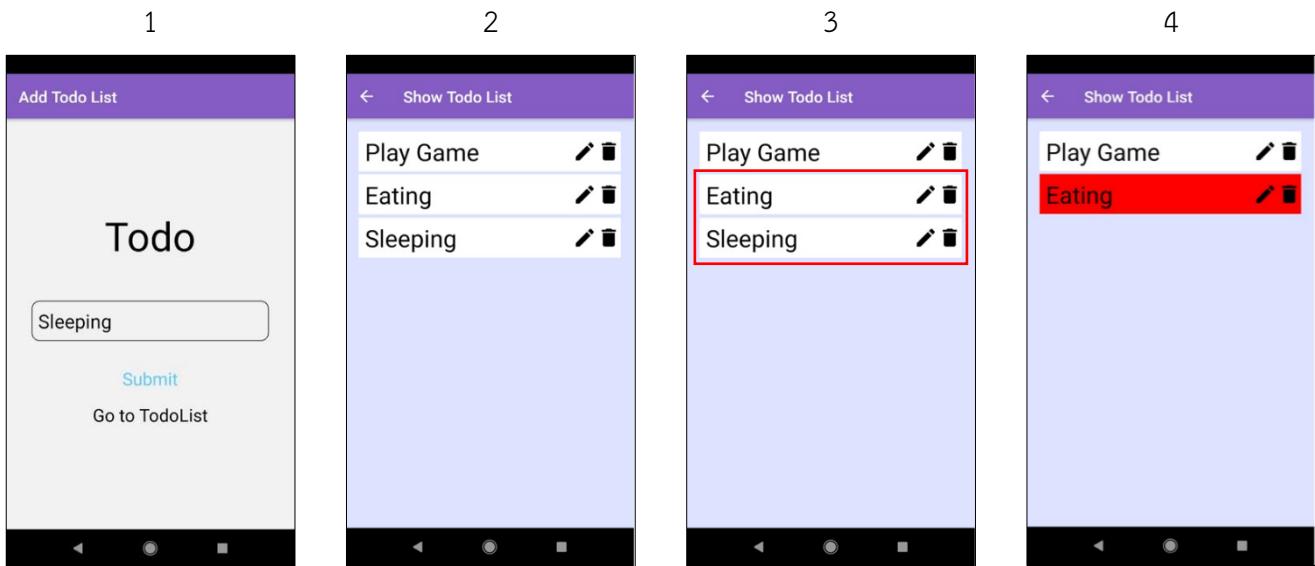
```
editTodo=(key)=>{
  let newTodoList = this.state.todoList.map(todo=>(todo.key==key)?{...todo,selected:d:!todo.selected}:todo);
  this.setState({todoList:newTodoList});
}
```

เมื่ออูปในหน้า ShowTodoList จะทำการแสดงข้อมูลอีเว้นท์ที่มีการสร้างไว้และหากมีการกดปุ่ม Edit จะทำการเรียก function editTodoList หากมีการกดปุ่มลบก็จะทำการเรียก deleteTodoList ตามคำสั่งด้านล่าง

```
renderItem={({item})=>{
  console.log(item);
  var newStyle;
  if(item.selected){
    newStyle = {alignItems:"center",backgroundColor:"red", flexDirection:"row", height:50};
  }else{
    newStyle = {alignItems:"center",backgroundColor:"white", flexDirection:"row", height:50};
  }
  return(
    <View style={newStyle}>
      <Text style={{fontSize:30,marginStart:8}}>{item.name}</Text>
      <View style={{position:'absolute', right:0,flexDirection:"row"} }>
        <TouchableOpacity onPress={()=>this.route.params.editTodoList(item.key)}>
          <MaterialIcons name="edit" size={30} color="black" />
        </TouchableOpacity>
        <TouchableOpacity onPress={()=>this.route.params.deleteTodoList(item.key)}>
          <MaterialIcons name="delete" size={30} color="black" />
        </TouchableOpacity>
      </View>
    </View>
  );
}
```

หากเราทำการทดสอบแอพฯตามตัวอย่างเหตุการณ์ด้านล่าง

1. เราทำการเพิ่มกิจกรรมที่เราต้องการทำตัวอย่าง Play Game, Eating และ Sleeping
2. ทำการกดปุ่ม Go To TodoList ไปยังหน้าแสดงรายการทั้งหมด
3. ทำการลบ Sleeping และแก้ไข Eating สังเกตรายการยังคงเหมือนเดิมไม่มีการเปลี่ยนแปลง
4. ทำการกลับไปยังหน้า Add Todo List และกลับมายังหน้า Show Todo List อีกครั้งผลลัพธ์แสดงด้านล่าง



สังเกตุผลลัพธ์แอพฯ ที่เราใช้งานจะไม่สามารถทำงานได้อย่างที่เราคาดหวังไว้กล่าวคือเมื่อมีการกดลบหรือแก้ไข แอพฯ ควรจะมีการเปลี่ยนแปลงทันทีทั้งนี้ โดยปัญหานี้เกิดจากเราได้ทำการแก้ไข todoList ซึ่งอยู่ใน Add Todo Component แต่มีการเรียกใช้งานใน Show Todo Component ทำให้เกิดการไม่อัพเดตใน Component นี้เมื่อมีการแก้ไขจึงต้องใช้วิธีการเปิดกลับไปกลับมาเพื่อให้ค่าที่ถูกแก้ไขถูกส่งมาใหม่ ด้วยปัญหานี้เราจึงทำการแก้ไขใหม่โดยใช้ Redux แทนที่วิธีการเดิมซึ่งจะต้องมีลำดับในการสร้างดังนี้

1. แบ่งกลุ่ม State
2. กำหนด Action Type
3. สร้าง Reducer
4. สร้าง Store
5. เรียกใช้งาน

1. แบ่งกลุ่ม State ในขั้นตอนนี้จะทำการแบ่งกลุ่ม State แยกเป็นสองกลุ่มคือ State ที่มีการใช้งานร่วมระหว่าง Component กับ State ที่นำไปใช้ของใครของมันในแต่ละ Component ทั้งนี้เราจะสนใจเฉพาะ State ที่มีการใช้งานร่วมกันระหว่าง Component ในตัวอย่างเราทำการสร้าง Initial State ของ todoList

```
const initialState = {
  todoList: []
}
```

2. กำหนด Action Type ในขั้นตอนนี้จะต้องกำหนด Action ต่างๆ ที่จะกระทำการต่อ State ซึ่งตัวอย่าง Todo แอพฯ นี้เราสามารถกระทำการต่างๆ ได้ดังนี้
 - a. เพิ่ม เราจะใช้ ADD_TODO เป็น Type โดยจะต้องมีการส่งค่า object todo={todo:Your_Event, key:Your_unique_keys} มาบังฟังก์ชันเพื่อที่จะสร้าง Object สำหรับ Reducer

- b. ลบ เราจะใช้ DELETE_TODO เป็น Type โดยหากต้องการจะลบเราจะต้องมีการส่ง Key มาyangฟังก์ชันเพื่อที่จะสร้าง Object สำหรับ Reducer
- c. แก้ไข เราจะใช้ Edit_TODO เป็น Type เราจะต้องทำการส่ง object todo={ todo:Your_Event, key:Your_unique_keys , selected:Your_selected_status} มาyangฟังก์ชันเพื่อที่จะสร้าง Object สำหรับ Reducer

```
//actions/types.js
export const ADD_TODO = "ADD_TODO";
export const DELETE_TODO = "DELETE_TODO";
export const EDIT_TODO = "EDIT_TODO";
```

```
//actions/todo.js
import {ADD_TODO, DELETE_TODO, EDIT_TODO} from './types'

export const addTodo=(todo)=>({
    type: ADD_TODO,
    data: todo,
    selected: false
});
export const deleteTodo=(key)=>({
    type: DELETE_TODO,
    key:key
});
export const editTodo=(item)=>({
    type: EDIT_TODO,
    item: item
});
```

3. สร้าง Reducer ในขั้นตอนนี้เราจะต้องสร้างฟังก์ชันใหม่มีพคติกรรมตาม Action ที่เราได้กำหนดไว้ ในตัวอย่างของเราสร้าง Action ไว้ 3 แบบในฟังก์ชันเดียว

```
const todoReducer = (state = intialState,action) =>{
  switch(action.type){
    case ADD_TODO:
      return{
        ...state,
        todoList:state.todoList.concat({
          key:Math.random(),
          name:action.data,
          selected:action.selected
        })
      };
    case DELETE_TODO:
      return{
        ...state,
        todoList:state.todoList.filter((item)=>item.key !== action.key)
      };
    case EDIT_TODO:
      console.log(state.todoList)
      return
{...state,todoList:state.todoList.map(todo=>(todo.key==action.item.key)?{...todo,selected:!todo.selected}:todo),test:true}
      default:
        return state;
  }
}
```

4. สร้าง Store ในขั้นตอนนี้เราจะสร้างที่เก็บรวม Reducer เอาไว้ใน Store ทั้งนี้เนื่องจากเราสามารถเข้าถึง Store ได้ในทุกๆ Component ในกรณีที่มี Reducer หลายตัวจะต้องถูกรวบไว้ใน Store เดียวเท่านั้น

```
const rootReducer = combineReducers({
  todoReducer: todoReducer
})
const configureStore = createStore(rootReducer);
```

5. การเรียกใช้งาน ในขั้นตอนนี้เราจะทำการเรียกใช้งาน โดยสร้าง Provider ซึ่งเป็น Parent Component ครอบ Children Component เพื่อที่จะให้ Children Component ทั้งหมดเข้าถึงได้ Store ได้

```
export default function App() {
  const store = configureStore;
  return (
    <Provider store={configureStore}>
      <NavigationContainer>
        <MyStack />
      </NavigationContainer>
    </Provider>
  );
}
```

จากเราจะต้องทำการเชื่อมต่อ ระหว่าง Store และ Component โดยใช้คำสั่ง Connect ใน Component โดยจะต้องมีการส่ง Argument ที่สำคัญ 2 อย่างคือ

mapStateToProps คือ พังก์ชันที่ใช้ในการติดตามค่าใน Store เมื่อถูกเปลี่ยนแปลงค่าจากคอมโพenenต์ ได้แก่ตามพังก์ชันนี้ที่จะทำการอัปเดตค่าเสมอ

mapDispatchToProps คือ พังก์ชันที่ใช้ในการส่ง Action ไปยัง Store เพื่อใช้ในการเปลี่ยนค่าต่างๆใน Store ตาม Action ที่เราได้กำหนดไว้

```
const mapStateToProps = (state) => {
  return {
    todos: state.todoReducer.todoList
  }
}
const mapDispatchToProps = (dispatch) => {
  return {
    add: (todo) => dispatch(addTodo(todo))
  }
}
export default connect(mapStateToProps, mapDispatchToProps)(TodoForm)
```

การเรียกใช้งานเรามารถเรียกผ่าน Props เมื่อันดับเช่น

```
this.props.todos
// ในการที่เราสร้าง mapDispatchToProps ให้
this.props.add(this.state.todo)

// ในกรณีที่ไม่ได้สร้าง mapDispatchToProps ให้
this.props.dispatch(editTodo(item))
```



Digital Academy Thailand

หากเราทำการทดสอบแอพฯตามตัวอย่างเหตุการณ์ด้านล่าง

1. เราทำการเพิ่มกิจกรรมที่เราต้องการทำตัวอย่าง Play Game, Eating และ Sleeping
2. ทำการกดปุ่ม Go To TodoList ไปยังหน้าแสดงรายการทั้งหมด
3. ทำการลบ Sleeping และแก้ไข Eating สีเกตผลลัพธ์แสดงด้านล่าง



Digital Academy Thailand

ตัวอย่างโค้ด

```
1. //App.js
2. import * as React from 'react';
3. import { Button, View } from 'react-native';
4. import { NavigationContainer } from '@react-navigation/native';
5. import { createStackNavigator } from '@react-navigation/stack';
6. import { useNavigation } from '@react-navigation/native';
7. import { useRoute } from '@react-navigation/native';
8. import TodoForm from './TodoForm'
9. import TodoList from './TodoList'
10. import configureStore from './Store'
11. import { Provider } from 'react-redux'
12.
13. function TodoFormScreen() {
14.   const navigation = useNavigation();
15.
16.   return (
17.     <TodoForm navigation={navigation}/>
18.   );
19. }
20.
21. function TodoListScreen() {
22.   const navigation = useNavigation();
23.   const route = useRoute();
24.   return (
25.     <TodoList navigation={navigation} route={route}/>
26.   );
27. }
28.
29. const Stack = createStackNavigator();
30.
31. const navigationAdderOptions={
32.   title: 'Add Todo List',
33.   headerTintColor: 'white',
34.   headerStyle: {
35.     backgroundColor: '#845cc3',
36.   },
37. };
38.
39. const navigationListOptions={
40.   title: 'Show Todo List',
41.   headerTintColor: 'white',
42.   headerStyle: {
43.     backgroundColor: '#845cc3',
44.   },
45. };
46.
47. function MyStack() {
48.   return (
49.     <Stack.Navigator>
50.       <Stack.Screen options={navigationAdderOptions} name="Form" component={TodoFormScreen} />
51.       <Stack.Screen options={navigationListOptions} name="List" component={TodoListScreen} />
52.     </Stack.Navigator>
53.   );
54. }
55.
56. export default function App() {
57.   const store = configureStore;
58.   return (
59. 
```

```
60.      <Provider store={configureStore}>
61.        <NavigationContainer>
62.          <MyStack />
63.        </NavigationContainer>
64.      </Provider>
65.    );
66. }
```



Digital Academy Thailand

```

1. //TodoForm.js
2. import React, {Component} from 'react';
3. import { Text, View, StyleSheet, TextInput, TouchableOpacity } from 'react-native';
4. import Constants from 'expo-constants';
5. import { addTodo } from './actions/todo';
6. import { connect } from 'react-redux';
7.
8. class TodoForm extends Component {
9.     constructor(props) {
10.         super(props);
11.         this.state = {
12.             todo: null
13.         }
14.         const {dispatch} = props;
15.         this.dispatch = dispatch;
16.     }
17.
18.     render() {
19.         return (
20.             <View style={styles.container}>
21.                 <Text style={styles.title}>Todo</Text>
22.                 <TextInput
23.                     value={this.state.todo}
24.                     placeholder='Event'
25.                     style={styles.input}
26.                     onChangeText={(todo) => this.setState({ todo })} />
27.                 <TouchableOpacity
28.                     style={{ marginBottom: 16 }}
29.                     onPress={() => {
30.                         this.props.add(this.state.todo)
31.                         this.setState({ todo: null })
32.                     }}>
33.                         <Text style={{ fontSize: 22, color: '#5fc9f8' }}>Submit</Text>
34.                     </TouchableOpacity>
35.                     <TouchableOpacity
36.                         style={{ marginBottom: 16 }}
37.                         onPress={() =>
38.                             this.props.navigation.navigate('List')}>
39.                             <Text style={{ fontSize: 22 }}>Go to TodoList</Text>
40.                         </TouchableOpacity>
41.                     </View>
42.                 );
43.             }
44.         }
45.     }
46.
47.     const styles = StyleSheet.create({
48.         container: {
49.             flex: 1,
50.             margin: 16,
51.             alignItems: 'center',
52.             justifyContent: 'center'
53.         },
54.         title: {
55.             fontSize: 50,
56.             marginBottom: 48
57.         },
58.         input: {
59.             fontSize: 24,
60.             marginBottom: 32,
61.             borderWidth: 1,
62.             padding: 8,
63.             width: '90%',
```

```
64.     borderRadius: 10,
65.   }
66. });
67.
68. const mapStateToProps = (state) => {
69.   return {
70.     todos: state.todoReducer.todoList
71.   }
72. }
73.
74. const mapDispatchToProps = (dispatch) => {
75.   return {
76.     add: (todo) => dispatch(addTodo(todo))
77.   }
78. }
79.
80. export default connect(mapStateToProps, mapDispatchToProps)(TodoForm)
```



Digital Academy Thailand

```

1. //TodoList.js
2. import React, {Component} from 'react';
3. import { Text, View, StyleSheet, TextInput, TouchableOpacity, FlatList } from 'react-native';
4. import Constants from 'expo-constants';
5. import { MaterialIcons } from '@expo/vector-icons';
6. import { deleteTodo, editTodo } from './actions/todo';
7. import { connect } from 'react-redux';
8.
9. class TodoList extends Component {
10.
11.   constructor(props) {
12.     super(props);
13.   }
14.
15.   renderItem={({item})=>{
16.     console.log(item);
17.     var newStyle;
18.     if(item.selected){
19.       newStyle = {alignItems:"center",backgroundColor:"red", flexDirection:"row", height:50};
20.     }else{
21.       newStyle = {alignItems:"center",backgroundColor:"green", flexDirection:"row", height:50};
22.     }
23.     return(
24.       <View style={newStyle}>
25.         <Text style={{fontSize:30,marginStart:8}}>{item.name}</Text>
26.         <View style={{position:'absolute', right:0,flexDirection:"row"} }>
27.           <TouchableOpacity onPress={()=>this.props.dispatch(editTodo(item))}>
28.             <MaterialIcons name="edit" size={30} color="black" />
29.           </TouchableOpacity>
30.           <TouchableOpacity onPress={()=>this.props.dispatch(deleteTodo(item.key))}>
31.             <MaterialIcons name="delete" size={30} color="black" />
32.           </TouchableOpacity>
33.         </View>
34.       </View>
35.     );
36.   }
37.
38.   renderSeparator=()=>{
39.     return(
40.       <View style={{height:4}} />
41.     );
42.   }
43.
44.   render(){
45.     return (
46.       <FlatList style={styles.listContainer}
47.         data={this.props.todos}
48.         renderItem={this.renderItem}
49.         keyExtractor={(item, index) => item.key.toString()}
50.         ItemSeparatorComponent={this.renderSeparator}
51.       />
52.     );
53.   }
54.
55. }
56.
57. const styles = StyleSheet.create({
58.   listContainer: {
59.     backgroundColor: '#dce2ff',

```

```
60.      padding: 16
61.    },
62.  });
63.
64. const mapStateToProps = (state) => {
65.   return {
66.     todos: state.todoReducer.todoList
67.   }
68. }
69.
70. export default connect(mapStateToProps)(TodoList);
```



Digital Academy Thailand

```
1. //reducers/todoReducer.js
2. import {ADD_TODO, DELETE_TODO, EDIT_TODO} from '../actions/types';
3. import { AccessibilityInfo } from 'react-native';
4.
5. const intialState ={
6.   todoList:[]
7. }
8.
9. const todoReducer = (state = intialState,action) =>{
10.   switch(action.type){
11.     case ADD_TODO:
12.       return{
13.         ...state,
14.         todoList:state.todoList.concat({
15.           key:Math.random(),
16.           name:action.data,
17.           selected:action.selected
18.         })
19.       };
20.     case DELETE_TODO:
21.       return{
22.         ...state,
23.         todoList:state.todoList.filter((item)=>item.key !== action.key)
24.       };
25.     case EDIT_TODO:
26.       console.log(state.todoList)
27.       return {...state,todoList:state.todoList.map(todo=>(todo.key===action.item.key)?{...todo,selected:!todo.selected}:todo),test:true}
28.     default:
29.       return state;
30.   }
31. }
32.
33. export default todoReducer;
```

Digital Academy Thailand

```
1. //actions/todo.js
2. import {ADD_TODO, DELETE_TODO, EDIT_TODO} from './types'
3.
4. export const addTodo=(todo)=>(
5.     {
6.         type: ADD_TODO,
7.         data: todo,
8.         selected: false
9.     }
10. );
11.
12. export const deleteTodo=(key)=>(
13.     {
14.         type: DELETE_TODO,
15.         key: key
16.     }
17. );
18.
19. export const editTodo=(item)=>(
20.     {
21.         type: EDIT_TODO,
22.         item: item
23.     }
24. );
```



Digital Academy Thailand

```
1. //actions/types.js
2. export const ADD_TODO = "ADD_TODO";
3. export const DELETE_TODO = "DELETE_TODO";
4. export const EDIT_TODO = "EDIT_TODO";
```



Digital Academy Thailand

```

1. //TodoList.js
2. import React, {Component} from 'react';
3. import { Text, View, StyleSheet, TextInput, TouchableOpacity, FlatList } from 'react-native';
4. import Constants from 'expo-constants';
5. import { MaterialIcons } from '@expo/vector-icons';
6. import { deleteTodo, editTodo } from './actions/todo';
7. import { connect } from 'react-redux';
8.
9. class TodoList extends Component {
10.
11.   constructor(props) {
12.     super(props);
13.   }
14.
15.   renderItem={({item})=>{
16.     console.log(item);
17.     var newStyle;
18.     if(item.selected){
19.       newStyle = {alignItems:"center",backgroundColor:"red", flexDirection:"row", height:50};
20.     }else{
21.       newStyle = {alignItems:"center",backgroundColor:"green", flexDirection:"row", height:50};
22.     }
23.     return(
24.       <View style={newStyle}>
25.         <Text style={{fontSize:30,marginStart:8}}>{item.name}</Text>
26.         <View style={{position:'absolute', right:0,flexDirection:"row"} }>
27.           <TouchableOpacity onPress={()=>this.props.dispatch(editTodo(item))}>
28.             <MaterialIcons name="edit" size={30} color="black" />
29.           </TouchableOpacity>
30.           <TouchableOpacity onPress={()=>this.props.dispatch(deleteTodo(item.key))}>
31.             <MaterialIcons name="delete" size={30} color="black" />
32.           </TouchableOpacity>
33.         </View>
34.       </View>
35.     );
36.   }
37.
38.   renderSeparator=()=>{
39.     return(
40.       <View style={{height:4}} />
41.     );
42.   }
43.
44.   render(){
45.     return (
46.       <FlatList style={styles.listContainer}
47.         data={this.props.todos}
48.         renderItem={this.renderItem}
49.         keyExtractor={(item, index) => item.key.toString()}
50.         ItemSeparatorComponent={this.renderSeparator}
51.       />
52.     );
53.   }
54.
55. }
56.
57. const styles = StyleSheet.create({
58.   listContainer: {
59.     backgroundColor: '#dce2ff',

```

```
60.      padding: 16
61.    },
62.  });
63.
64. const mapStateToProps = (state) => {
65.   return {
66.     todos: state.todoReducer.todoList
67.   }
68. }
69.
70. export default connect(mapStateToProps)(TodoList);
```



Digital Academy Thailand

บทที่ 8 การใช้บริการ Google Cloud Firestore

Firebase คืออะไร

Firebase คือ Platform ที่รวมเครื่องมือต่าง ๆ สำหรับการจัดการในส่วนของ Back-end หรือฝั่ง Server ซึ่งทำให้สามารถสร้างแอพฯ ได้อย่างมีประสิทธิภาพ ลดเวลาและค่าใช้จ่ายของการทำ Back-end service เอง อีกทั้งยังมีเครื่องมือที่ช่วยวิเคราะห์ข้อมูลการใช้งาน โดยมีทั้งเครื่องมือที่ฟรีและมีค่าใช้จ่าย Firebase นี้หมายความกับคนที่ต้องการสร้างแอพฯ ทั่วไปโดยเฉพาะ Mobile App และมีความต้องการที่จะเชื่อมกับ Back-end ต่างๆ แต่มีข้อจำกัดทางด้านค่าใช้จ่ายไม่ว่าจะเป็นค่าเครื่อง Server ค่าการติดตั้งโปรแกรมย่อยต่างๆ หรือแม้กระทั้งการตั้งค่าการเขื่อมต่อต่าง ๆ ความปลอดภัยอีกลักษณะเป็นอุปสรรค เช่น กัน ซึ่ง Firebase ได้จัดการสิ่งเหล่านี้ให้เรียบร้อย อีกทั้งความสามารถอื่น ๆ อย่างภาษาที่ช่วยเพิ่มประสิทธิภาพ และคุณภาพให้กับงานโดยที่จะช่วยเหลือเราให้มีความสะดวกมากยิ่งขึ้น เช่น รายงานข้อมูลผลิตภัณฑ์ ข้อมูลสถิติผู้ใช้งานหรือแม้กระทั้งระบบที่ช่วยในการทดสอบแอพฯ ของเราเป็นต้น โดย Firebase มีบริการหลากหลายสามารถแบ่งได้เป็น 3 กลุ่มหลัก ดังนี้

- สร้างแอพฯให้ดีขึ้น(Build better apps) คือกลุ่มเครื่องมือนี้จะเป็น Back-end service ต่างๆที่พร้อมสำหรับเชื่อมต่อแอพฯของเราโดยจะมี 5 ผลิตภัณฑ์ได้แก่
 - Cloud Firestore คือบริการฐานข้อมูล NoSQL ใช้วิธีการเก็บข้อมูลแบบ Collection และยังสามารถเชื่อมต่อเข้าถึงข้อมูลได้แบบ Real-time รองรับการทำงานแบบ Offline และ Online ได้ทั้ง Android และ iOS
 - Authentication คือบริการตรวจสอบผู้ใช้ โดยสามารถตรวจสอบได้หลายวิธี เช่น Email/Password, เบอร์โทรศัพท์, บัญชี Google, Facebook, Twitter, Github เป็นต้น มีฐานข้อมูลเป็นของตัวเองไม่ต้องสร้างใหม่ มีระบบสนับสนุนการใช้งานสามารถดูวิธีการสมัคร วันที่สมัครและประวัติการใช้งานล่าสุด
 - Hosting คือบริการฝากไฟล์ Website เช่น HTML, CSS, JS, JPG เพื่อให้คนอื่นสามารถเข้าใช้งานเว็บไซต์ของเราได้ โดยที่ไม่ต้องมีทักษะในการฝากไฟล์ที่ได้จากการสร้างของ JavaScript Web Framework ต่าง ๆ เช่น Angular, React, Vue สังเกตว่าจะได้ไฟล์ HTML, CSS, JS ต่าง ๆ ตามที่ได้บอกไว้ข้างต้น หรือจะเป็นไฟล์ที่เขียนเองก็ได้ ไม่จำเป็นต้องใช้ Framework ก็ได้เหมือนกัน อีกทั้งมี CDN และ SSL มาด้วยแบบพรีเพื่อให้ผู้ใช้งานคุณได้รับประสบการณ์การใช้งานที่ปลอดภัยเชื่อถือได้และไม่มีความล่าช้าแม้ว่าจะอยู่ที่ไหนก็ตาม
 - Cloud Storage คือบริการเก็บไฟล์รูปภาพ, ไฟล์เสียง, วิดีโอ เพื่อใช้บนแอพฯ
 - ML Kit คือ Machine Learning SDK ที่ช่วยให้แอพฯของเราระบบสามารถใช้ความสามารถของ ML ได้ง่ายยิ่งขึ้น สามารถทำงานได้ทั้งแบบ Online และ Offline
 - ปรับปรุงคุณภาพแอพฯ(Improve app quality) คือกลุ่มเครื่องมือเสริมที่ช่วยวัดวิเคราะห์สรุปการทำงานของแอพฯเรา และแจ้งเตือนกลับมาถังนักพัฒนาหากพบข้อผิดพลาดโดยสามารถแบ่งได้ 3 ผลิตภัณฑ์ได้แก่

- 2.1. Crashlytics คือบริการตรวจสอบจับและแจ้งเตือนหากแอพฯ เรากีดอาการ Crash ขึ้นแบบ Realtime เพื่อให้แอพฯ เรา เสียรอยู่เสมอ โดยจะทำการแจ้งให้ทราบถึงข้อผิดพลาดและผลกระทบ ผ่านทาง E-mail และ Firebase Console เพื่อการแก้ปัญหาที่รวดเร็วและตรงจุด
- 2.2. Performance Monitoring คือบริการตรวจสอบคุณภาพของแอพฯ เพื่อให้แอพฯ ของเรารอดูสนองได้เร็วอยู่เสมอ โดยสามารถตรวจสอบเวลาและรายละเอียดการทำงานต่าง ๆ เช่น เวลาที่ใช้ในการเปิดแอพฯ, เวลาที่ใช้การเปลี่ยนหน้า UI, เวลาที่ใช้ในการโหลด API, ขนาดข้อมูลที่ Download/Upload, จำนวน API ที่สำเร็จหรือล้มเหลว เป็นต้น
- 2.3. Test Lab คือบริการทดสอบแอปฯ บนฮาร์ดแวร์จริง ๆ เพื่อให้มั่นใจว่าแอพฯ ของเรามีความสามารถให้แอพฯ ของเรารอดูสนองต่อความต้องการได้จริง ๆ โดยสามารถระบุรุ่นและเวอร์ชันที่ต้องการได้ แล้วระบุรูปแบบการทดสอบต่าง ๆ เพื่อทดสอบและรายงานผลกลับมา ไม่ต้องซื้อโทรศัพท์เอง
3. ขยายธุรกิจ(Grow your business) คือกลุ่มเครื่องมือที่ช่วยเพิ่มขีดความสามารถให้แอพฯ ของเรารอดูสนองต่อความต้องการของผู้ใช้งานได้มากยิ่งขึ้นโดยสามารถแบ่งได้ 8 ผลิตภัณฑ์ได้แก่
- 3.1. In-App Messaging คือบริการแสดงข้อความ pop-up ภายในแอพฯ ของเรา เช่น โฆษณา, การแจ้งเตือน, ข่าวสาร เป็นต้น
 - 3.2. Google Analytics คือบริการแสดงข้อมูลสถิติต่างๆ ของแอพฯ เช่น จำนวนระบบปฏิบัติการที่ติดตั้งแอพฯ ของเรา, จำนวนผู้ใช้งาน ณ ปัจจุบัน เป็นต้นเพื่อวิเคราะห์กลุ่มเป้าหมายและรับทราบพฤติกรรมของผู้ใช้งาน
 - 3.3. Predictions คือบริการวิเคราะห์ข้อมูลการใช้งานแอพฯ ช่วยให้เราทราบว่าผู้ใช้งาน ใช้งานส่วนใดบ้างในแอพฯ ช่วยให้เรารู้ว่าส่วนใดตอบสนองได้ดี ส่วนใดควรปรับปรุงและช่วยวิเคราะห์ข้อมูลพฤติกรรมที่จะเกิดขึ้นในอนาคต ของผู้ใช้งาน เพื่อช่วยให้เราวางแผนกลยุทธ์ทั้งปรับปรุงความสามารถของแอพฯ ให้ตอบสนองต่อความต้องการของผู้ใช้งานมากยิ่งขึ้น
 - 3.4. Cloud Messaging คือบริการส่งการแจ้งเตือนไปยังมือถือหรือเว็บของเรา เพื่อแจ้งข้อมูลไปยังผู้ใช้งานแม้ว่า จะปิดแอพฯ หรือไม่ก็ตาม
 - 3.5. Remote Config คือเครื่องมือที่ช่วยเพิ่มความสามารถที่จะเปลี่ยนลักษณะการทำงานหรือลักษณะที่ปรากฏของแอพฯ ของเราได้ทันทีโดยไม่ต้องปรับเวอร์ชันและรออนุมัติจาก Store เช่น การเปลี่ยนรูปแบบตามเทศกาล, เปลี่ยนภาษาตามผู้ใช้งาน เป็นต้น
 - 3.6. App Indexing คือการปรับแต่งแอพฯ ของเราให้แสดงผลข้อมูลภายในแอปฯ บน Google Search ได้ เช่น ค้นชื่อร้านอาหารแล้วปรากฏผลในขึ้นมาให้ดูรายละเอียดและรีวิว เป็นต้น
 - 3.7. A/B Testing (Beta) คือความสามารถในการแสดงผลแอฟฟลาร์ยูปแบบเพื่อทดสอบการแสดงผลหรือการทำงาน ว่าสิ่งไหนจะมีประสิทธิภาพมากกว่าที่เดิม ให้แก่ผู้ใช้งาน เช่น การวางแผนที่ผู้ใช้งานใช้สะดวก สมมุติว่ามีผู้ใช้งาน 100 คน อาจจะมี 50 คนได้ปุ่มที่อยู่มุมบน อีก 50 คนได้ปุ่มอยู่มุมล่าง หากว่ามีการใช้งานแบบหนึ่งมากกว่ากันก็อาจจะสรุปผลและเลือกใช้แบบนั้นกับทุกคนในท้ายที่สุด

โดยเนื้อหารainรายวิชานี้เราจะมุ่งเน้นใช้งานเครื่องมือในกลุ่ม สร้างแอพฯให้ดีขึ้น(Build better apps) ซึ่งเป็นกลุ่มเครื่องมือพื้นฐานที่แอพฯส่วนใหญ่

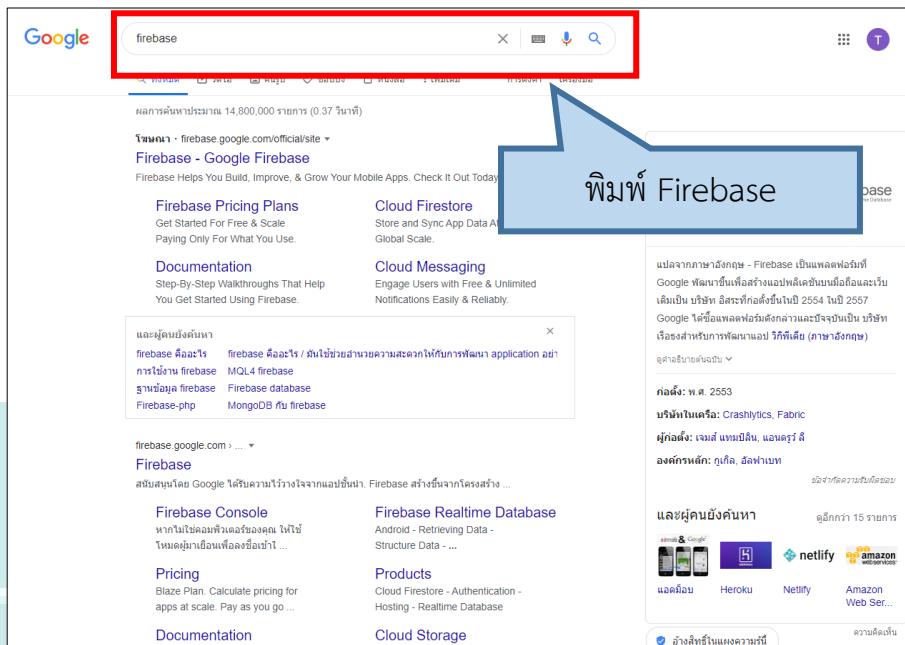


Digital Academy Thailand

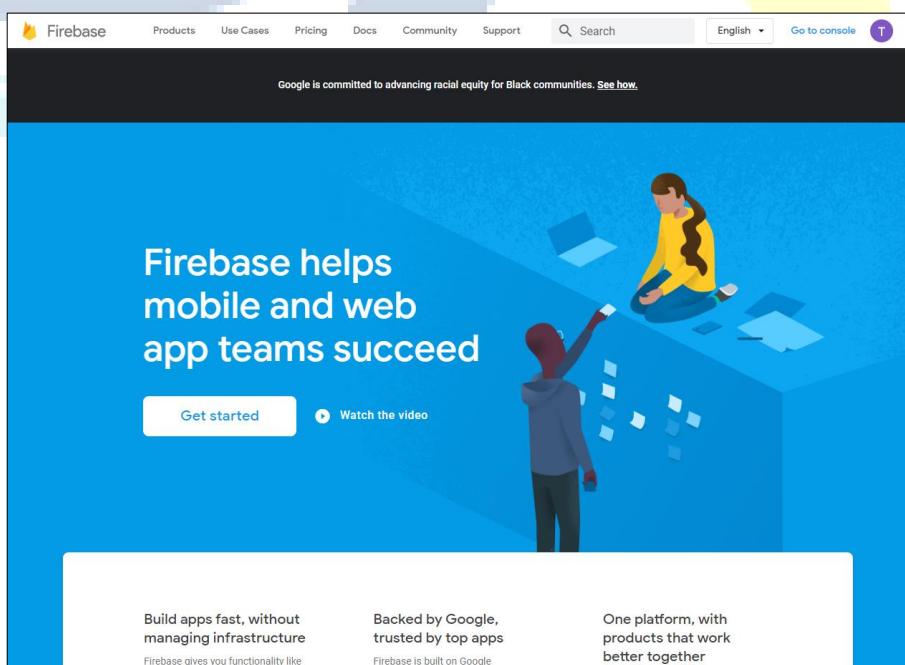
การสร้าง Project และ ตั้งค่าเพื่อใช้งานเครื่องมือของ Firebase

การใช้งานเครื่องมือบน Firebase เราจำเป็นจะต้องมี Gmail Account ก่อนซึ่งสามารถสมัครได้ผ่าน Google ทั้งนี้ จะไม่ขออธิบายวิธีการสมัครใช้งาน Gmail โดยการสร้างโปรเจกบน Firebase เราสามารถทำได้ดังนี้

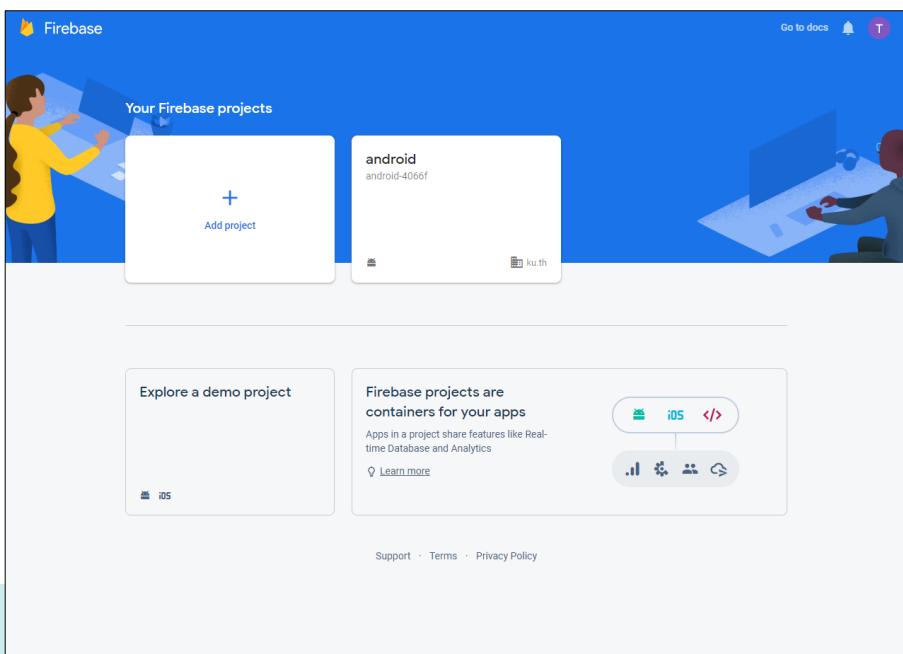
1. เปิดเว็บเบราว์เซอร์และทำการค้นหาโดยพิมพ์คีย์เวิร์ด Firebase



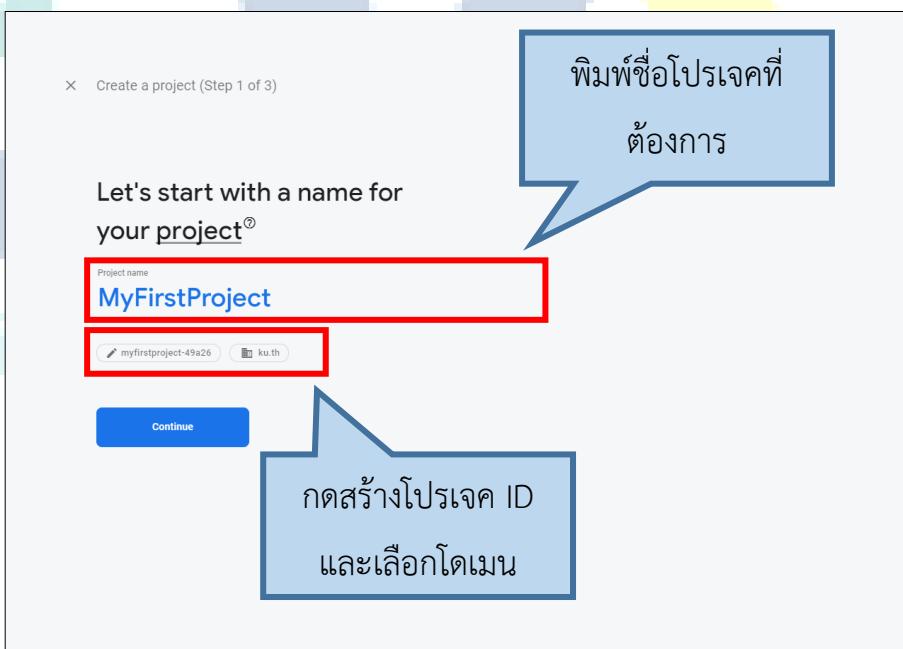
2. เปิดเข้าไปยัง WebSite แรกหรือ Firebase - Google Firebase ทั้งนี้เราสามารถเข้าถึงผ่าน URL <https://firebase.google.com/> ได้โดยตรง



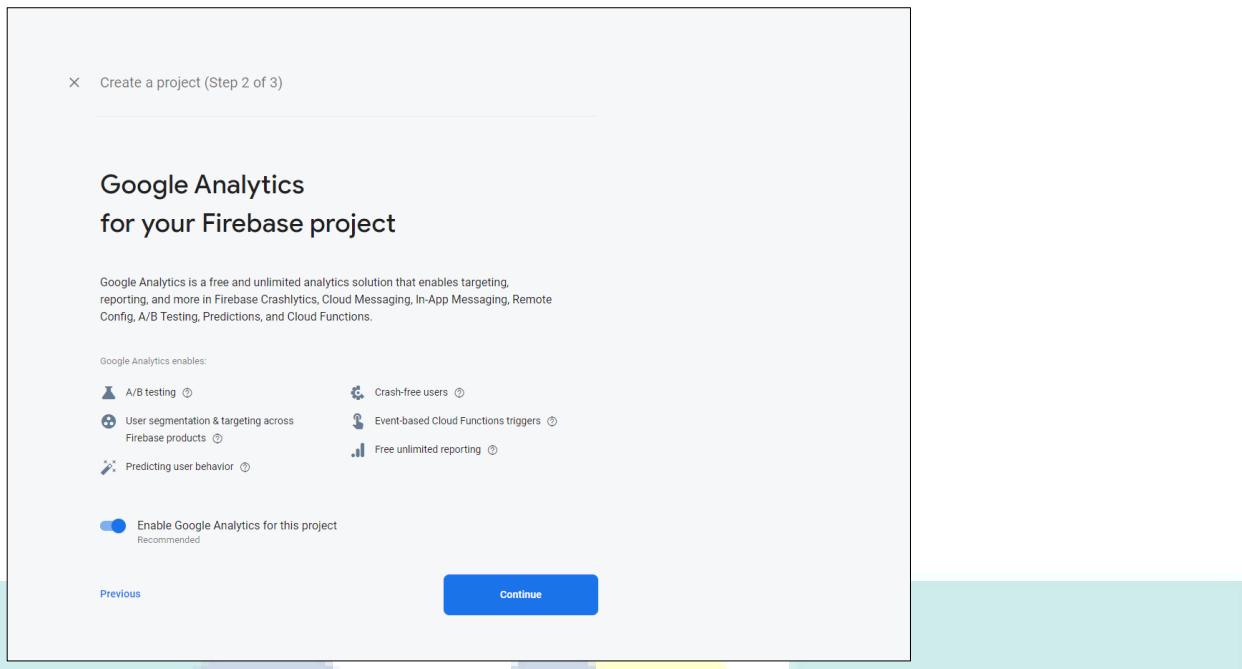
3. ทำการสร้างโปรเจกโดยทำการกดปุ่ม Add project



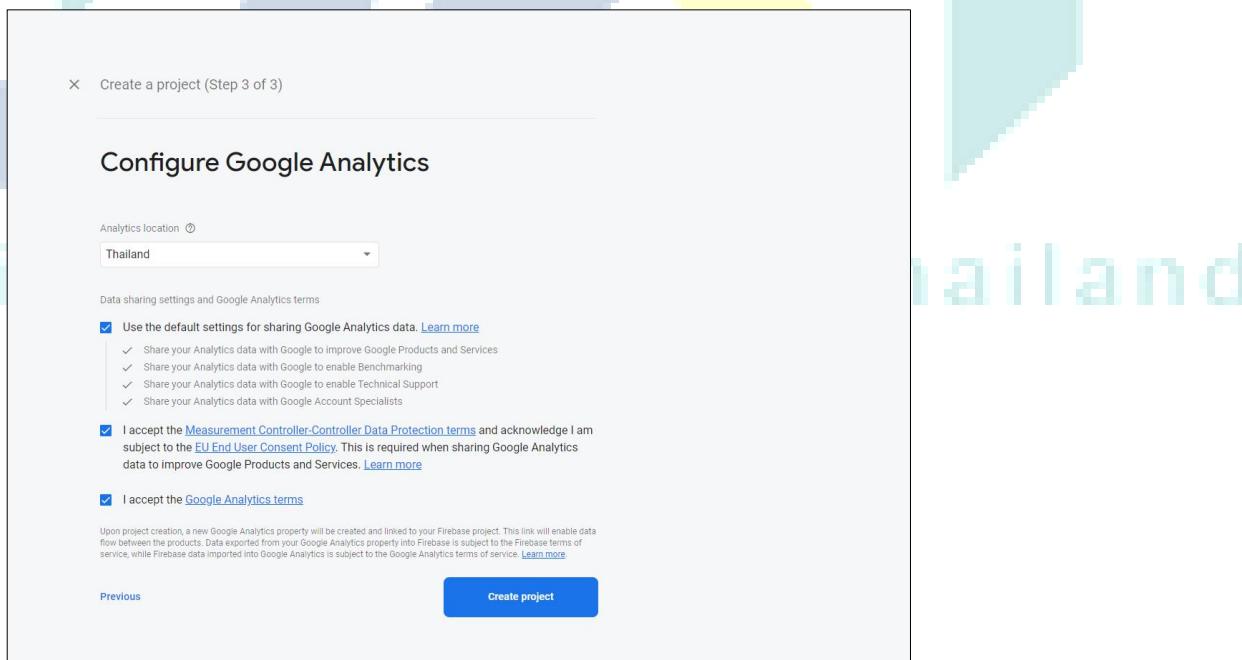
4. ทำการตั้งชื่อโปรเจค, สร้างโปรเจคoid และ กำหนดโดเมน เมื่อเสร็จสิ้นให้กด Continue



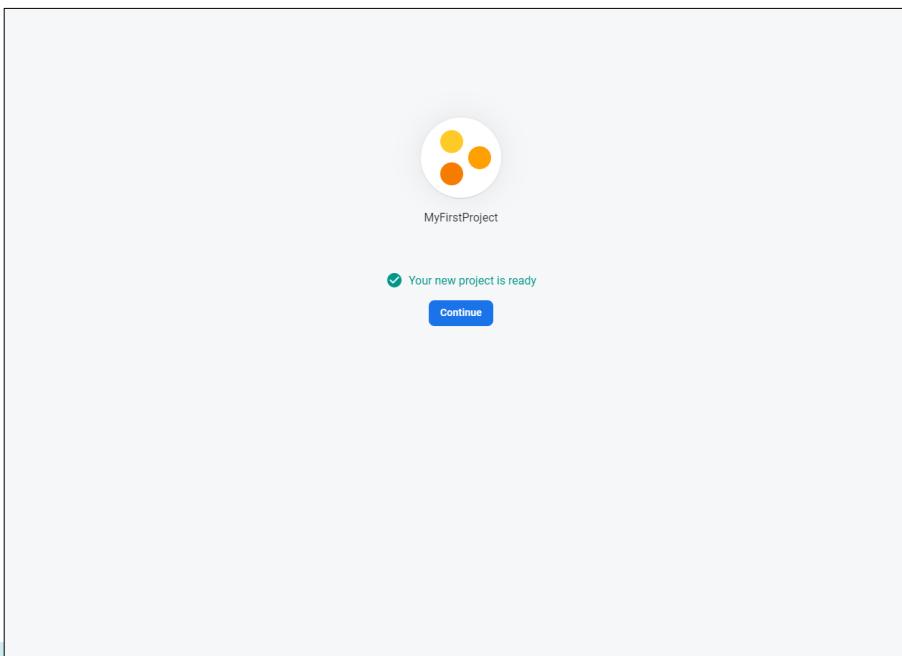
5. ในขั้นนี้ Google firebase จะสอบถามเรากيف่ากับการเปิดใช้งาน Google Analytics เราสามารถอนุญาตหรือไม่ก็ได้จากนั้นกด Continue



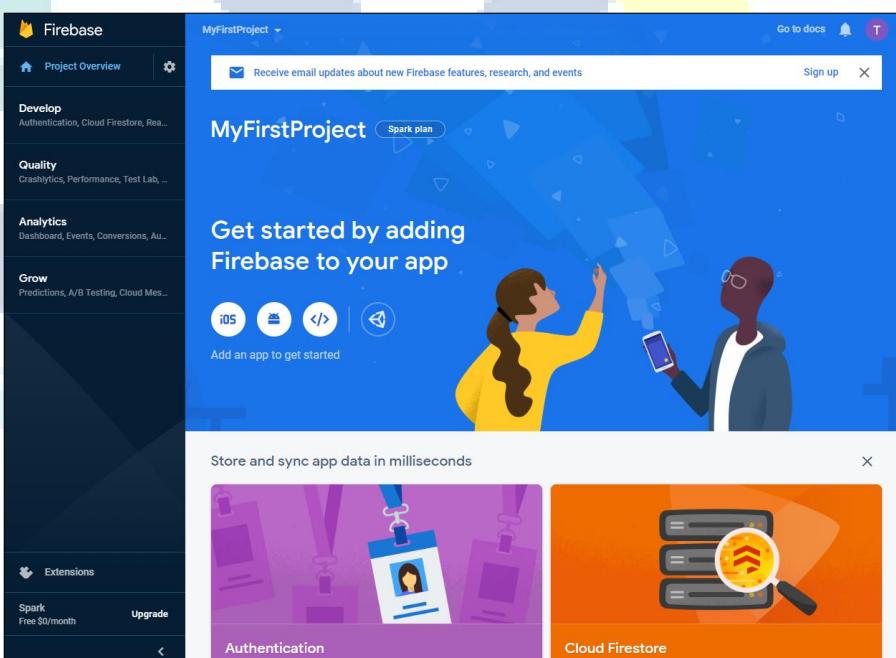
6. จากขั้นตอนที่แล้วหากเราอนุญาตใช้งาน Google Analytics ทาง Google firebase จะให้เรารับทราบกฎระเบียบเพิ่มเติมและเลือกประเภทที่ต้องการวิเคราะห์



7. จากนั้นให้รอสักครู่ระบบจะทำการสร้างโปรเจคจนเสร็จสมบูรณ์ให้กด Continue



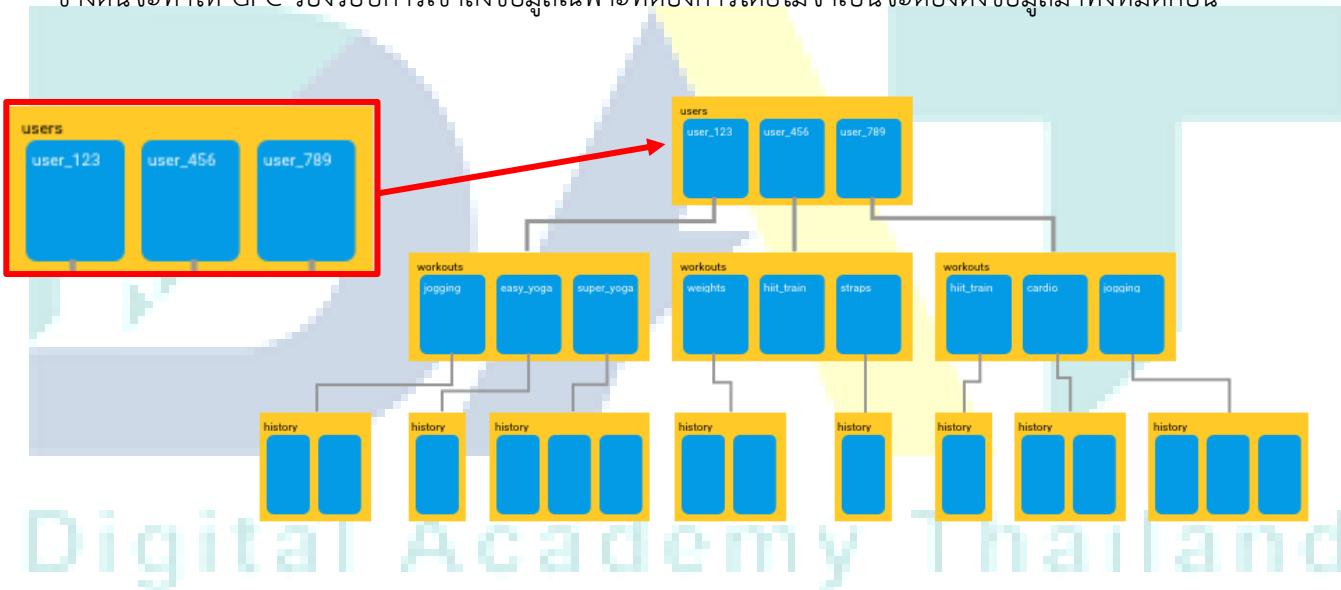
8. เมื่อสร้างโปรเจคเสร็จสมบูรณ์จะมายัง Project ที่สร้างไว้พร้อมใช้งาน



Google Cloud Firestore(GCF)

GFC คือบริการฐานข้อมูลแบบ NoSQL เมื่อต้องดึง MongoDB ซึ่งได้รับความนิยมอย่างแพร่หลาย แต่ GCF ตัวนี้ถูกพัฒนาด้วย Google ในลักษณะ Cloud Platform คือผู้ใช้งานไม่ต้องติดต่อโปรแกรมใดๆ เพื่อใช้งาน เมื่อต้องดึง MongoDB โดย GCF ถูกพัฒนาต่อยอกมากจาก Real-Time Database ซึ่งฐานข้อมูลแบบ NoSQL เมื่อกันนี้แล้ว GCF ได้ปรับปรุงคุณภาพในด้านต่างๆ ให้มีประสิทธิภาพสูงกว่าดังนี้

- เพิ่มประสิทธิภาพในการเข้าถึงข้อมูล(Better querying and more structured data) ในช่วงแรก Real-Time Database ได้ถูกสร้างขึ้นโดยมีการจัดเก็บข้อมูลในรูปแบบ Tree : ซึ่งจะมีปัญหาในการเข้าถึงข้อมูลหากจัดเก็บข้อมูลปริมาณมาก ด้วยเหตุนี้ GCF จึงได้เปลี่ยนวิธีจัดเก็บข้อมูลในรูปแบบ Collection Document ตามรูปภาพด้านล่าง ทั้งนี้ในการเก็บข้อมูลจริงยังมีการรองรับการเก็บข้อมูลที่มีลักษณะ Collection ซ้อน Collection กล่าวคือจะมี Sub collections ซ้อนอยู่ข้างในซึ่งมีการเก็บข้อมูลแบบลำดับชั้นตามตัวอย่างภาพด้านล่าง โดยการเก็บข้อมูลในรูปแบบข้างต้นจะทำให้ GFC รองรับการเข้าถึงข้อมูลเฉพาะที่ต้องการโดยไม่จำเป็นจะต้องดึงข้อมูลมาทั้งหมดก่อน



- ขยายฐานข้อมูลให้ใหญ่ได้โดยไม่มีผลต่อการเข้าถึงข้อมูล(Designed to Scale) GFC ได้ถูกออกแบบให้สามารถขยายฐานข้อมูลให้ใหญ่ขึ้นเพื่อรับข้อมูลปริมาณมหาศาล โดยที่ไม่เปลี่ยนขนาดใหญ่ขึ้นเรื่อยๆ แม้จะมีผลต่อการเข้าถึงกล่าวคือจะเข้าถึงข้อมูลได้ช้าลงแต่ GFC สามารถเข้าถึงได้อย่างรวดเร็ว
- เข้าถึงข้อมูลแบบเรียลไทม์ได้อย่างง่ายดาย(Easier manual fetching of data) เนื่องด้วย GFC มีการพัฒนาต่อยอกจาก Real-Time Database ซึ่งมีจุดเด่นคือสามารถเข้าถึงข้อมูลได้แบบ Real-Time ตามชื่อ ซึ่ง GFC เองก็ยังคงจุดเด่นนี้ไว้
- สามารถเข้าถึงข้อมูลได้ในทุกๆ ที่(Multi-Region support for better reliability and more locations) หากเราใช้งานฐานข้อมูลส่วนตัวเรามักจะมีการสร้างที่เก็บฐานข้อมูลน้ำหนักเดิมที่หนึ่งซึ่งแตกต่างกัน GFC ที่มีการเก็บข้อมูลในหลายๆ ที่ๆ เพื่อป้องกันข้อมูลสูญหายเมื่อที่ใดที่หนึ่งเกิดปัญหาน้ำหนักอีกทั้งเรายังสามารถเข้าถึงข้อมูลได้จากทุกภูมิภาคทั่วโลก
- คิดราคาขึ้นกับจำนวนการอ่านและเขียนข้อมูล(Different pricing model) หากเราคำนึงตัดสินใจเลือกใช้งาน GFC สิ่งเราต้องคำนึงถึงเป็นลำดับต้นๆ คือราคาค่าบริการ โดยราคาค่าบริการจะคิดจากจำนวนการอ่านและเขียนข้อมูลใน

ฐานข้อมูลโดยจะมีข้อดีคือ หากเราทำการเก็บข้อมูลครั้งละปริมาณมากๆ ก็จะทำให้จ่ายค่าบริการที่ถูกลงไปด้วยเมื่อเทียบกับ Real-Time database ซึ่งคิดราคาตามจำนวนข้อมูลที่อ่านและเขียนต่อครั้ง

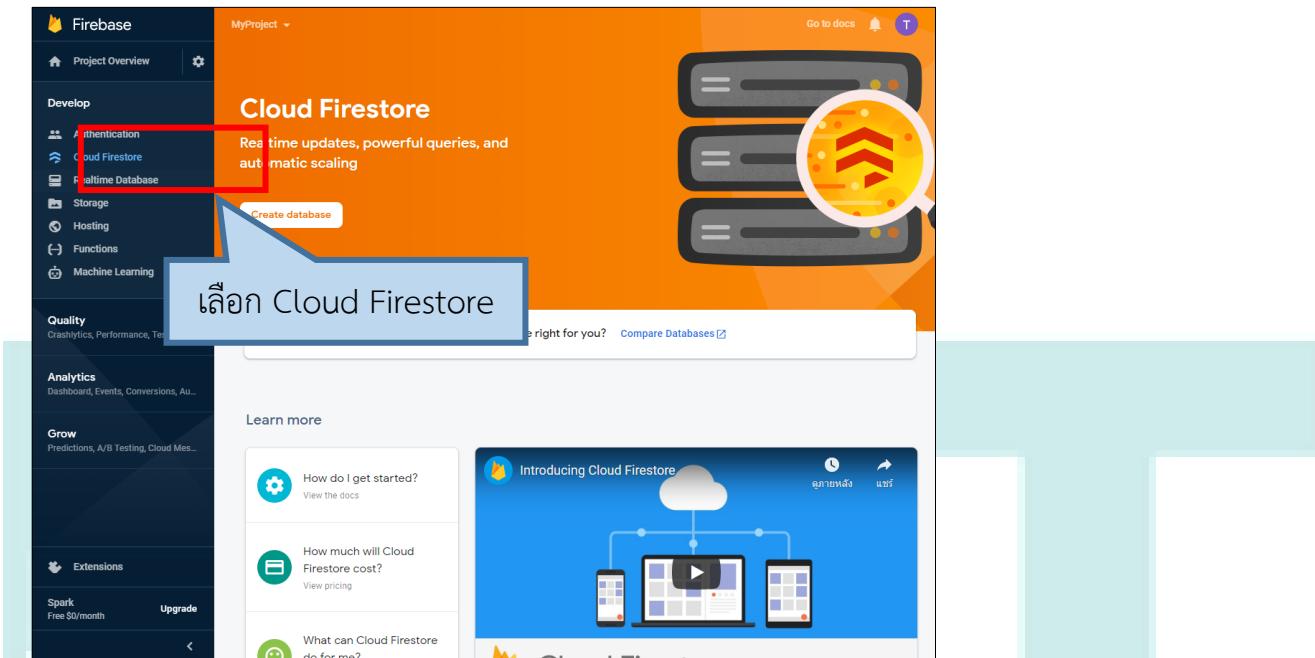


Digital Academy Thailand

การใช้งาน Google Cloud Firestore

การใช้งาน GCF เราจำเป็นจะต้องทำการสร้างฐานข้อมูล, กำหนดเงื่อนไขต่างๆที่จำเป็นต่อการใช้งานและทำการสร้าง Collection ไว้เพื่อรับการจัดเก็บข้อมูลจากแอพฯที่เราสร้าง โดยแอพฯที่เราสร้างนี้จะประกอบไปด้วยระบบสมัครสมาชิกและรองรับการแก้ไขข้อมูล เพราะฉะนั้นฐานข้อมูลเราจะต้องมีการรองรับส่วนนี้ด้วย โดยมีวิธีการสร้างดังนี้

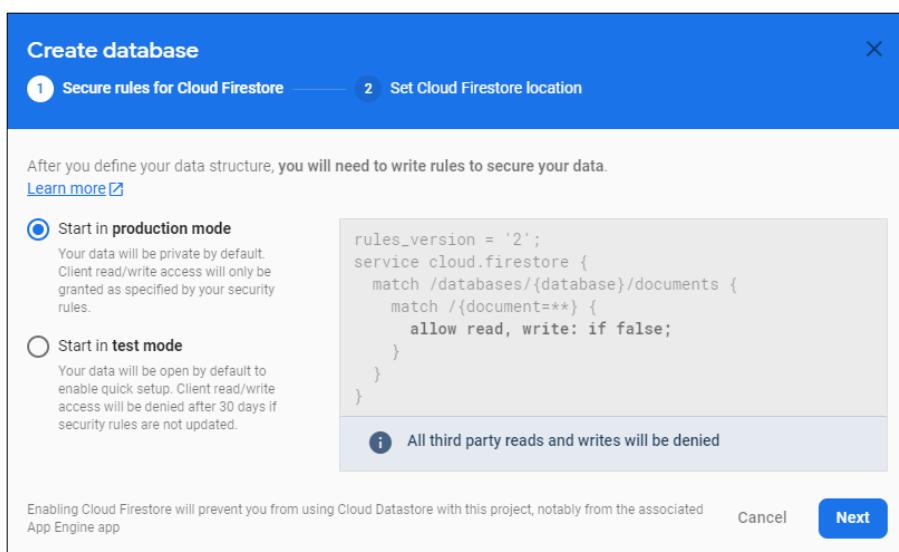
1. เมื่อเปิดมายังหน้าแรกให้เราเลือก Cloud Firestore จากแท็บเครื่องมือ Development



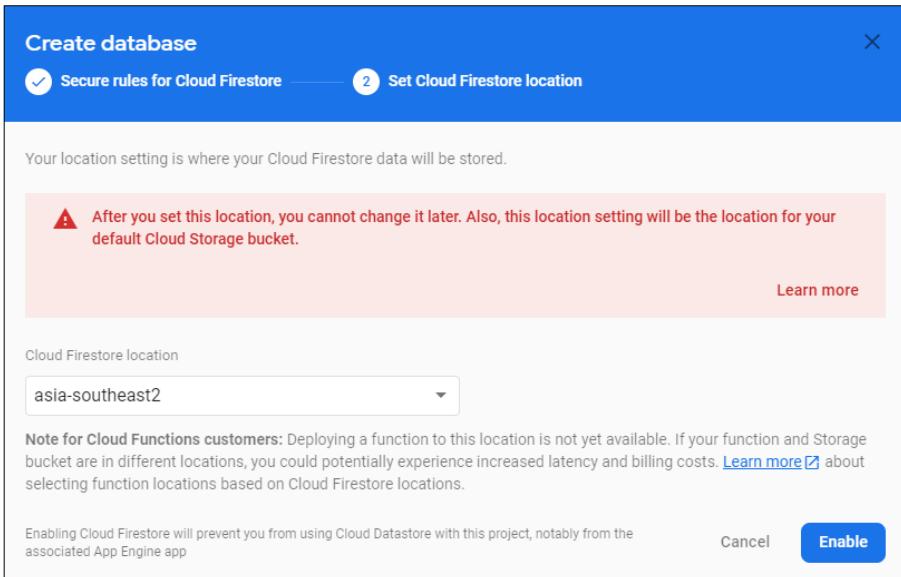
2. จากนั้นเราจะต้องทำการเลือกกลักษณ์ฐานข้อมูลที่ต้องการสร้างซึ่งมี 2 แบบคือ

- a. Start in production mode เป็นการสร้างฐานข้อมูลที่เราสามารถแก้ไข กำหนด นโยบายความเป็นส่วนตัวได้
- b. Start in test mode เป็นการสร้างฐานข้อมูลที่เราจัดการเข้าถึงแก้ไขอัปเดตได้ภายใน 30 นาทีตั้งแต่วันสร้าง

ในตัวอย่างนี้เราจะทำการสร้างฐานข้อมูลในลักษณะ Start in production mode เพื่อง่ายต่อการตั้งค่าต่างๆ

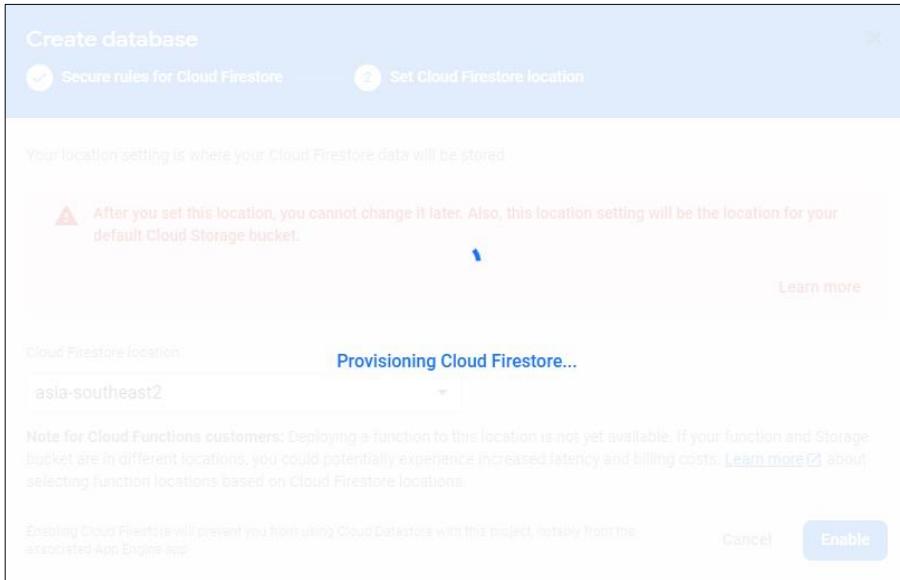


3. เลือกภูมิภาคหลักในการสร้างฐานข้อมูล โดยในที่นี่จะพยายามเลือกให้ใกล้กับประเทศที่เราอยู่ให้มากที่สุดซึ่งประเทศไทยอยู่ในเขตภูมิเอเชียตะวันออกเฉียงใต้ เราจึงเลือก asia-southeast และจึงกดปุ่ม Enable

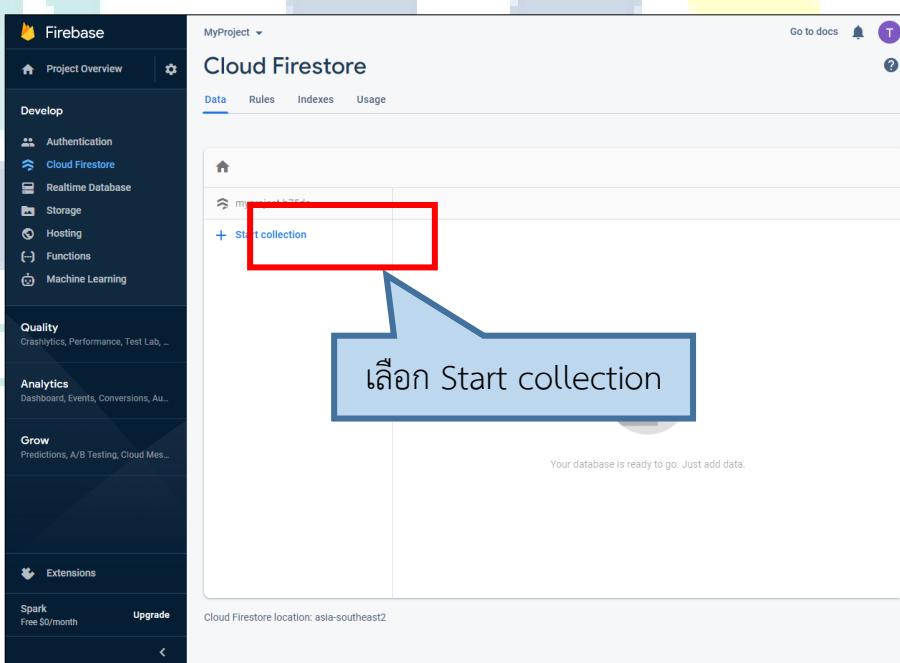


Digital Academy Thailand

4. GCF จะทำการสร้างฐานข้อมูลโดยใช้เวลาสักพักให้เรารอจนกว่าจะเสร็จสมบูรณ์



5. หลังจากสร้างฐานข้อมูลเสร็จแล้ว ถ้ามาระยะต่อไปที่ต้องการสร้าง Collection เพื่อรับข้อมูลที่จะป้อนเข้ามาทั้งนี้เราว่าจะมีการสร้างข้อมูลทดสอบไว้ด้วยเพื่อที่ใช้อ้างอิงข้อมูลที่จะอินพุตเข้ามาจากแอพฯที่เราจะเขียน โดยทำการกดไปที่ Start collection



6. จากนั้นทำให้ชื่อ Collection ที่สื่อความหมาย ทั้งนี้ເພົ່າທີ່ເຮົາຈະສ້າງມີຮບບສົມຄຣສາຊືກເຮົາຈຶ່ງສ້າງ Collection ທີ່ມີ
ຊື່ວ່າ Account

Start a collection

1 Give the collection an ID — 2 Add its first document

Parent path /

Collection ID

A collection is a set of documents that contain data
Example: Collection "users" would contain a unique document for each user

Cancel **Next**

7. จากนั้นทำการใส่ข้อมูลตัวอย่างที่เราต้องการ ลงใน Document ทั้งนี้ในส่วนของ Document ID สามารถกำหนดเอง
หรือให้ระบบสร้างให้ได้ ในตัวอย่างเราจะให้ระบบสร้างให้

Start a collection

1 Give the collection an ID — 2 Add its first document

Document parent path /Account

Document ID

Field	Type	Value
ID	string	5230300540
FirstName	string	Tamnuwat
LastName	string	Valeeprakhon

โดยข้อมูลทั้งหมดแสดงในตารางด้านล่าง

Field	Type	Value
ID	String	5230300540
FirstName	String	Tamnuwat
LastName	String	Valeeprakhon
Address	String	199 Moo.9
Tambon	String	Tungsukla
Amphur	String	Sriracha
Province	String	Chonburi
PostalCode	String	20230
CreatedDate	Timestamp	9/9/2020
UserName	String	tamnuwat

8. เมื่อทำการสร้าง Document เสร็จแล้วเราจะสามารถดูข้อมูลเหล่านี้ได้

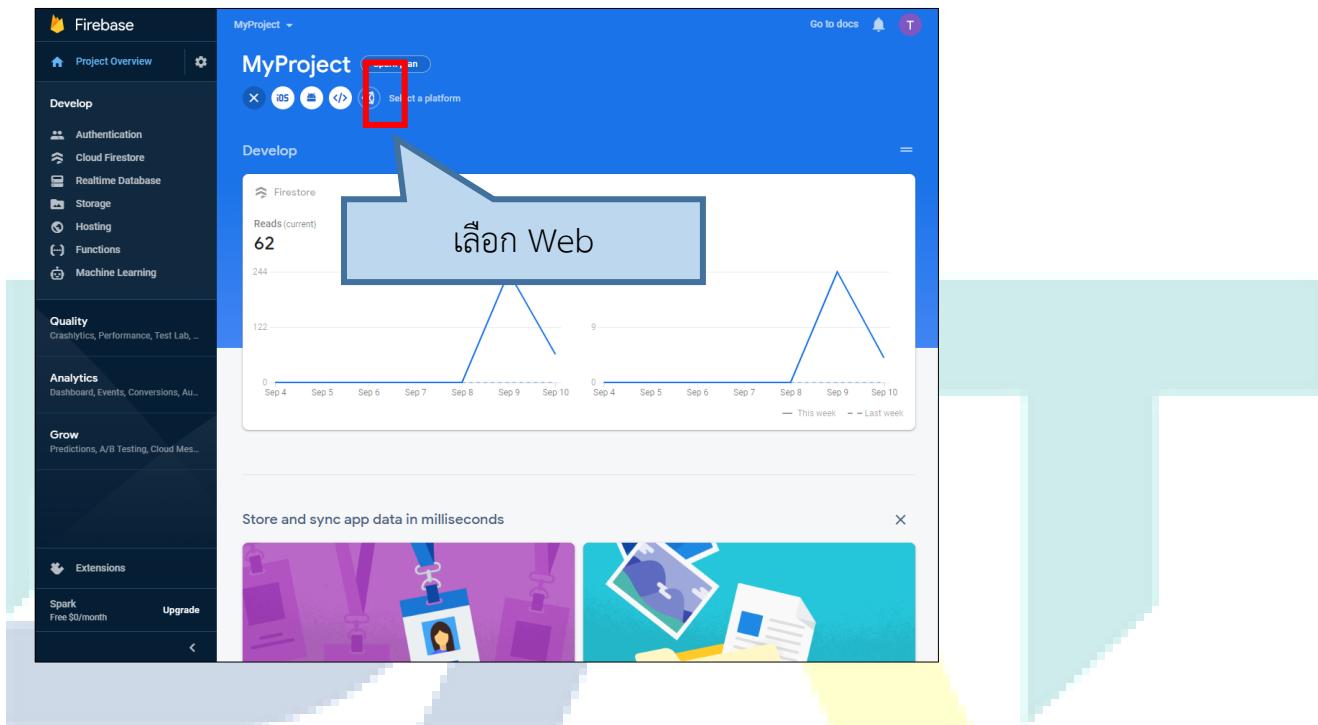
The screenshot shows the MongoDB Atlas interface. At the top, there's a navigation bar with a home icon, 'Account' (selected), and a document icon. Below the navigation, there are three main buttons: '+ Start collection', '+ Add document', and '+ Start collection'. The middle section shows a tree view of collections: 'Account' > 'uTcdZrWZ1TeD6Wm4l0ED'. The right side of the screen displays the contents of the selected document:

```
Address: "199 Moo.9"
Amphur: "Sriracha"
CreatedDate: September 9, 2020 at 12:00:00 AM UTC+7
FirstName: "Tamnuwat"
ID: "5230300540"
LastName: "Valeeprakhon"
PostalCode: "20230"
Province: "Chonburi"
Tambon: "Tungsukla"
UserName: "tamnuwat"
```

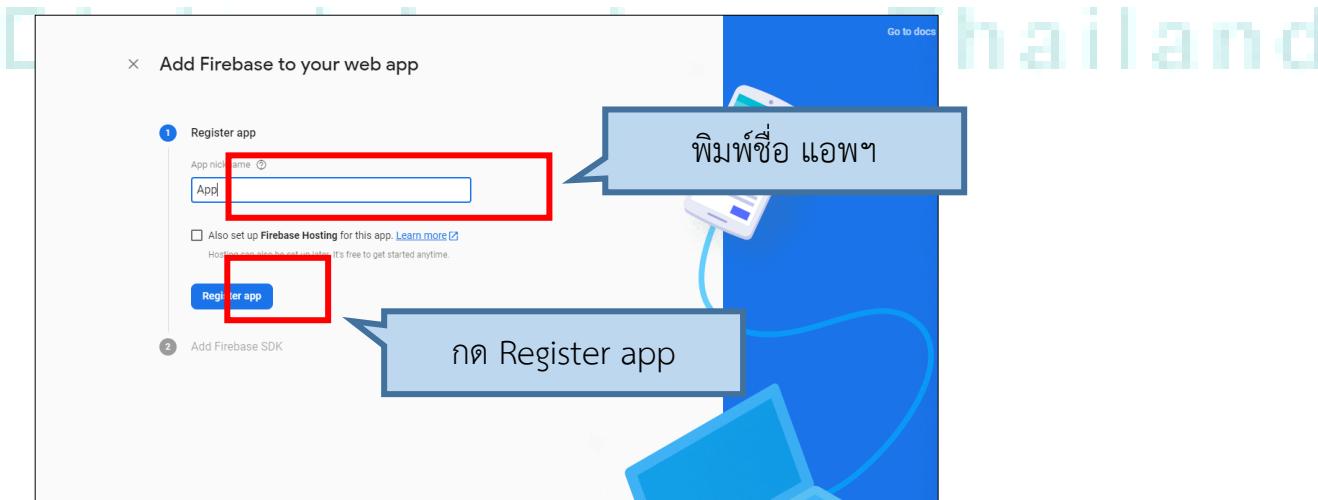
การสร้าง Configuration Script เพื่อเชื่อมต่อแอพฯ กับ Google Cloud Firestore

การเชื่อมต่อ GCF กับแอพฯ ของเราระบบ Hybrid Platform ที่ทำงานร่วมกันระหว่าง JavaScript core กับ Native core ซึ่งการเชื่อมต่อเราจะทำการเชื่อมต่อให้ฟัง JavaScript core เมื่อนักพัฒนาต้องการใช้งาน API ของ GCF บน JavaScript core

1. เปิด GCF มาที่หน้า Console
2. กดที่ไอคอน </> Web Platform



3. ทำการใส่ชื่อแอพฯ ที่เราสร้างและกด Register app



4. จากนั้น GCF จะสร้าง Firebase configuration script ดังภาพด้านล่าง ซึ่งเราจะใช้ Script ผ่านไปใน แอพฯ ของเรา
เนื่องด้วย Script นี้จะต้องเก็บเป็นความลับ เพราะฉะนั้นเราไม่ควรให้คนอื่นทราบเด็ดขาดมิฉะนั้นอาจจะถูกขโมยข้อมูลได้

Register app

Add Firebase SDK

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src='https://www.gstatic.com/firebasejs/7.20.0.firebaseio-app.js'></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-libraries -->
<script src='https://www.gstatic.com/firebasejs/7.20.0/firebase-analytics.js'></script>

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyRwJ0Q0Yr4CDTb503nGHuWKqYR5TNKck",
    authDomain: "myproject-b75da.firebaseioapp.com",
    databaseURL: "https://myproject-b75da.firebaseio.com",
    projectId: "myproject-b75da",
    storageBucket: "myproject-b75da.appspot.com",
    messagingSenderId: "1041757860081",
    appId: "1:1041757860081:web:6e06968e71785dcdb17950",
    measurementId: "G-84X265XWNS"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
</script>
```

Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

Continue to console

การเพิ่มข้อมูลใน Google Cloud Firestore

ก่อนการเพิ่มข้อมูลลงใน GCF เราจำเป็นจะต้องทราบถึงลักษณะของข้อมูลเสียก่อนโดยข้อมูลที่ต้องการเพิ่มลงใน GCF จะเป็นลักษณะ Object คือตัวแปรที่มีการเก็บข้อมูลในลักษณะ key-Value ตัวอย่างด้านล่าง

```
var docData = {  
    name: "Los Angeles",  
    state: "CA",  
    country: "USA"  
}
```

ในบางครั้งข้อมูลที่เราต้องการเก็บจะมีความสำพันธ์ในลักษณะซ้อนกัน(Sub-Collections) ซึ่งก็สามารถเพิ่มลงใน GCF ได้ เช่นกันโดยมีลักษณะตามตัวอย่างด้านล่าง

```
var docData = {  
    stringExample: "Hello world!",  
    booleanExample: true,  
    numberExample: 3.14159265,  
    dateExample: firebase.firestore.Timestamp.fromDate(new Date("December 10, 1815")),  
    arrayExample: [5, true, "hello"],  
    nullExample: null,  
    objectExample: {  
        a: 5,  
        b: {  
            nested: "foo"  
        }  
    }  
};
```

การเพิ่มข้อมูลลงใน GCF เราสามารถทำได้ 2 ลักษณะคือการเพิ่มแบบไม่ระบุ document กับ การสร้างแบบระบุ document โดยทั้งสองแบบนี้จะมีการใช้งานคล้ายๆกันแต่มีวัตถุประสงค์การใช้งานที่ต่างกันดังนี้

- การเพิ่มแบบไม่ระบุ document เป็นการเพิ่มข้อมูลโดยให้ระบบเป็นคนสร้าง Document ID ที่ใช้อ้างอิง เมื่อทำการเพิ่มข้อมูลเสร็จ GCF จะส่ง document id กลับมา焉ั้งแอพฯของเรา โดยมากเรามักจะใช้การเพิ่มข้อมูลด้วยวิธีนี้กับข้อมูลที่ไม่มีการเฉพาะเจาะจงโดยจะใช้คำสั่งตามตัวอย่างด้านล่าง

```
db.collection(Your_Collection_Name).add(Your_Object_Data)  
    .then(function(docRef) {  
        // Add Success  
    })  
    .catch(function(error) {  
        // Add Unsuccess  
    });
```

ตัวอย่างโค้ด

```
var data={  
    Address : "199 Moo.9",  
    Amphur : "Sriracha",  
    CreatedDate : firebase.firestore.FieldValue.serverTimestamp(),  
    FirstName : "Tamnuwat",  
    LastName : "Valeeprakhon",  
    ID: "5230300540",  
    PostalCode: "20230",  
    Province: "Chonburi",
```

```

        Tambon: "Tungsukla",
        UserName: "tamnuwat"
    }
    db.collection('Account').add(data)
        .then(function(docRef) {
            console.log(docRef.id) // Print id if add success
        })
        .catch(function(error) {
            console.log(error);
        });
    });

```

ผลลัพธ์ที่ได้

myproject-b75da	Account	uTcdZrWZlTeD6Wm4l0ED
+ Start collection	+ Add document	+ Start collection
Account >	uTcdZrWZlTeD6Wm4l0ED >	+ Add field
		Address: "199 Moo.9" Amphur: "Sriracha" CreatedDate: September 9, 2020 at 12:00:00 AM UTC+ FirstName: "Tamnuwat" ID: "5230300540" LastName: "Valeeprakhon" PostalCode: "20230" Province: "Chonburi" Tambon: "Tungsukla" UserName: "tamnuwat"

2. การสร้างแบบรูปแบบ document เป็นการเพิ่มข้อมูลโดยเราเป็นผู้กำหนด Document ID ที่ใช้อ้างอิงเมื่อทำการเพิ่ม โดยมากเรามักจะใช้การเพิ่มข้อมูลด้วยวิธีนี้กับข้อมูลที่ต้องการอ่านข้อมูลโดยเฉพาะเจาะจง แต่เราควรระวังหากมีการ เขียนข้อมูลใหม่จะมีการทับข้อมูลเดิมทันทีโดยจะใช้คำสั่งตามตัวอย่างด้านล่าง

```

db.collection(Your_Collection_Name).doc(Your_Document_ID).set(Your_Object_Data)
    .then(function(docRef) {
        // Add Success
    })
    .catch(function(error) {
        // Add Unsuccess
    });

```

ตัวอย่างโค้ด

```

var data={
    Address : "199 Moo.9",
    Amphur : "Sriracha",
    CreatedDate : firebase.firestore.FieldValue.serverTimestamp(),
    FirstName : "Janejira",
    LastName : "Wangsaklang",
    ID: "5230300650",
    PostalCode: "20230",
    Province: "Chonburi",
    Tambon: "Tungsukla",
    UserName: "tamnuwat"
}

```

```

}
db.collection('Account').doc('5230300650').add(data)
  .then(function(docRef) {
    console.log(docRef.id) // Print id if add success
  })
  .catch(function(error) {
    console.log(error);
  });

```

ผลลัพธ์ที่ได้

The screenshot shows the MongoDB Atlas interface with a document in the 'Account' collection. The document has the ID '5230300650'. The fields and their values are:

- Address: "199 Moo.9"
- Amphur: "Sriracha"
- CreatedDate: September 12, 2020 at 12:49:29 AM UTC+7
- FirstName: "Janejira"
- ID: "5230300650"
- LastName: "Wangsaklang"
- PostalCode: "20230"
- Province: "Chonburi"
- Tambon: "Tungsukla"
- UserName: "janjira"

Digital Academy Thailand

การอ่านข้อมูลจาก Google Cloud Firestore

การอ่านข้อมูลจาก GCF สามารถทำได้ 2 วิธีได้แก่ 1. การอ่านแบบครั้งเดียว คือ การอ่านข้อมูลที่ล็อกรังตาม Event ที่กำหนดไว้โดยเราจะได้ข้อมูลในลักษณะของ Array of Object ออกมา ซึ่งอาจจะมีข้อมูลหรือไม่มีข้อมูลก็ได้ขึ้นอยู่ กับเงื่อนไขในการอ่านที่เราได้ตั้งไว้ 2. การอ่านแบบผู้ฟัง(listener) คือการอ่านข้อมูลแบบเรียลไทม์อ่านข้อมูลเมื่อมีการเปลี่ยนแปลงใดๆเกิดขึ้นในฐานข้อมูลของเราจะไม่ขึ้นกับ Event ของแอพฯ(โดยจะกล่าวโดยละเอียดในบทต่อไป)

เราได้ทำการ เรายังคงอ่านข้อมูลของเมืองต่างๆโดยทำการเพิ่มข้อมูลลงฐานข้อมูลด้วยคำสั่งด้าน

```
var citiesRef = db.collection("cities");

citiesRef.doc("SF").set({
  name: "San Francisco", state: "CA", country: "USA",
  capital: false, population: 860000,
  regions: ["west_coast", "norcal"] });

citiesRef.doc("LA").set({
  name: "Los Angeles", state: "CA", country: "USA",
  capital: false, population: 3900000,
  regions: ["west_coast", "socal"] });

citiesRef.doc("DC").set({
  name: "Washington, D.C.", state: null, country: "USA",
  capital: true, population: 680000,
  regions: ["east_coast"] });

citiesRef.doc("TOK").set({
  name: "Tokyo", state: null, country: "Japan",
  capital: true, population: 9000000,
  regions: ["kanto", "honshu"] });

citiesRef.doc("BJ").set({
  name: "Beijing", state: null, country: "China",
  capital: true, population: 21500000,
  regions: ["jingjinji", "hebei"] });
```

ผลลัพธ์ที่ได้แสดงด้านล่าง

The screenshot shows the Google Cloud Firestore interface. On the left, there's a navigation bar with a home icon, followed by 'myproject-b75da', 'cities', and 'TOK'. Below this is a sidebar with 'Start collection' and 'Add document' buttons, and a list of documents: 'BJ', 'DC', 'LA', 'SF', and 'TOK'. The 'TOK' document is currently selected. The main panel displays the document details for 'TOK':
- Fields:
 - capital: true
 - country: "Japan"
 - name: "Tokyo"
 - population: 9000000
 - regions:
 - 0: "kanto"
 - 1: "honshu"
 - state: null

การอ่านข้อมูลแบบครั้งเดียวเราสามารถทำได้ 2 แบบคือการอ่านข้อมูล 1 Document และการอ่านแบบหลาย Document ซึ่งทั้งสองอย่างนี้จะมีรูปแบบคำสั่งและการเข้าถึงที่ข้อมูลที่ต่างกัน

- การอ่านข้อมูล 1 Document คือการอ่านข้อมูลที่เฉพาะเจาะจงระบุ DocumentID ในการค้นหาข้อมูลซึ่งผลลัพธ์ที่ได้จะได้ข้อมูลเพียง 1 Document เท่านั้นโดยใช้คำสั่งด้านล่าง

```
var docRef = db.collection("Your_Collection").doc("Your_DocumentID");
docRef.get().then(function(doc) {
  if (doc.exists) {
    // IF Success and Document existing
  } else {
    // Document Does'not existing
  }
}).catch(function(error) {
  // Error
});
```

ตัวอย่างทำการดึงข้อมูลจาก Collection ที่มีชื่อว่า cities และต้องการ DocumentID ชื่อว่า SF

```
var docRef = db.collection("cities").doc("SF");

docRef.get().then(function(doc) {
  if (doc.exists) {
    console.log("Document data:", doc.data());
  } else {
    // doc.data() will be undefined in this case
    console.log("No such document!");
  }
}).catch(function(error) {
  console.log("Error getting document:", error);
});
```

ผลลัพธ์ที่ได้แสดงด้านล่าง

```
{
  name: "San Francisco", state: "CA", country: "USA",
  capital: false, population: 860000,
  regions: ["west_coast", "norcal"]
}
```

- การอ่านข้อมูลแบบหลาย Document คือการดึงข้อมูลที่ไม่มีการเจาะจง Document ใด Document โดยจะอาศัยการกรองข้อมูลตามเงื่อนไขที่ต้องการหรือทำการอ่านข้อมูลทั้งหมดใน Collection ซึ่งผลลัพธ์ที่ได้อาจจะมากกว่า 1 Document

- การอ่านข้อมูลทั้งหมด คือการอ่านข้อมูลทั้งหมดภายใน Collection โดยผลลัพธ์ที่ได้จะเป็น Document ทั้งหมดที่อยู่ภายใต้ Collection

```
db.collection("Your_Collection").get().then(function(querySnapshot) {
  querySnapshot.forEach(function(doc) {
    // doc.data() is never undefined for query doc snapshots
    console.log(doc.id, " => ", doc.data());
  });
});
```

ตัวอย่างทำการดึงข้อมูลทั้งหมดภายใน Collection ที่ชื่อว่า cities

```
db.collection("cities").get().then(function(querySnapshot) {
  querySnapshot.forEach(function(doc) {
    // doc.data() is never undefined for query doc snapshots
  });
});
```

```
//console.log(doc.data());
});
});
```

ผลลัพธ์ที่ได้

```
{name: "San Francisco", state: "CA", country: "USA",
  capital: false, population: 860000,
  regions: ["west_coast", "norcal"]}

{name: "Los Angeles", state: "CA", country: "USA",
  capital: false, population: 3900000,
  regions: ["west_coast", "socal"]}

{name: "Washington, D.C.", state: null, country: "USA",
  capital: true, population: 680000,
  regions: ["east_coast"] }

{name: "Tokyo", state: null, country: "Japan",
  capital: true, population: 9000000,
  regions: ["kanto", "honshu"] }

{name: "Beijing", state: null, country: "China",
  capital: true, population: 21500000,
  regions: ["jingjinji", "hebei"] }
```

2.2. การอ่านข้อมูลด้วยการกำหนดเงื่อนไขการกรอง(Query operator) คือ การอ่านข้อมูลโดยมีการกำหนดเงื่อนไขของข้อมูลซึ่งเราจะใช้เมธอด(Method) ที่ชื่อว่า where() เป็นตัวกรอกโดยมีรูปแบบด้านล่าง

```
where("Your_field", "Your_operator", "Your_parameter");
```

เงื่อนไขการกรอง

Operators	ความหมาย
<	น้อยกว่า
<=	น้อยกว่าหรือเท่ากับ
==	เท่ากับ
>	มากกว่า
>=	มากกว่าหรือเท่ากับ
array-contains	ข้อมูลที่อยู่ในอาร์เรย์ 1 ตัว
In or array-contains-any	ข้อมูลที่อยู่ในอาร์เรย์มากกว่าหรือเท่ากับ 1 ตัว

ตัวอย่างการกำหนดเงื่อนไข

```
db.collection("cities").where("capital", "==", true)
.get()
.then(function(querySnapshot) {
  querySnapshot.forEach(function(doc) {
    console.log(doc.data());
  });
})
.catch(function(error) {
  console.log("Error getting documents: ", error);
});
```

หรือ

```
var citiesRef = db.collection("cities");
var query = citiesRef.where("capital", "==", true);
query.get()
.then(function(querySnapshot) {
  querySnapshot.forEach(function(doc) {
    console.log(doc.data());
  });
})
.catch(function(error) {
  console.log("Error getting documents: ", error);
});
```

ผลลัพธ์ที่ได้

```
{name: "Beijing", state: null, country: "China",
capital: true, population: 21500000,
regions: ["jingjinji", "hebei"] }

{name: "Tokyo", state: null, country: "Japan",
capital: true, population: 9000000,
regions: ["kanto", "honshu"] }

{name: "Washington, D.C.", state: null, country: "USA",
capital: true, population: 680000,
regions: ["east_coast"] }
```

ตัวอย่างการกรอกข้อมูลโดยใช้ Operators ตัวอื่นๆ

```
citiesRef.where("population", "<", 100000)
citiesRef.where("name", ">=", "San Francisco")
citiesRef.where("regions", "array-contains", "west_coast")
citiesRef.where('country', 'in', ['USA', 'Japan']);
citiesRef.where('regions',           'array-contains-any', ['west_coast',
'east_coast']);
citiesRef.where('region', 'in', [['west_coast', 'east_coast']] );
```

2.3. การอ่านข้อมูลแบบเรียงลำดับ(Order data) คือการอ่านข้อมูลโดยจะมีการเรียงลำดับ(Order)จากข้อมูลจากมากไปน้อยหรือจากน้อยไปมาก โดยเราจะใช้เมธอดด้านล่าง

```
orderBy("Your_field", optional blink or "desc")
```

หากไม่มีการกำหนดพารามิเตอร์ใดจะเป็นการเรียงลำดับจากน้อยไปมากแต่ถ้าหากกำหนดพารามิเตอร์เป็น "desc" จะเป็นการเรียงจากมากไปน้อย

```
var citiesRef = this.db.collection('cities');
var query = citiesRef.where('population', ">", 5000000)
    .orderBy('population',"desc");
query.get()
.then(function (querySnapshot) {
    querySnapshot.forEach(function (doc) {
        console.log(doc.data());
    });
})
.catch(function (error) {
    console.log('error');
});
```

ตัวอย่างผลลัพธ์ที่ได้

```
{name: "Beijing", state: null, country: "China",
capital: true, population: 21500000,
regions: ["jingjinji", "hebei"] }

{name: "Tokyo", state: null, country: "Japan",
capital: true, population: 9000000,
regions: ["kanto", "honshu"] }
```

หมายเหตุ ใช้งานเมธอด where ร่วมกับ orderBy เราจะต้องใช้ field เดียวกันเท่านั้น

2.4. การจำกัดข้อมูล(Limit) คือการระบุจำนวนที่ต้องอ่านโดยผลลัพธ์ที่ได้จะมีขนาดเท่ากับจำนวนที่กำหนดไว้ซึ่งเราจะใช้เมธอดด้านล่างเป็นตัวกำหนด

```
limit(Your_Limit_Number)
```

ตัวอย่างการใช้งาน limit

```
var citiesRef = this.db.collection('cities');
var query = citiesRef.orderBy('population',"desc").limit(1);
query.get()
.then(function (querySnapshot) {
    querySnapshot.forEach(function (doc) {
        console.log(doc.data());
    });
})
.catch(function (error) {
    console.log('error');
});
```

ผลลัพธ์ที่ได้

```
{name: "Beijing", state: null, country: "China",
capital: true, population: 21500000,
regions: ["jingjinji", "hebei"] }
```

การอัปเดตข้อมูลใน Google Cloud Firestore

การอัปเดตข้อมูลคือการเปลี่ยนแปลงข้อมูลบาง field ในฐานข้อมูลซึ่งการอัปเดตข้อมูลนี้เราจะต้องทราบ DocumentID ที่แน่นอนเพื่อใช้เป็นตัวอย่างอ้างอิงตำแหน่งในการอัปเดตข้อมูลซึ่งการอัปเดตข้อมูลสามารถแบ่งได้เป็นสองประเภทตามประเภทข้อมูลคือการอัปเดตข้อมูลประเภท Object และ การอัปเดตข้อมูลประเภท Array

- การอัปเดตข้อมูลประเภท Object การอัปเดตข้อมูลประเภทนี้เราสามารถแก้ไขข้อมูลลงใน field ที่ต้องการได้ทันที ตัวอย่างการอัปเดตข้อมูลโดยจะใช้มетодด้านล่าง

```
update({key1:value1, key2:value2, ...})
```

ตัวอย่างการใช้งาน update

```
var citiesRef = this.db.collection('cities').doc('DC');
citiesRef.update({
  capital:false,
})
.then(function() {
  console.log("Document successfully updated!");
})
.catch(function(error) {
  console.error("Error updating document: ", error);
});
```

ก่อนอัปเดต	หลังอัปเดต
<pre>capital: true country: "USA" name: "Washington, D.C." population: 680000 ▼ regions 0 "east_coast" state: null</pre>	<pre>capital: false country: "USA" name: "Washington, D.C." population: 680000 ▼ regions 0 "east_coast" state: null</pre>

ผลลัพธ์ที่ได้

2. การอัปเดจข้อมูลประเภท Array เป็นด้วยในฐานข้อมูลอาจมีการเก็บข้อมูลในลักษณะ Object และมี Array ข้อนอยู่ด้านใน การแก้ไขสามารถทำได้ 2 ลักษณะคือการเพิ่มข้อมูลลงใน Array และการลบข้อมูลใน Array โดยใช้ Method ด้านล่าง

```
arrayUnion ("Your_parameter") // เพิ่มข้อมูล  
arrayRemove ("Your_parameter") // ลบข้อมูล
```

ตัวอย่างการใช้งาน

```
var citiesRef = this.db.collection('cities').doc('DC');  
citiesRef.update({  
    regions:firebase.firestore.FieldValue.arrayUnion("greater_virginia")  
});  
  
citiesRef.update({  
    regions: firebase.firestore.FieldValue.arrayRemove("east_coast")  
});
```

ผลลัพธ์ที่ได้

ก่อนอัปเดจ	หลังอัปเดจ
<pre>capital: true country: "USA" name: "Washington, D.C." population: 680000 ▼ regions 0 "east_coast" state: null</pre>	<pre>capital: false country: "USA" name: "Washington, D.C." population: 680000 ▼ regions 0 "greater_virginia" state: null</pre>

การลบข้อมูลใน Google Cloud Firestore

การลบข้อมูลคือการลบ Document ออกจากฐานข้อมูลเท่านั้นเราไม่สามารถลบ Collection ออกจากฐานข้อมูลได้ หากต้องการลบ Collection เราจะต้องเข้าไปลบด้วยตนเองใน Firebase Console การลบ Document เราสามารถลบได้ ด้วย Method ด้านล่าง

```
delete()
```

ตัวอย่างการใช้งาน

```
db.collection("cities").doc("DC").delete().then(function() {
    console.log("Document successfully deleted!");
}).catch(function(error) {
    console.error("Error removing document: ", error);
});
```

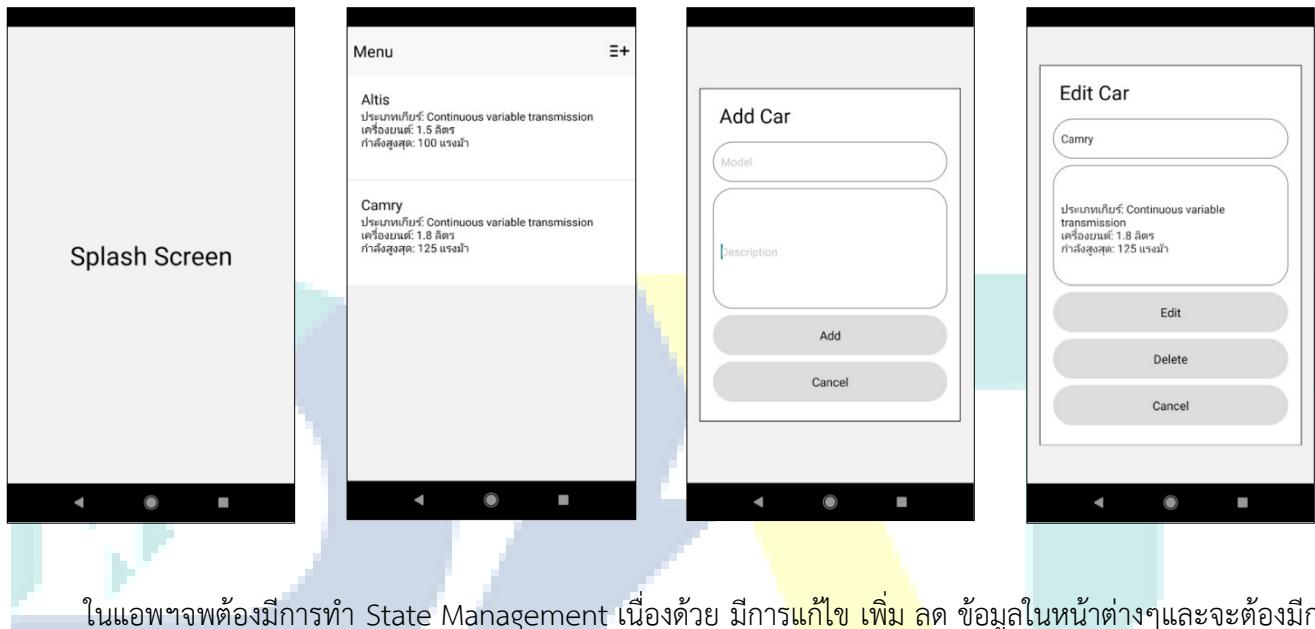
ผลลัพธ์ที่ได้

The screenshot shows the Firebase Firestore interface. On the left, there's a sidebar with a project named 'myproject-b75da'. Under 'Account', there's a 'cities' collection, which is currently selected and expanded, indicated by a grey arrow icon next to its name. To the right of the sidebar, the main area displays the 'cities' collection. It shows four documents: 'BJ', 'LA', 'SF', and 'TOK'. The 'BJ' document is highlighted with a yellow background, suggesting it has been deleted. The other three documents ('LA', 'SF', and 'TOK') are visible below it.

Thailand

ตัวอย่างการใช้งาน Google Cloud Firestore

ตัวอย่างการใช้งาน GCF โดยเราจะทำการสร้างแอพฯเพื่อเชื่อมต่อกับ Firebase โดยจำลองการเพิ่มข้อมูลและแสดงข้อมูลของรถพร้อมทั้งแสดงรายละเอียดในลักษณะของลีสข้อมูลโดยจะประกอบไปด้วย 4 หน้า(Screen) ได้แก่ SplashScreen, Menu Screen, AddCar Screen, EditCar Screen ทั้ง 4 หน้าเชื่อมต่อด้วยการทำ Navigation ในลักษณะของ Stack Navigator โดยรายละเอียดของแต่ละหน้าแสดงด้านล่าง



ในแอพฯจะพึ่งมีการทำ State Management เนื่องด้วย มีการแก้ไข เพิ่ม ลด ข้อมูลในหน้าต่างๆและจะต้องมีการอัปเดตข้อมูลในทุกๆหน้าให้เป็นปัจจุบัน โดยขั้นตอนในการสร้างแอพพลิเคชั่นมีดังนี้

- สร้างหน้า(Screen)แอพฯ ขั้นแรกเราจะต้องทำการหน้าแอพฯเสียก่อนโดยแนะนำให้ทำในลักษณะ Class Component เพื่อจัดการ State Management
- เชื่อมหน้า ขั้นต่อมาเราจะต้องทำการเชื่อมต่อ ลิงค์ หน้าไปยังหน้าอื่นๆด้วยการทำ Navigation
- สร้าง State Management ขั้นต่อมาเราจะต้องทำการ State และทำการเชื่อมต่อ(Connect) ไปยังทุกๆหน้าของแอพฯที่เราสร้าง
- เขียนโค้ดในส่วน Business logic ขั้นต่อมาให้เราทำการเขียนโค้ดในการเพิ่ม ลบ และแก้ไข โดยใช้เฉพาะ State เท่านั้น
- เชื่อมต่อกับฐานข้อมูล ขั้นสุดท้ายให้เราทำการเชื่อมต่อฐานข้อมูลตาม Business logic ที่เราได้เขียนไว้ก่อนหน้านี้

โดยตัวอย่างโปรแกรมสามารถดูได้จากลิงค์ด้านล่าง https://github.com/valeeprakhon/react-native-class-08_post

บทที่ 9 การเรียกใช้บริการ Google Cloud Firestore แบบ Real-Time

Google Cloud Firestore ในโหมด Real-Time-Database

ในบทก่อนหน้านี้เราได้ทำการใช้งาน Google Cloud Firestore เพื่อสร้างฐานข้อมูลและได้ลองทำการเพิ่ม ลบ แก้ไข ข้อมูลลงในฐานข้อมูลไปบ้างแล้ว ในบทนี้เราจะลองทำการใช้งาน GCF ในลักษณะ Real-Time Database(RTD) โดย RTD คือฐานข้อมูลที่สามารถติดตามดูการเปลี่ยนแปลงข้อมูลหรือเหตุการณ์ต่างๆที่เกิดขึ้นกับฐานข้อมูลได้ทันทีซึ่งการติดตามการเปลี่ยนแปลงนี้เราจะใช้ Application Programming Interface(API) ที่คอยตรวจสอบการเปลี่ยนแปลงของข้อมูลในฐานข้อมูล(data synchronization) และค่อยรายงานผลมาบ้างแอพฯของเรา ทำให้เราสามารถทราบถึงการเปลี่ยนแปลงของข้อมูลในฐานข้อมูลอยู่เสมอ

การตรวจจับการเปลี่ยนแปลงในฐานข้อมูลแบบ Real-time

การตรวจจับการเปลี่ยนแปลงเราจะใช้เมธอด `onSnapshot()` เพื่อดูการเปลี่ยนแปลงทุกสิ่งทุกอย่างที่เกิดขึ้นกับฐานข้อมูลของเราตามตัวอย่างโค้ดด้านล่างจะทำการเฝ้าดูการเปลี่ยนแปลงของเอกสารที่มีอิดี(Document ID) เท่ากับ SF เมื่อมีการแก้ไขข้อมูล เพิ่ม ลบ ก็จะทำการส่ง Document ใหม่กลับมายังแอพฯ

```
db.collection("cities").doc("SF")
  .onSnapshot(function(doc) {
    console.log("Current data: ", doc.data());
});
```

ทั้งนี้ในการเฝ้าดูการเปลี่ยนแปลงในเอกสารหากมีเอกสารอื่นๆเกิดการเปลี่ยนแปลงจะไม่ส่งผลถึงแอพฯของเราเนื่องจากการเราได้กำหนดอิດที่เราเฝ้ามองไว้หากต้องการเฝ้าดูทั้ง Collection โดยเฝ้าดูเอกสารที่มีข้อมูลบางอย่างที่เราต้องการ เราสามารถใช้ชีวิติการกรองข้อมูลด้วยเมธอด `where()` ที่อธิบายไปก่อนหน้านี้ โดยผลลัพธ์ที่ได้จะมีลักษณะเป็นชุดข้อมูลที่มีหลายๆ Document โดยตัวอย่างดังโค้ดด้านล่าง

```
db.collection("cities").where("state", "==", "CA")
  .onSnapshot(function(querySnapshot) {
    var cities = [];
    querySnapshot.forEach(function(doc) {
      cities.push(doc.data().name);
    });
    console.log("Current cities in CA: ", cities.join(", "));
});
```

การผู้ดูแลการเปลี่ยนแปลงของข้อมูลในฐานข้อมูลนักจากจะดูการเปลี่ยนแปลงทุกอย่างได้แล้วเรียบง่ายสามารถเลือก
ผู้ดูแลเฉพาะบางเหตุการณ์ที่เกิดขึ้นได้ซึ่งเหตุการณ์ที่เกิดขึ้นกับ Document สามารถเกิดขึ้นได้ 3 ลักษณะคือ การเพิ่มข้อมูล,
การลบข้อมูล และ การแก้ไขข้อมูล โดยเราสามารถเขียนโค้ดเพื่อแยกตามเหตุการณ์ได้ดังตัวอย่างด้านล่าง

```
db.collection("cities").where("state", "==", "CA")
  .onSnapshot(function(snapshot) {
    snapshot.docChanges().forEach(function(change) {
      if (change.type === "added") {
        console.log("New city: ", change.doc.data());
      }
      if (change.type === "modified") {
        console.log("Modified city: ", change.doc.data());
      }
      if (change.type === "removed") {
        console.log("Removed city: ", change.doc.data());
      }
    });
  });
});
```

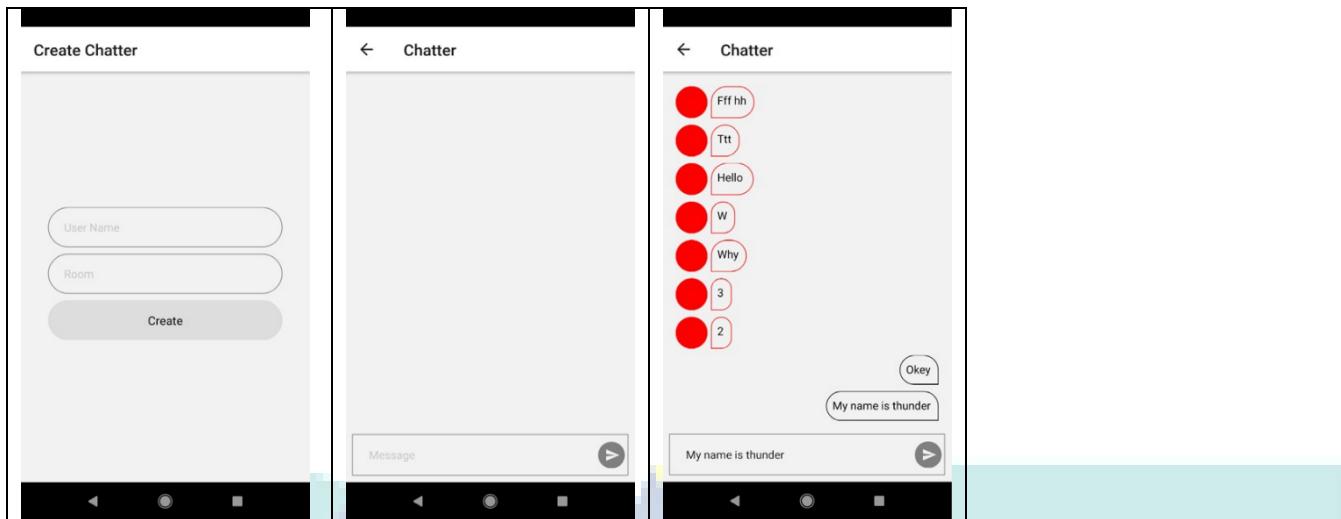
การยกเลิกการผู้ดูแลการเปลี่ยนแปลงหลังจากที่ทำงานเสร็จสิ้นเป็นเรื่องที่สำคัญ เนื่องจากตลอดเวลาที่แอพฯ เรา^{การทำงานก็}ผู้ดูแลการเปลี่ยนแปลงตลอดเวลาการทำให้ลืมเปลี่ยนทรัพย์กรของโทรศัพท์ได้ โดยการยกเลิกการผู้ดูแลสามารถกระทำ
ได้ตามตัวอย่างด้านล่าง

```
var unsubscribe = db.collection("cities")
  .onSnapshot(function () {
    // Respond to data
    // ...
  });
// Later ...
// Stop listening to changes
unsubscribe();
```

Digital Academy Thailand

ตัวอย่างแอพพลิเคชัน

ตัวอย่างแอพพลิเคชันแชท(Chat) เราจะใช้งาน GCF ในโหมด Real-Time ในการรับส่งข้อความของเราและคุ้มสันหนาโดยมีลักษณะตามตัวอย่างด้านล่าง



ในการออกแบบฐานข้อมูลเราจะสร้าง Collection ชื่อ Message เพื่อเก็บ Document ที่เกี่ยวกับข้อมูลการแซทซึ่งประกอบด้วย

Date คือ เวลาที่ส่ง Message

Room คือ ห้องที่ใช้ในการสนทนา

Sender คือ ผู้ส่ง

Text คือข้อความผู้ส่ง

รายละเอียดฐานข้อมูลตามภาพด้านล่าง

A screenshot of the Firebase console showing the "Message" collection. On the left, there's a sidebar with "myproject-b75da" and sections for "Start collection", "Account", "Friends", "Message", and "cities". The "Message" section is expanded, showing a list of document IDs. One document is selected, showing its details: "date: September 15, 2020 at 12:26:55 AM UTC+7", "room: "Test\"", "sender: "Test\"", and "text: "OK\". The entire interface is overlaid with a decorative pattern of blue and yellow triangles.

การเขียนโปรแกรมเพื่อเข้มต่อฐานข้อมูลเราจะแบ่งเป็นสองแบบคือการดึงข้อมูลประวัติการแชทเก่าและการเพ้าดู การแชทใหม่เมื่อเปิดโปรแกรมมาเราจะทำการดึงประวัติการแชทเก่าขึ้นมาแสดงก่อน ตัวอย่างโค้ดด้านล่าง

```
getMessage (room, getSuccess, getUnsuccess) {
  this.db.collection ('Message') .where ('room', '==', room)
  .get()
  .then(function (querySnapshot) {
    getSuccess (querySnapshot);
  })
  .catch (function (error) {
    getUnsuccess (error);
  });
}
```

```
firestore.getMessage (this.room, this.getSuccess, this.unsuccess);

getSuccess=(querySnapshot)=>{
  let mes=[];
  querySnapshot.forEach(function (doc) {
    mes.push(doc.data());
  });
  this.setState ({messages:this.state.messages.concat(mes)});
}
```

เมื่อทำการดึงประวัติการแชทเก่าเสร็จแล้วเราจะทำการเพ้าร้อการเข้ามาของ Document ที่มี Room ที่เราแชทอยู่ โดยทำการกรองข้อมูลด้วยเมธอด where('room', '==', this.room) และทำการเลือกอีเว้นที่เกิดขึ้นกับ document เนื่องจาก การเพิ่ม(Add) Document เท่านั้น ตัวอย่างโค้ดด้านล่าง

```
listeningMessage (room, listeningSuccess, listeningUnsuccess) {
  this.db.collection ('Message') .where ('room', "==" , room)
  .onSnapshot (function (snapshot) {
    snapshot.docChanges () .forEach (function (change) {
      if (change.type === "added") {
        listeningSuccess (change.doc);
      }
    });
  }, function (error) {
    listeningUnsuccess (error);
  });
}
```

```
firestore.listeningMessage (this.room, this.listeningSuccess, this.unsuccess);

listeningSuccess=(doc)=>{
  console.log(doc)
  this.setState ({messages:this.state.messages.concat (doc.data())})
}
```

ตัวอย่างโค้ดทั้งหมด https://github.com/valeeprakhon/react-native-class-09_post

บทที่ 10 การเรียกใช้บริการ Google Authentication

Google Authentication คืออะไร

แอพฯ ส่วนใหญ่จำเป็นต้องมีส่วนยืนยันตัวตนของผู้ใช้งาน การทราบข้อมูลประจำตัวของผู้ใช้งานทำให้แอพฯ สามารถบันทึกข้อมูลต่างๆ จากผู้ใช้งานลงในฐานข้อมูลได้อย่างปลอดภัยและเพิ่มความเป็นส่วนตัวแบบเดียวกันในอุปกรณ์ทั้งหมดของผู้ใช้งาน Firebase Authentication คือ SDK ที่ให้บริการแบบ Back-end มีลักษณะที่ใช้งานง่ายพร้อมกับไลบรารีและ API สำหรู่รูปเพื่อตรวจสอบผู้ใช้กับแอพฯ ของเรา รองรับการยืนยันตัวตนของผู้ใช้งานโดยใช้รหัสผ่าน, รองรับการยืนยันตัวตนผ่านทางหมายเลขโทรศัพท์ หรือแม้กระทั้งการยืนยันตัวตนของผู้ใช้งานผ่าน Social network ต่างๆ เช่น Google, Facebook และ Twitter และอื่น ๆ โดยมีคุณลักษณะดังนี้

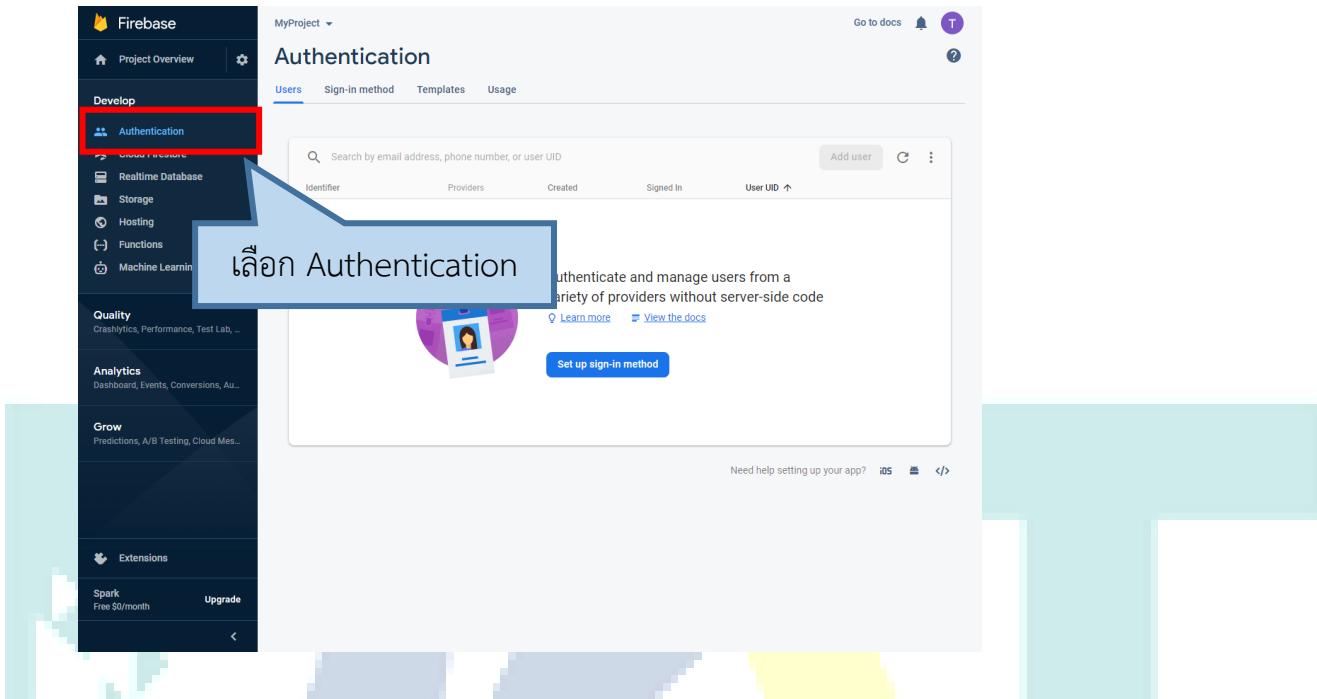
คุณลักษณะ	คำอธิบาย
Email and password based authentication	สามารถยืนยันผู้ใช้งานด้วยที่อยู่อีเมลและรหัสผ่าน โดยมีวิธีการสร้างและจัดการผู้ใช้งานโดยใช้อีเมลและรหัสผ่านในการลงทะเบียนเข้าใช้งานและมีระบบจัดการการรีเซ็ตอีเมลรหัสผ่าน
Federated identity provider integration	สามารถยืนยันผู้ใช้งานผ่าน Social Network ด้วยบัญชี Google, Facebook, Twitter และ GitHub
Phone number authentication	รับรองความการยืนยันความถูกต้องของผู้ใช้งานผ่านข้อความ SMS ในโทรศัพท์
Custom auth system integration	สามารถเชื่อมต่อระบบลงชื่อเข้าใช้งานที่มีอยู่ของแอพฯ ร่วมกับฐานข้อมูลเรียลไทม์ของ Firebase และบริการอื่น ๆ ได้
Anonymous auth	รองรับการยืนยันตัวตนโดยไม่จำเป็นต้องให้ผู้ใช้งานลงชื่อเข้าใช้ก่อนโดยสร้างบัญชีขึ้นมาแบบไม่ระบุตัวตน หากผู้ใช้งานประสงค์จะลงทะเบียนภายหลังก็สามารถคืออัปเกรดบัญชีได้

ในการลงชื่อผู้ใช้งานผ่านระบบเบินยันตัวตนของแอพฯ เราจะต้องได้รับข้อมูลการรับรองการตรวจสอบสิทธิ์เสียก่อน การตรวจสอบสิทธิ์สามารถใช้อีเมลและรหัสผ่านของผู้ใช้ส่งไปยัง Firebase Authentication Service ที่ทำงานในของ Back-end service จากนั้นจะตอบกลับมาอย่างแอพฯ ของเราในลักษณะคือการยืนยันตัวตนเสร็จสมบูรณ์หรือการยืนยันล้มเหลวโดยการใช้งานจะมีขั้นตอนตามรายลับอีกด้านล่าง

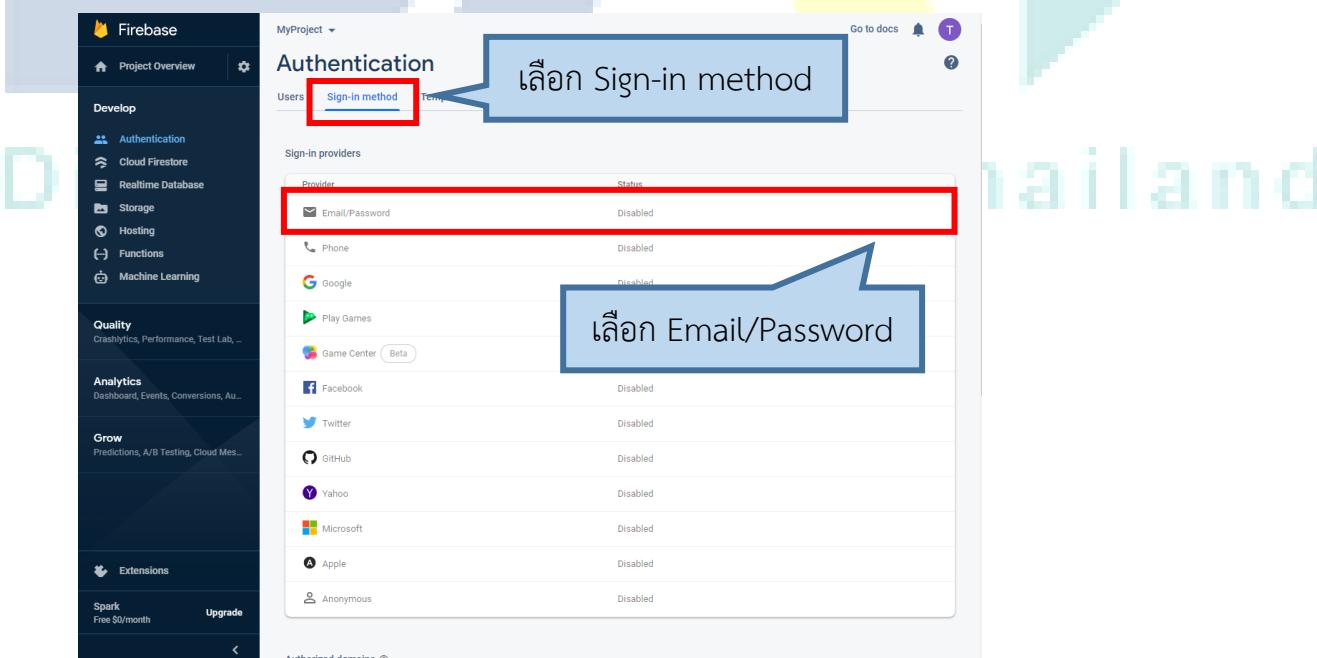
การตั้งค่าโปรเจกเพื่อใช้งาน Google Authentication

ก่อนการใช้งาน Google Authentication เราจำเป็นจะต้องตั้งค่าใน Firebase Project เสียก่อนซึ่งการตั้งค่านี้เรา จะต้องทำการอนุญาตวิธีการยืนยันตัวบุคคล ซึ่งในที่นี้จะใช้งานการยืนยันผ่านอีเมลและรหัสผ่านโดยมีวิธีดังนี้

1. เปิด Firebase และเลือกเมนู Authentication



2. เลือก Sign-in method และเลือก Email/Password



3. ทำการเปิดการใช้งานการยืนยันด้วย Email/Password และกด Save

Email/Password

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. [Learn more](#)

Enable

Email link (passwordless sign-in)

Enable

Cancel Save

4. เมื่อนำมาแล้วจะแสดงข้อความ Enabled

Provider	Status
✉ Email/Password	Enabled
📞 Phone	Disabled
🌐 Google	Disabled
🎮 Play Games	Disabled
👾 Game Center <small>Beta</small>	Disabled
🌐 Facebook	Disabled
🐦 Twitter	Disabled
🐙 GitHub	Disabled
📧 Yahoo	Disabled
🌐 Microsoft	Disabled
🍎 Apple	Disabled
👤 Anonymous	Disabled

Digital Academy Thailand

การใช้งาน Google Authentication เพื่อสร้างบัญชีแบบ Manual

การใช้งานแบบ Google Authentication แบบ Manual เป็นการใช้งานโดยทำการเพิ่มอีเมล์และรหัสผ่านด้วยตนเองโดยทำการเขียนโปรแกรมเพื่อทดสอบการยืนยันตัวบุคคลโดยมีรายละเอียดดังนี้

1. กด Add user

The screenshot shows the Firebase console's Authentication section. On the left sidebar, under 'Develop', 'Authentication' is selected. The main area shows a table with columns: Identifier, Providers, Created, Signed In, and User UID. A red box highlights the 'Add user' button at the top right of the table. A blue callout bubble points to this button with the text 'กด Add user'.

2. พิมพ์อีเมล์และรหัสผ่านที่ต้องการ เสร็จแล้วกด Add user

The screenshot shows a modal dialog titled 'Add an Email/Password user'. It has two input fields: 'Email' containing 'tamnuwat.va@ku.th' and 'Password' containing '123456789'. At the bottom are 'Cancel' and 'Add user' buttons.

3. เมื่อเพิ่มอีเมล์และรหัสผ่านเสร็จแล้วจะแสดงรายละเอียดที่ถูกเพิ่มเข้ามา

The screenshot shows a table of users. The first row contains the newly added user: Identifier 'tamnuwat.va@ku.th', Providers '✉️', Created 'Sep 15, 2020', Signed In '2oYixGLCz8TLuyrP33ZedAzHxu83', and User UID. The table includes a search bar at the top and pagination controls at the bottom.

Identifier	Providers	Created	Signed In	User UID
tamnuwat.va@ku.th	✉️	Sep 15, 2020	2oYixGLCz8TLuyrP33ZedAzHxu83	

เมื่อทำการทดลองเพิ่มค่าอีเมล์และรหัสผ่านเมื่อสักครู่นี้โดย ตัวอย่างโค้ดแสดงด้านล่าง

```
firebase.auth().createUserWithEmailAndPassword(email, password)
  .catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

เมื่อทำการยืนยันตัวบุคคลเสร็จเรียบร้อยแล้วจะไม่มีการ Return ค่ากลับมายังฟังก์ชันเดิมแต่จะมีการอัพเดตค่าผ่านฟังก์ชันตัวแปร currentUser เราสามารถเรียกดูสถานะของค่านี้ได้โดยตรงหรือจะทำการเฝ้าดู(Listening)การเปลี่ยนแปลงค่านี้ได้ผ่านเมธอด onAuthStateChanged() ตัวอย่างโค้ดแสดงด้านล่าง

```
var user = firebase.auth().currentUser;

firebase.auth().onAuthStateChanged(function(user) {
  if (user) {
    console.log(user);
  } else {
    console.log('No user');
  }
});
```

โดยผลลัพธ์เมื่อสำเร็จแสดงด้านล่าง

```
{uid:"2oYixGLCz8TLuyrP33ZedAzHxu83",displayName:null,photoURL:null,email:"tamnuwat.v@ku.th",emailVerified:false,phoneNumber:null,isAnonymous:false,tenantId:null,providerData:[...],apiKey:"AIzaSyBRwJoQOqYr4CDTb5O3hGHuWKqYR5TNKck",appName:"[DEFAULT]",authDomain:null,stsTokenManager:{...},redirectEventId:null,lastLoginAt:"1600184761872",createdAt:"1600171816052",multiFactor:{...}}
```

เมื่อเราทำการปิดและเปิดแอพฯของเรามีอีกรึ่งระบบก็ยังคงทำการยืนยันตัวบุคคลได้อยู่เป็นเรื่องที่สะดวกสำหรับผู้ใช้งานที่ไม่ต้องยืนยันตัวบุคคลใหม่แต่ถ้าเราต้องการออกจากออกจากการยืนยันบุคคล(Sign Out) เพื่อให้ผู้ใช้คนใหม่สามารถใช้งานแอพฯนี้ได้ ตัวอย่างโค้ดแสดงด้านล่าง

```
firebase.auth().signOut().then(function() {
  // Sign-out successful.
}).catch(function(error) {
  // An error happened.
});
```

การเขียนโปรแกรมเพื่อสร้างบัญชี

ในแอพฯ ส่วนใหญ่จะมีระบบสมัครสมาชิกด้วยอีเมล์และรหัสผ่านโดยผู้ใช้งานเป็นผู้กำหนดเอง ซึ่งเราจะต้องทำการเขียนโค้ดเพื่อสร้างส่วนสมัครสมาชิกแล้วสร้างบัญชีไปยัง Firebase authentication โดยใช้メธอด createUserWithEmailAndPassword() ตัวอย่างโค้ดแสดงด้านล่าง

```
firebase.auth().createUserWithEmailAndPassword(email, password)
  .catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

เนื่องด้วยการสร้างบัญชีส่วนใหญ่จะมีข้อมูลที่สำคัญต่างๆ ของผู้ใช้งานนอกเหนือจากอีเมล์ที่ใช้สร้างบัญชี เช่น รูปภาพ, ชื่อที่แสดง(User Name) และหมายเลขโทรศัพท์ ทั้งนี้หากเราต้องการเก็บข้อมูลอื่นๆ นอกเหนือจากนี้ เราจะต้องทำการสร้างฐานข้อมูลเพื่อกีบเพิ่มเติม

```
var user = firebase.auth().currentUser;

user.updateProfile({
  displayName: "Your Display Name",
  photoURL: "your photo URL",
  phoneNumber: "your phone number"
}).then(function() {
  // Update successful.
}).catch(function(error) {
  // An error happened.
});
```

การสร้างระบบกู้คืนรหัสผ่าน

การลืมรหัสผ่านแอพฯ เป็นเรื่องที่เกิดขึ้นได้เสมอระบบกู้คืนรหัสผ่านยอมเป็นสิ่งที่จำเป็นสำหรับทุกๆ แอพฯ ที่มีการล็อกคินเสมอ การกู้รหัสผ่านสามารถทำได้หลายวิธี เช่น การกู้ผ่านเบอร์โทรศัพท์ หรือ การกู้ผ่านอีเมล์ ในกรณีที่เราสมัครสมาชิกด้วยอีเมล์ การกู้รหัสผ่านย่อมต้องผ่านอีเมล์ โดยการกู้ข้อมูลผ่านอีเมล์ด้วย Firebase Authentication จะใช้เมธอด sendPasswordResetEmail() ตัวอย่างโค้ดแสดงด้านล่าง

```
var auth = firebase.auth();
var emailAddress = "Your Email";

auth.sendPasswordResetEmail(emailAddress).then(function() {
  // Email sent.
}).catch(function(error) {
  // An error happened.
});
```

ตัวอย่างอีเมลที่ถูกส่งมา

Hello,

Follow this link to reset your project-1041757860081 password for your tamnuwat_val@gmail.com account.

https://myproject-b75da.firebaseio.com/_auth/action?mode=resetPassword&oobCode=9l1Z8ccvpghRhwKmu1fxYLJDpeWIN4kPLuu5sQgeTjgAAAF0krJ-tA&apiKey=AlzaSyBRwJoQOqYr4CDTb5O3hGHuWKqYR5TNKck&lang=en

If you didn't ask to reset your password, you can ignore this email.

Thanks,

Your project-1041757860081 team

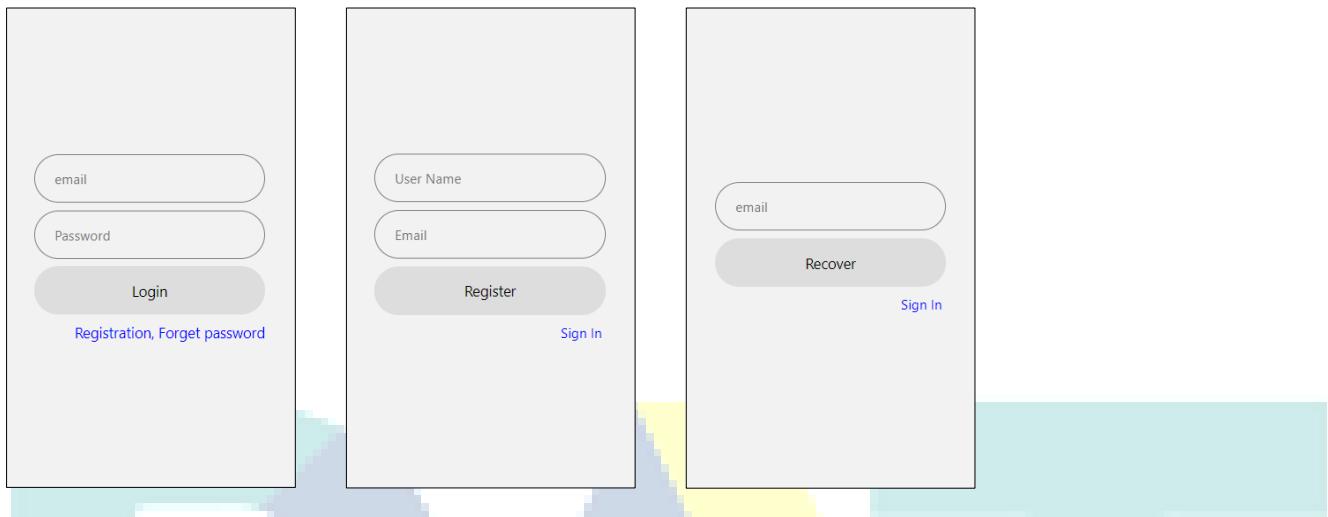
ความสามารถแก้ไขรายละเอียดอีเมลที่ส่งมาได้ด้วยการเข้าไปแก้ไขในส่วนของ Templates ที่ Firebase ได้สร้างไว้ตามภาพด้านล่าง

The screenshot shows the Firebase console's Authentication section with the 'Templates' tab selected. On the left, there's a sidebar with various project settings like Project Overview, Develop, Quality, Analytics, and Grow. The main area displays a 'Templates' card for 'Email'. Under the 'Email' card, the 'Password reset' template is selected. It shows the template content with placeholders for Sender name (not provided), From (noreply@myproject-b75da.firebaseio.com), Reply to (noreply), Subject (Reset your password for project-1041757860081), and Message (Hello, Follow this link to reset your project-1041757860081 password for your %EMAIL% account. https://myproject-b75da.firebaseio.com/_auth/action?mode=resetPassword&oobCode=<code>). Below the message, it says 'If you didn't ask to reset your password, you can ignore this email.' and 'Thanks, Your project-1041757860081 team'. At the bottom of the card, it says 'Template language English' and has a pencil icon for editing.

Digital Academy Thailand

ตัวอย่างแอพพลิเคชันที่เชื่อมต่อกับ Google Authentication

ตัวอย่างแอพพลิเคชันเป็นระบบสมัครสมาชิกซึ่งประกอบไปด้วยหน้าล็อกอิน, หน้าสมัครสมาชิก และหน้ากู้คืนรหัสผ่าน แสดงดังภาพด้านล่าง



1. หน้าล็อกอิน

เราได้สร้างเมธอด `getCurrentUser()` เพื่อค่อยເຝັດສະນະຂອງຜູ້ໃຊ້ຈາກຍັງໄມ່ມີກາລືອກອີນຈະໄມ່ແສດງຂໍ້ຄວາມ ໂດຍແຕ່ດ້າທາກລືອກອີນແລ້ວຈະແສດງຂໍ້ມູນຕ່າງໆຂອງຜູ້ໃຊ້ຈານ

```
componentDidMount () {  
    auth.getCurrentUser(this.getCurrentUserSuccess);  
}  
getCurrentUserSuccess=(user)=>{  
    console.log(user)  
}
```

```
getCurrentUser=(getSuccess)=>{  
    this.auth.onAuthStateChanged(function (user) {  
        getSuccess(user);  
    });  
}
```

เมื่อมີກາລືອກອີນດ້ວຍກາຣໂກອີເມວີແລ້ວຮ້າສິ່ງເຂົ້າມາເຮົາຈະທຳການສ່າງຂໍ້ມູນເທົ່ານີ້ໄປຢັ້ງເມືອດ `signIn()` ຍາກລືອກອີນສໍາເສົ່າງພຶກໍ່ນີ້ `getCurrentUserSuccess()` ຈະແສດງຂໍ້ມູນຜູ້ໃຊ້ຈານ

```
signIn=(email,password,unsuccess)=>{  
    this.auth.signInWithEmailAndPassword(email,password)  
    .catch(function(error){  
        unsuccess(error);  
    });  
}
```

2. หน้าสมัครสมาชิก

ในส่วนนี้จะเป็นส่วนที่ผู้ใช้งานสามารถสมัครสมาชิกได้เมื่อผู้ใช้งานทำการกรอกข้อมูลอีเมล์และรหัสผ่านที่ต้องการแล้วเราจะทำการเรียกเมธอด createUser() เพื่อทำการสร้างบัญชีผู้ใช้งาน

```
createUser=(email,password,unsuccess)=>{
  this.auth.createUserWithEmailAndPassword(email,password)
  .catch(function(error) {
    unsuccess(error);
  })
}
```

3. หน้ากู้รหัสผ่าน

ในการณ์ที่ผู้ใช้งานลืมรหัสผ่านเราก็สามารถทำการกู้รหัสผ่านได้โดยให้ผู้ใช้งานกรอกอีเมล์ที่ใช้ในการสมัครสมาชิกจากนั้นจะทำการส่งอีเมล์ไปยังเมธอด recoverAccount() เมื่อเสร็จสมบูรณ์ Firebase authentication จะทำการส่งอีเมล์กู้รหัสคืนไปยังเมล์ที่สมัครเราก็สามารถกู้รหัสคืนได้

```
onRecover= ()=>{
  auth.recoverAccount(this.state.email,this.success,this.unsuccess);
}

unsuccess=(error)=>{
  console.log(error)
}

success=()=>{
  console.log("Email sent");
}
```

```
recoverAccount=(email,success,unsuccess)=>{
  this.auth.sendPasswordResetEmail(email)
  .then(function(){
    success(null);
  })
  .catch(function(error){
    unsuccess(error);
  });
}
```

ตัวอย่างโค้ดทั้งหมด <https://github.com/valeeprakhon/react-native-class-10-post>

บทที่ 11 การเตรียมความพร้อมในการอัพโหลดแอพฯ ขึ้นสโตร์

สโตร์(Store) คืออะไร

Store คือแหล่งรวม Application มากมายจากเป็นช่องทางที่ช่วยให้ผู้ใช้งานสามารถดาวน์โหลดได้จากโทรศัพท์ ตัวเครื่องได้ทันที ภายใน Store มีการแบ่งหมวดหมู่แอพฯต่างๆไว้อย่างชัดเจน สามารถเลือกดาวน์โหลดทั้งแบบฟรีและเสียค่าใช้จ่าย รวมไปถึงหน้าแนะนำแอพฯยอดนิยมต่างๆไว้อีกด้วย โดยสโตร์ที่ให้บริการและเป็นที่นิยมในปัจจุบันได้แก่เพย์สโตร์ (PlayStore) และแอปเปิลสโตร์(Appple Store) การอัพโหลดแอพฯที่พัฒนาขึ้นสโตร์สิ่งจะต้องทำการสร้าง(Build)แอพฯให้อยู่ในรูปแบบที่เหมาะสมกับการอัพโหลดขึ้นสโตร์แต่ละสโตร์

การเตรียมความพร้อมในการ Build แอพฯ

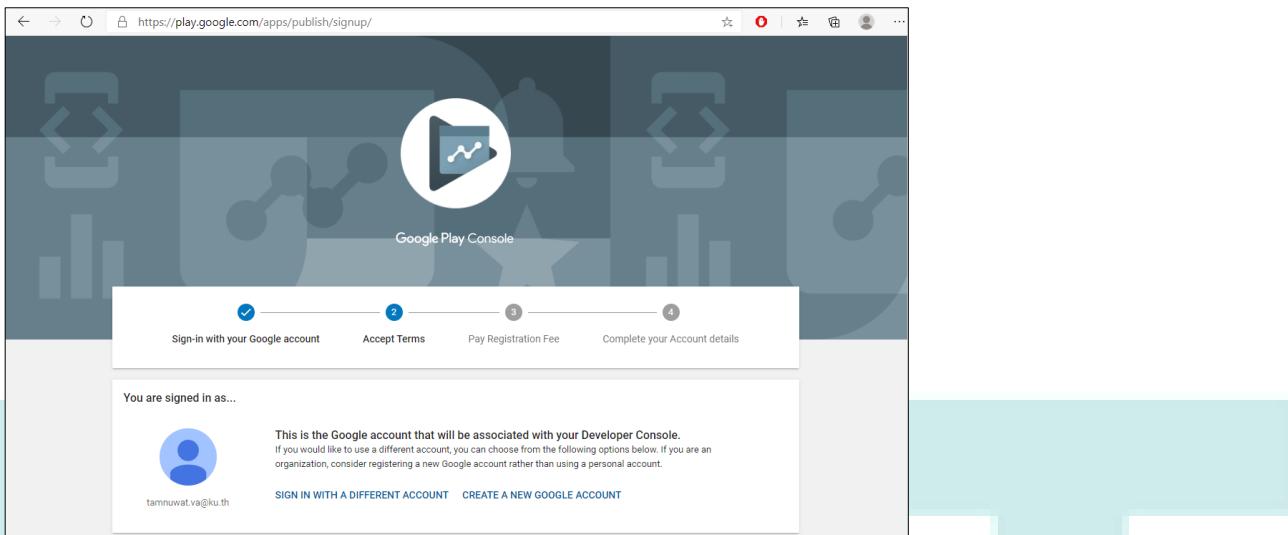
เมื่อสร้างแอพฯเสร็จสิ้นและทำการทดสอบจนไม่พบข้อผิดพลาดพร้อมที่จะเปิดให้บริการ โดยเราจะต้องทำการ สร้าง(Build) แอพฯให้อยู่ในรูปแบบพร้อมที่จะอัพโหลด การเตรียมความพร้อมก่อนการอัพโหลดเราจะต้องมีสิ่งสำคัญ 2 ลิ่งดังนี้

1. บัญชี Expo คือบัญชีที่เอาไว้เข้าใช้งาน Service ต่างของ Expo เช่น Snek, Build Service เป็นต้น เราสามารถสมัครสมาชิกผ่านลิงค์ <https://expo.io/> ได้แบบฟรี
2. บัญชีนักพัฒนา(Developer Account) คือบัญชีที่ผู้กับสโตร์ นักพัฒนาจำเป็นจะต้องสมัครสมาชิกก่อนจึงจะสามารถทำแอพฯอัพโหลดขึ้นสโตร์เพื่อเผยแพร่จำหน่ายจ่ายเงินได้โดยการสมัครสมาชิกแต่ละสโตร์

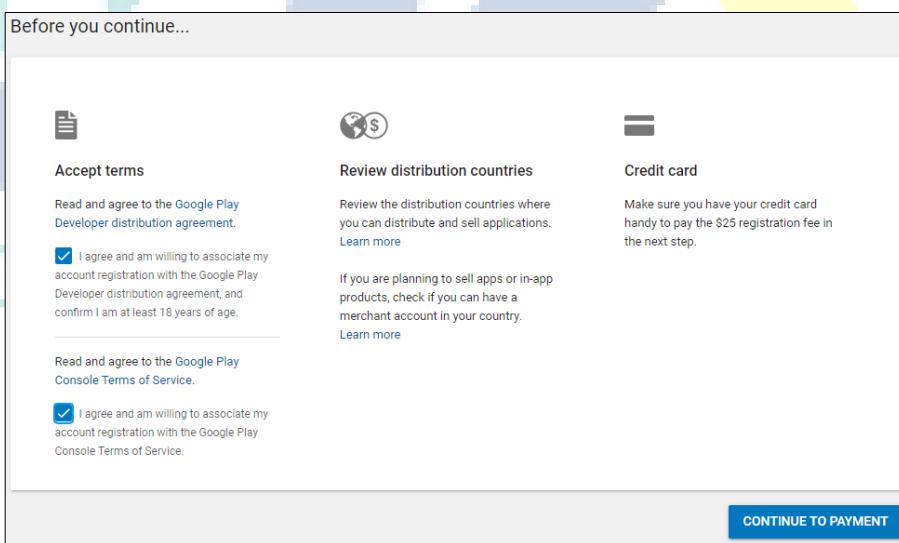
การสมัคร Google developer account

ในการสมัคร Google developer account เราจำเป็นจะต้องมี บัญชี Gmail เสียก่อนและทำการเตรียมค่าสมัครสมาชิก 25 USD ครั้งเดียวสามารถใช้ได้ตลอด วิธีการสมัครสามารถทำได้ดังนี้

1. ทำการล็อกอิน Google ด้วยบัญชี Gmail
2. เข้าไปยังเว็บไซต์ <https://play.google.com/apps/publish/signup/>



3. จากนั้นยอมรับข้อตกลงต่างๆ และกด



4. กรอกรายละเอียดเพื่อใช้งานการชำระเงิน

The screenshot shows a payment form for a 'Developer Registration Fee' of \$25.00. It includes fields for adding a credit or debit card, entering a card number (with a placeholder for logos like American Express, Discover, JCB, MasterCard, and Visa), and selecting a expiration date (MM / YY) and CVC. Error messages indicate that 'Card number is required' and 'Cardholder name is required'. Below the card input is a 'Billing address' field with a location pin icon. At the bottom, a note states: 'By continuing, you create a Google Payments account and agree to [Google Payments Terms of Service](#) and [Privacy Notice](#)'. A large blue 'BUY' button is centered at the bottom.

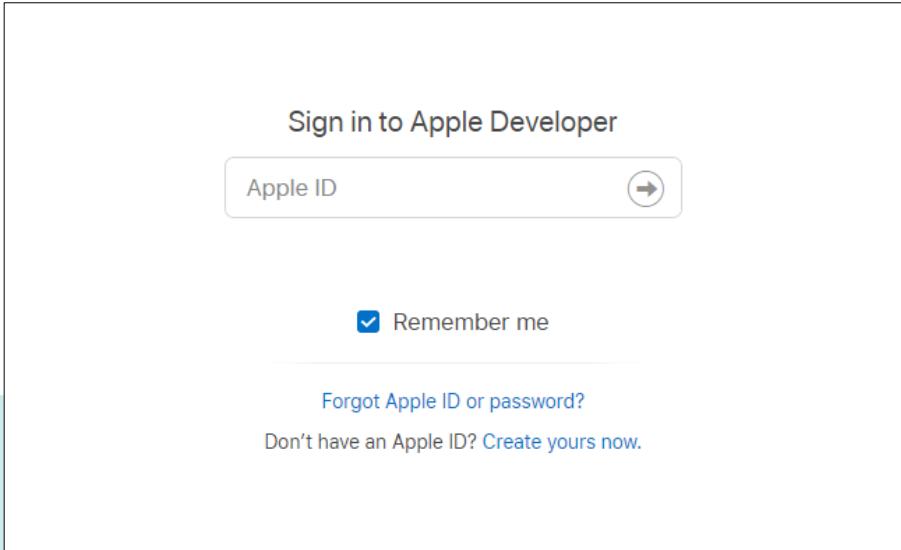
5. เมื่อชำระเงินเสร็จสิ้นจะเข้าสู่หน้าอัปโหลดแอพฯ กีว่าสมัครเสร็จสิ้น



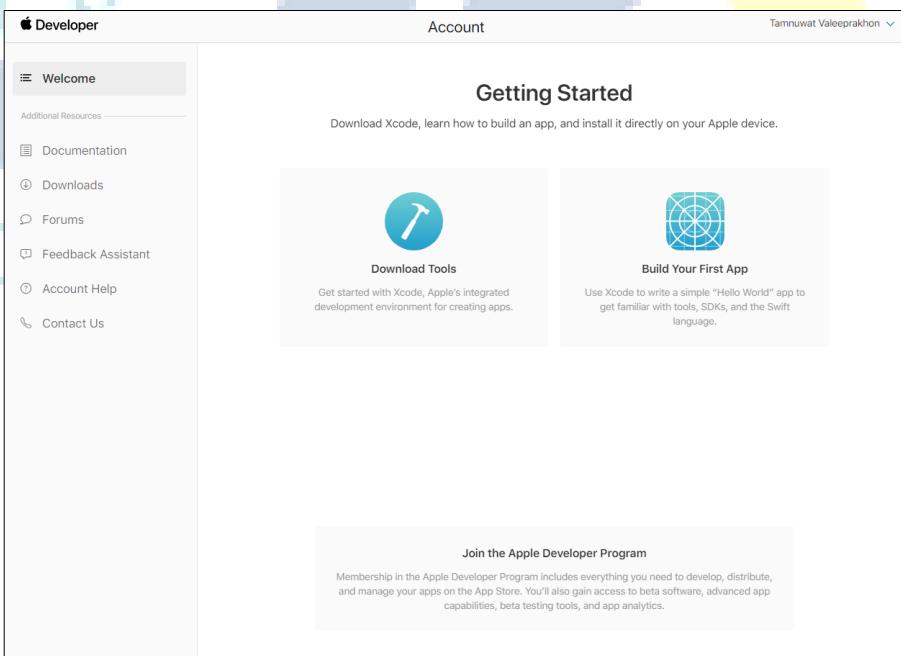
การสมัคร Apple developer account

ในการสมัคร Apple developer account เราจำเป็นจะต้องมี Apple ID เสียก่อนและว่างແຜ່ນการສັບສົນໃນນາມໂຄຣທັງນີ້ຕ້ອງຢ່າງນີ້ຈະທຳການສັບສົນໃນນາມປະເທດທີ່ຈະເສີຍຄ່າຮຽນຮ່າງປີ 99 USD ວິທີການສັບສົນສາມາດທຳໄດ້ດັ່ງນີ້

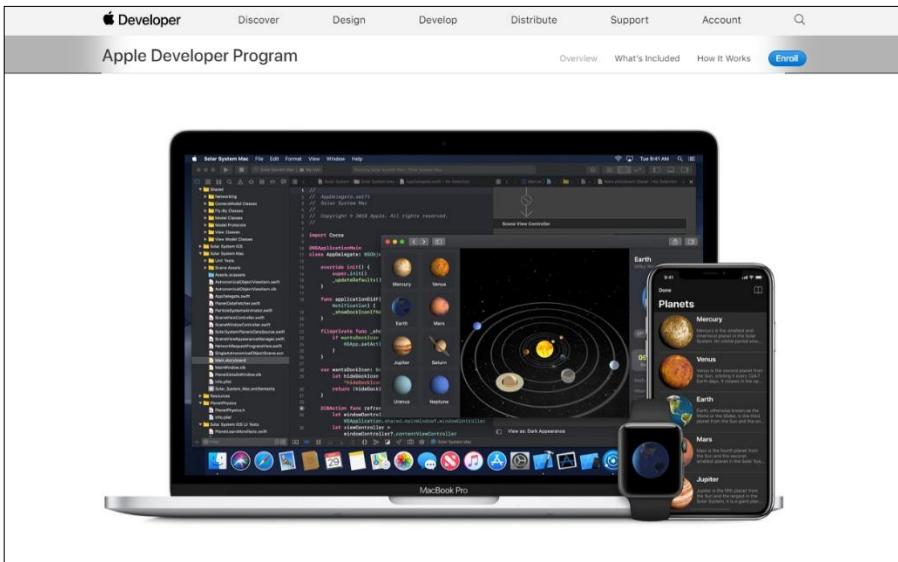
1. เข้าໄປຢັ້ງເວັບໄຊ໌ <https://developer.apple.com/account>
2. ທຳການກຣອກ AppleID ແລະ ຮຫ້ສຳຜ່ານ



3. ທຳການຍອມຮັບເຈື້ອນໄຂ



4. เลือก Enroll



5. ยอมรับข้อตกลงต่างๆจากนั้นกด Start Your Enrollment

If you're enrolling your organization, you'll need an Apple ID with [two-factor authentication](#) turned on, as well as the following to get started:

A D-U-N-S® Number
Your organization must have a D-U-N-S Number so that we can verify your organization's identity and legal entity status. These unique nine-digit numbers are assigned by Dun & Bradstreet and are widely used as standard business identifiers. You can check to see if your organization already has a D-U-N-S Number and request one if necessary. They are free in most jurisdictions. [Learn more](#)

Legal Entity Status
Your organization must be a legal entity so that it can enter into contracts with Apple. We do not accept DBAs, fictitious businesses, trade names, or branches.

Legal Binding Authority
As the person enrolling your organization in the Apple Developer Program, you must have the legal authority to bind your organization to legal agreements. You must be the organization's owner/founder, executive team member, senior project lead, or an employee with legal authority granted to you by a senior employee.

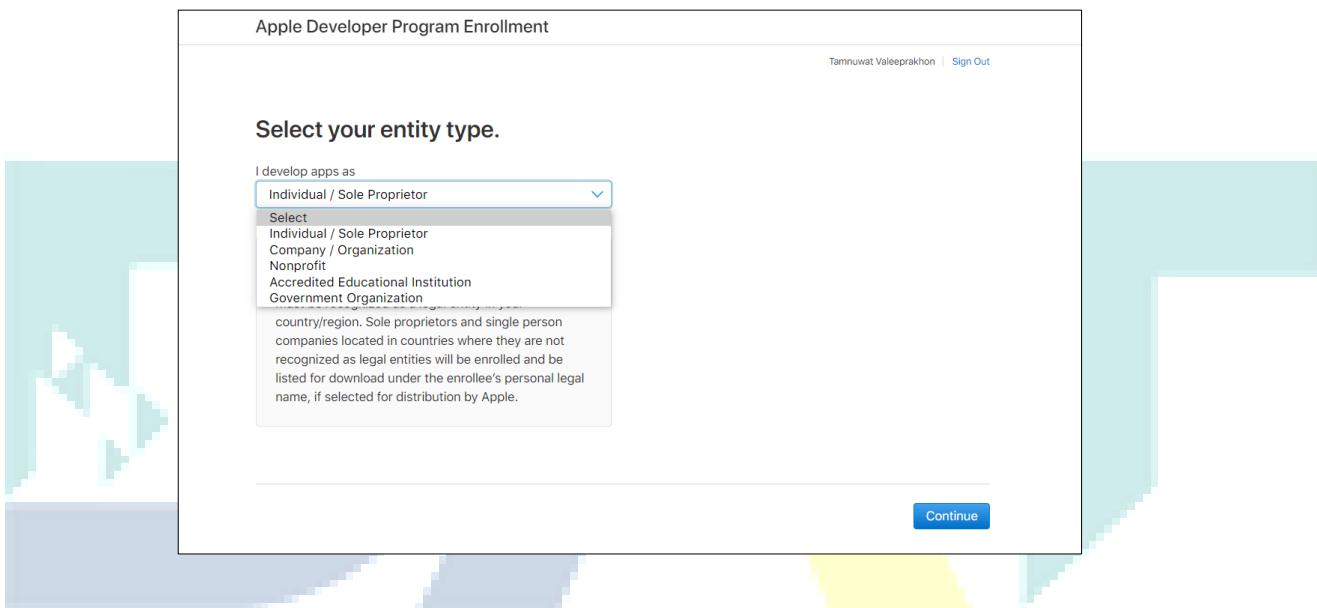
A Website
Your organization's website must be publicly available and the domain name must be associated with your organization.

[Start Your Enrollment](#)

6. เลือกประเภทการสมัครสมาชิกโดยแบ่งได้ 5 ประเภทดังนี้

- Individual / Sole Proprietor คือการสมัครสมาชิกในนามบุคคล
- Company / Organization คือการสมัครสมาชิกในนามบริษัท
- Nonprofit คือการสมัครสมาชิกในนามองค์กรที่ไม่แสวงหาผลกำไร
- Accredited Educational Institution คือการสมัครสมาชิกในนามหน่วยงานการศึกษา
- Government Organization คือการสมัครสมาชิกในนามหน่วยราชการต่างๆ

โดยมากการสมัครสมาชิกจะเป็นในนามบริษัท(Company / Organization) เนื่องจากการสมัครสมาชิก จะเสียค่าสมัครสมาชิกปีละ 99 USD ซึ่งจะเป็นภาระมากหากสมัครสมาชิกในนามบุคคล(Individual / Sole Proprietor)



7. ในการสมัครแบบ Organization ทางบริษัทจำเป็นต้องขอ DUNS (Data Universal Numbering System) ก่อน ซึ่ง เป็นตัวตนของนิติบุคคลบนเว็บการค้าโลก โดยเลือก Check now

• The Authority to Sign Legal Agreements

As the person enrolling your organization in a developer program, you must have the legal authority to bind your organization to legal agreements. You must be the organization's owner/founder, executive team member, senior project lead, or an employee with legal authority granted to you by a senior employee.

• A Website

Your organization's website must be publicly available and the domain name must be associated with your organization.

• A D-U-N-S® Number

Your organization must have a D-U-N-S® Number so that we can verify your organization's identity and legal entity status. These unique nine-digit numbers are assigned by Dun & Bradstreet and are widely used as standard business identifiers. You can check to see if your organization already has a D-U-N-S® Number and request one if necessary. They are free in most jurisdictions. [Check now >](#)

Continue

8. จักนี้นทำการกรอกข้อมูลรายละเอียดของบริษัทให้เรียบร้อย เมื่อทำการร้องขอ D-U-N-S Number จะใช้เวลาประมาณ 1-2 สัปดาห์

Look up your D-U-N-S Number

Before enrolling, look up your organization to see if you have a D-U-N-S Number. Dun & Bradstreet may have already assigned one to you. If your organization is not listed, you'll have the option to submit your information to D&B for a free D-U-N-S Number.

Organization Information

Country / Region

Country or region where your company is physically located. If you don't see your location listed, [contact us](#).

Please use Roman characters, as other character sets are not supported.

Legal Entity Name

Headquarters Address

Street Address

9. เมื่อได้รับหมายเลข D-U-N-S Number เรียบร้อยแล้วให้ทำการกรอกรายละเอียดเกี่ยวกับบริษัทพร้อมหมายเลข D-U-N-S ที่ได้รับ

Apple Developer Program Enrollment

Tell us about your organization.

Legal Entity Name

Include the entity type, such as Inc., LLC, GmbH, etc.

D-U-N-S® Number

In order for your company name to be listed on your App Store product page, your company must be recognized as a legal entity in your country/region. Sole proprietors and single person companies located in countries/regions where they are not recognized as legal entities will be enrolled and be listed for download under the enrollee's personal legal name, if selected for distribution by Apple.

Enter the characters in the image below. Switch to audio Try another

J6HJU

Letters are not case-sensitive.

Back Continue

10. จากนั้นจะเข้าสู่การชำระเงินผ่านบัตรเครดิต
11. จากนั้น Apple จะใช้เวลาในการตรวจสอบ โดยในระหว่างนี้จะมีการสอบถามรายละเอียดต่างๆเกี่ยวกับบริษัทชื่อ Apple 逼迫มาพูดคุยสอบถามรายละเอียดหรืออาจจะให้ส่งเอกสารทาง Email
12. เมื่อผ่านกระบวนการทั้งหมดแล้วเราจะได้รับอีเมลยืนยันจากทาง Apple การสมัครเสร็จสมบูรณ์

การสร้าง APK ไฟล์เพื่ออัปโหลดขึ้น Google Play Store

เมื่อทำการสมัครบัญชีนักพัฒนาและบัญชี Expo เรียบร้อยแล้ว เราจะทำการสร้างแพ็กเกจไฟล์เพื่อใช้สำหรับอัปโหลดขึ้น Google play store ซึ่งเราจะเรียกว่า APK (Android Package Kit) ซึ่ง APK จะมี 2 ลักษณะคือ

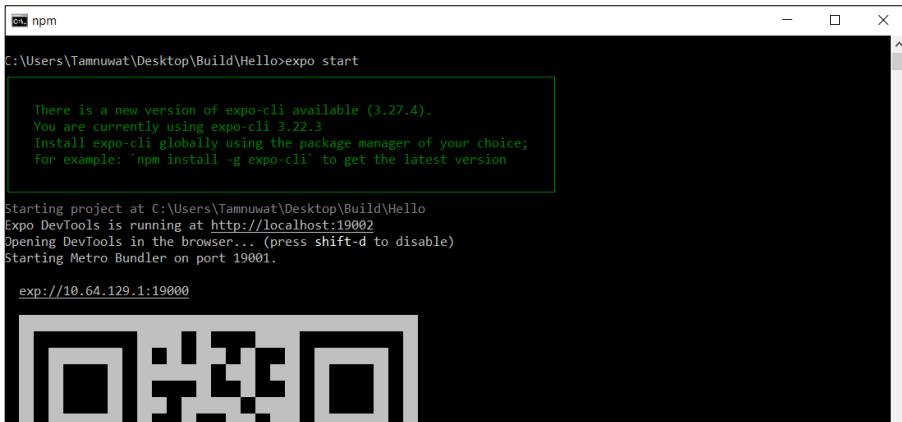
1. APK with keystore คือ แพ็กเกจไฟล์ที่มีการสร้าง keystore ฝังไว้อยู่ภายใน ส่วนใหญ่เรามักจะสร้างเมื่ออุปกรณ์ใน阶段 (Release) เพื่อที่จะอัปโหลดไปยังสโตร์
2. APK without keystore คือแพ็กเกจไฟล์ที่ไม่ได้มีการสร้าง ส่วนใหญ่เรามักจะสร้างเมื่ออุปกรณ์ในขั้นทดสอบเพื่อหาข้อผิดพลาด(Debug) สามารถนำไฟล์ไปติดตั้งในโทรศัพท์และรออยู่ได้และทำงานได้เหมือนดังเช่นดาวน์โหลดจากสโตร์

ในการสร้าง APK with keystore เราจำเป็นจะต้องกำหนดรายละเอียดของแพ็กเกจและเวอร์ชันของโค้ดให้เรียบร้อยในไฟล์ app.json เสียก่อนโดยมีรายละเอียดดังนี้

```
{
  "expo": {
    "name": "Your App Name",
    "icon": "./path/to/your/app-icon.png",
    "version": "1.0.0",
    "slug": "your-app-slug",
    "ios": {
      "bundleIdentifier": "com.yourcompany.yourappname",
      "buildNumber": "1.0.0"
    },
    "android": {
      "package": "com.yourcompany.yourappname",
      "versionCode": 1
    }
  }
}
```

วิธีการสร้างไฟล์ APK with keystore สามารถทำได้ดังนี้

1. ทำการรัน Metro Bundler ขึ้นมาเพื่อตอนที่เราทดสอบแอพฯ ปกติ



```
npm
C:\Users\Tamnuwat\Desktop\Build\Hello>expo start

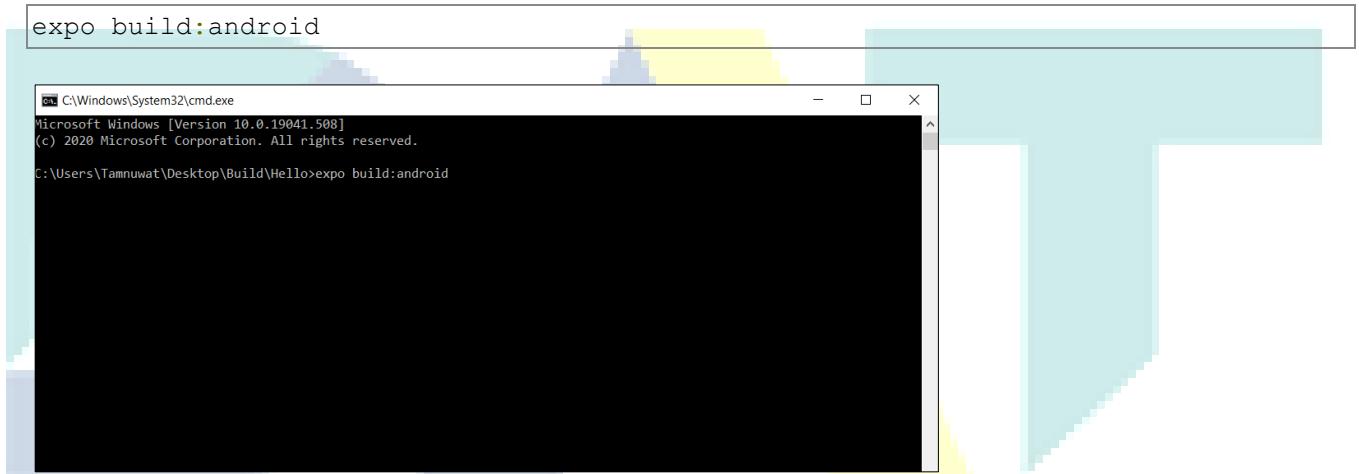
There is a new version of expo-cli available (3.27.4).
You are currently using expo-cli 3.22.3
Install expo-cli globally using the package manager of your choice;
for example: 'npm install -g expo-cli' to get the latest version

Starting project at C:\Users\Tamnuwat\Desktop\Build\Hello
Expo DevTools is running at http://localhost:19002
Opening DevTools in the browser... (press shift-d to disable)
Starting Metro Bundler on port 19001.

exp://10.64.129.1:19000

[QR code]
```

2. ทำการเปิด Command Line ขึ้นมาใหม่จากนั้นพิมพ์คำสั่ง

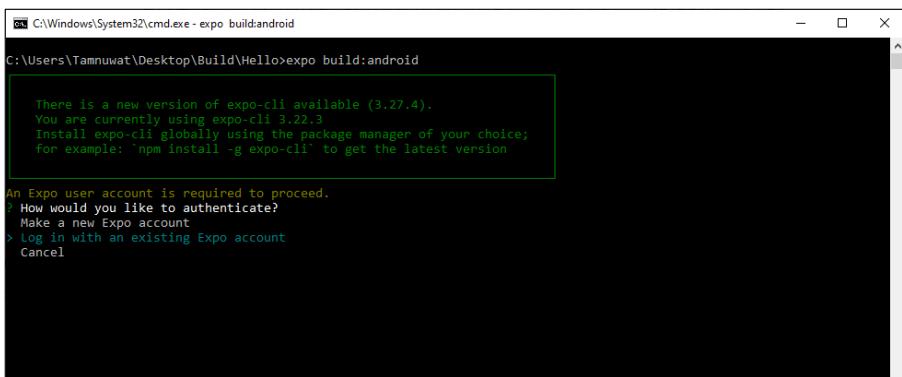


```
expo build:android

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Tamnuwat\Desktop\Build\Hello>expo build:android
```

3. หากไม่ได้ล็อกอินด้วยบัญชี Expo ระบบจะบังคับให้เราสมัครสมาชิกหรือล็อกอิน ในที่นี้ให้เราเลือก Log in with an existing Expo account



```
C:\Windows\System32\cmd.exe - expo build:android

C:\Users\Tamnuwat\Desktop\Build\Hello>expo build:android

There is a new version of expo-cli available (3.27.4).
You are currently using expo-cli 3.22.3
Install expo-cli globally using the package manager of your choice;
for example: 'npm install -g expo-cli' to get the latest version

An Expo user account is required to proceed.
? How would you like to authenticate?
  Make a new Expo account
> Log in with an existing Expo account
  Cancel
```

4. ทำการกรอก UserName และ Password ที่เราใช้สมัครบัญชี Expo

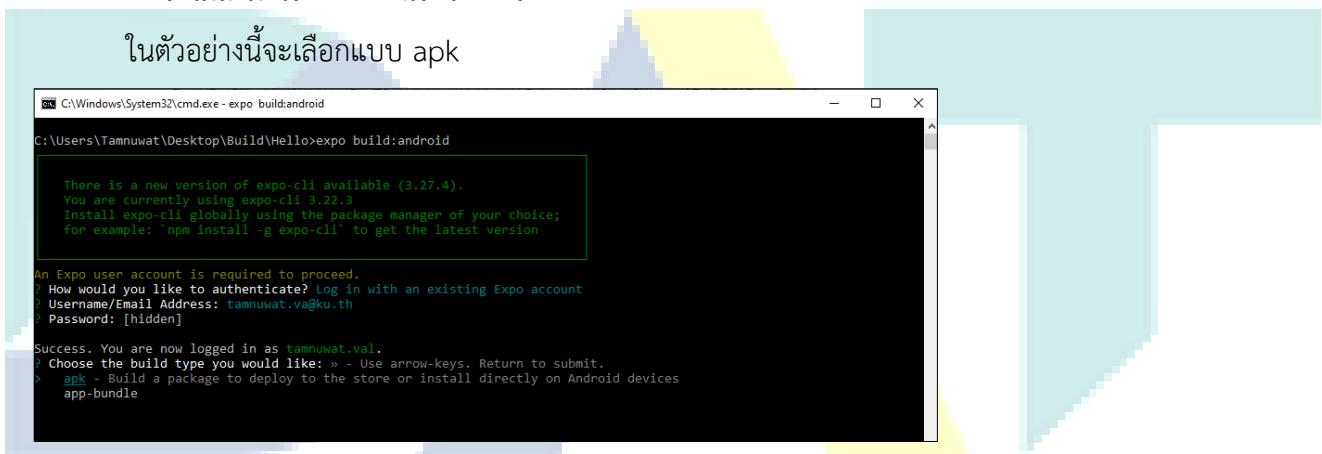
```
C:\Windows\System32\cmd.exe - expo build:android

There is a new version of expo-cli available (3.27.4).
You are currently using expo-cli 3.22.3
Install expo-cli globally using the package manager of your choice;
for example: 'npm install -g expo-cli' to get the latest version

An Expo user account is required to proceed.
? How would you like to authenticate? Log in with an existing Expo account
? Username/Email Address: tamnuwat.va@ku.th
? Password: [input is hidden]
```

5. 既然นี่เราจะต้องเลือกลักษณะการสร้าง โดยมี 2 ลักษณะดังนี้

- apk เป็นการสร้าง(Build)ที่จะได้ไฟล์ APK ขนาดปกติสามารถอัปโหลดขึ้นเพย์สโตร์และติดตั้งผ่านโทรศัพท์ได้ทันที
- app-bundle เป็นการสร้าง(Build)ไฟล์ที่ใช้สำหรับอัปโหลดขึ้นเพย์สโตร์โดยไฟล์จะมีขนาดเล็กกว่าวิธีการแรก แต่จะไม่สามารถติดตั้งผ่านโทรศัพท์ได้



6. 既然นี่เลือก Generate new keystore

```
C:\Windows\System32\cmd.exe - expo build:android

There is a new version of expo-cli available (3.27.4).
You are currently using expo-cli 3.22.3
Install expo-cli globally using the package manager of your choice;
for example: 'npm install -g expo-cli' to get the latest version

An Expo user account is required to proceed.
? How would you like to authenticate? Log in with an existing Expo account
? Username/Email Address: tamnuwat.va@ku.th
? Password: [hidden]

Success. You are now logged in as tamnuwat.val.
✓ Choose the build type you would like: » apk
Checking if there is a build in progress...

Configuring credentials for tamnuwat.val in project Hello
? Would you like to upload a Keystore or have us generate one for you?
If you don't know what this means, let us generate it! :) » - Use arrow-keys. Return to submit.
> Generate new keystore
I Want to upload my own file
```

7. 既然นี่รอสักครู่ระบบจะทำการอัปโหลดไฟล์โค้ดเราทั้งหมดขึ้นไปบนคราวน์และสร้าง APK ไว้บนนี่เราสามารถดูความคืบหน้าการสร้างได้จากลิงค์ที่แสดงไว้

```
npm
Publishing to channel 'default'...
Building iOS bundle
Building Android bundle
Analyzing assets
Uploading assets
No assets changed, skipped.
Processing asset bundle patterns:
- C:\Users\tamnuwat\Desktop\Build\Hello\**\*
Uploading JavaScript bundles
Publish complete

The manifest URL is: https://exp.host/@tamnuwat.val/Hello ( https://exp.host/@tamnuwat.val/Hello/index.exp?sdkVersion=38.0.0 ). Learn more. ( https://expo.fyi/manifest-url )
The project page is: https://expo.io/@tamnuwat.val/Hello ( https://expo.io/@tamnuwat.val/Hello ). Learn more. ( https://expo.fyi/project-page )
Checking if this build already exists...

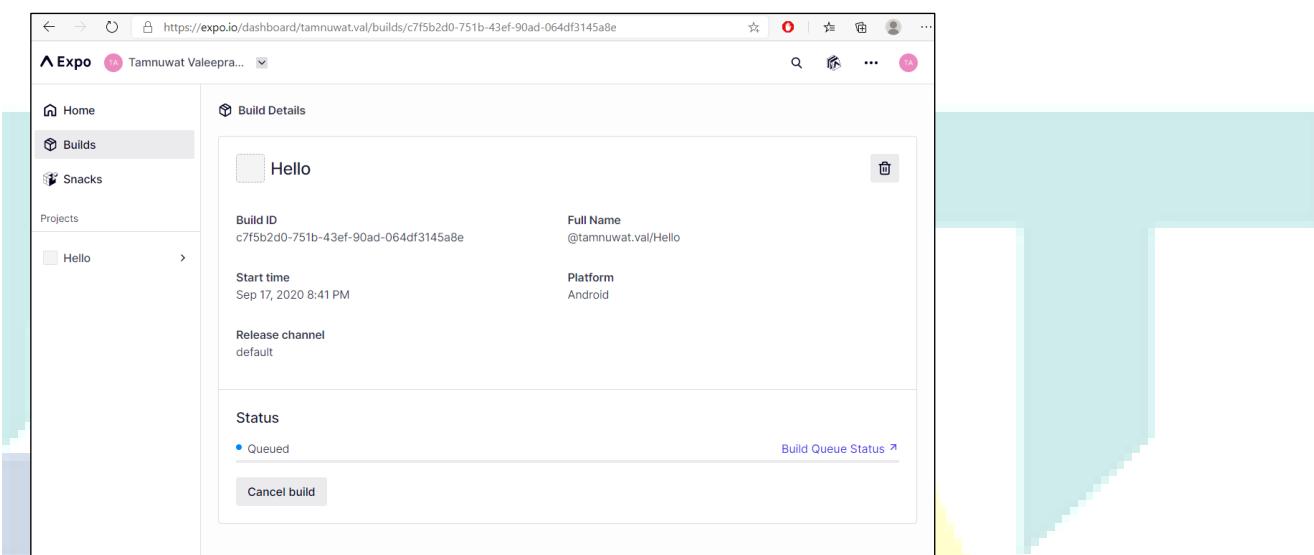
Build started, it may take a few minutes to complete.
You can check the queue length at https://expo.io/turtle-status

You can make this faster. ⓘ
Get priority builds at: https://expo.io/settings/billing

You can monitor the build at
https://expo.io/dashboard/tamnuwat.val/builds/c7f5b2d0-751b-43ef-90ad-064df3145a8e

Waiting for build to complete.
You can press Ctrl+C to exit. It won't cancel the build, you'll be able to monitor it at the printed URL.
```

8. เมื่อเข้าไปยังเว็บไซต์ตามลิงค์ที่ส่งมาให้จะรายละเอียดและสถานะการดำเนินงาน



Digital Academy Thailand

9. เมื่อเสร็จแล้วเราจะสามารถดาวน์โหลด APK ได้จากลิงค์ที่ส่งมาให้

Build ID: c7f5b2d0-751b-43ef-90ad-064df3145a8e
Full Name: @tamnuwat.val/Hello
Start time: Sep 17, 2020 8:41 PM
Platform: Android
Release channel: default
Status: Finished
Build artifact: Download (Artifact available for 29 days)



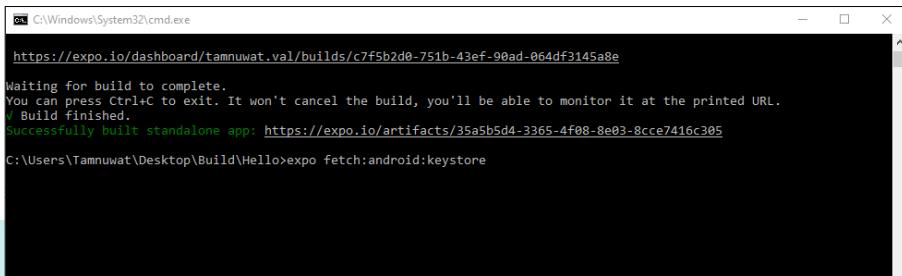
Digital Academy Thailand

การตั้งไฟล์ keystore

เมื่อสร้างไฟล์ APK สำเร็จสิ่งแรกที่ควรทำคือการทำ Backup keystore ไฟล์นี้องด้วยนามีการแก้ไขแอพฯหรือทำการเปลี่ยน Version ของแอพฯเมื่อต้องการอัพเดจแอพฯบน Play Store จะต้องใช้ไฟล์ keystore อันนี้เสมอ ห้ามทำหายเด็ด โดยวิธีการตั้ง keystore ไฟล์สามารถทำได้ดังนี้

1. เปิด terminal และพิมพ์คำสั่งด้านล่าง

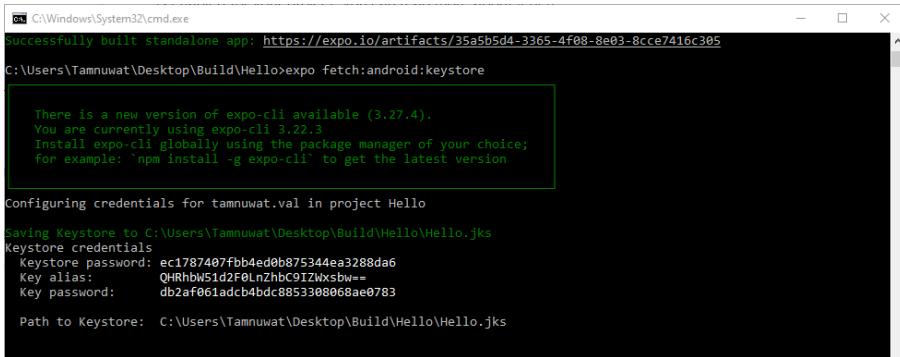
```
expo fetch:android:keystore
```



```
C:\Windows\System32\cmd.exe
https://expo.io/dashboard/tamnuwat.val/builds/c7f5b2d0-751b-43ef-90ad-064df3145a8e
Waiting for build to complete.
You can press Ctrl+C to exit. It won't cancel the build, you'll be able to monitor it at the printed URL.
✓ Build finished.
Successfully built standalone app: https://expo.io/artifacts/35a5b5d4-3365-4f08-8e03-8cce7416c305
C:\Users\tamnuwat\Desktop\Build>Hello>expo fetch:android:keystore
```

Digital Academy Thailand

2. เมื่อวันคำสั่งเสร็จแล้ว terminal จะแสดงข้อมูลและที่อยู่ในของไฟล์ keystore



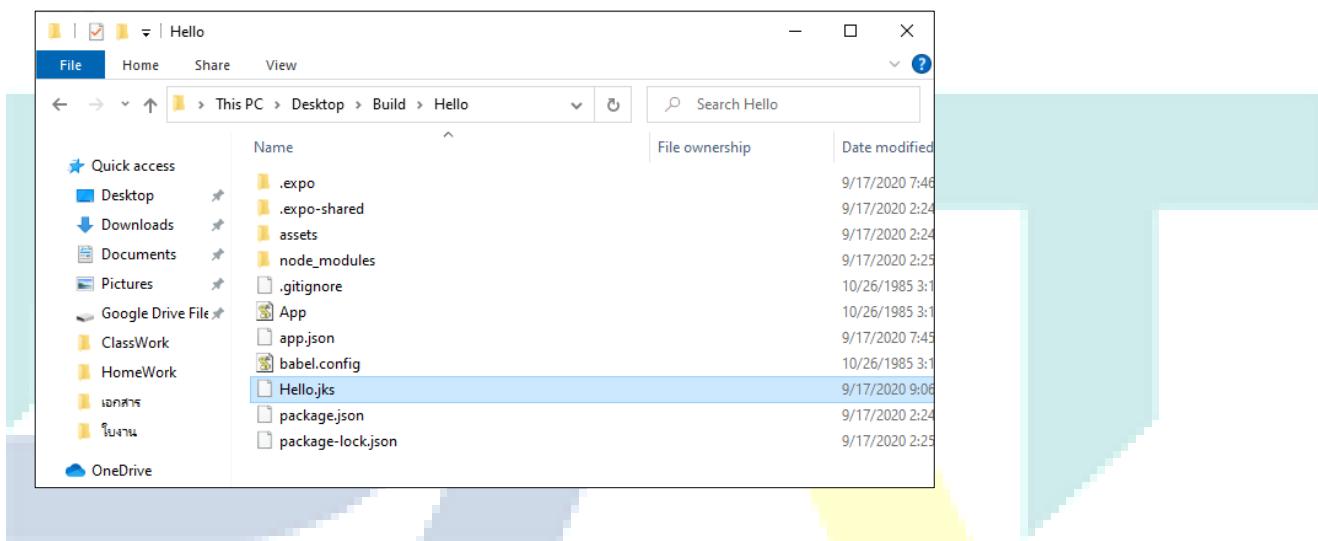
```
C:\Windows\System32\cmd.exe
Successfully built standalone app: https://expo.io/artifacts/35a5b5d4-3365-4f08-e03-8cce7416c305
C:\Users\tamnuwat\Desktop\Build\Hello>expo fetch:android:keystore

There is a new version of expo-cli available (3.27.4).
You are currently using expo-cli 3.22.3
Install expo-cli globally using the package manager of your choice;
for example: `npm install -g expo-cli` to get the latest version

Configuring credentials for tamnuwat.val in project Hello
Saving Keystore to C:\Users\tamnuwat\Desktop\Build\Hello\Hello.jks
Keystore credentials
Keystore password: ec1787407fbb4ed0b87534ea3288da6
Key alias: QHRhbw51dzF0LnzhbC9IZWxsbw==
Key password: db2af061adcb4bcd8853308068ae0783

Path to Keystore: C:\Users\tamnuwat\Desktop\Build\Hello\Hello.jks
```

3. ให้เราทำการเก็บไฟล์ .jks ไว้ใช้ปลดภัย



Digital Academy Thailand

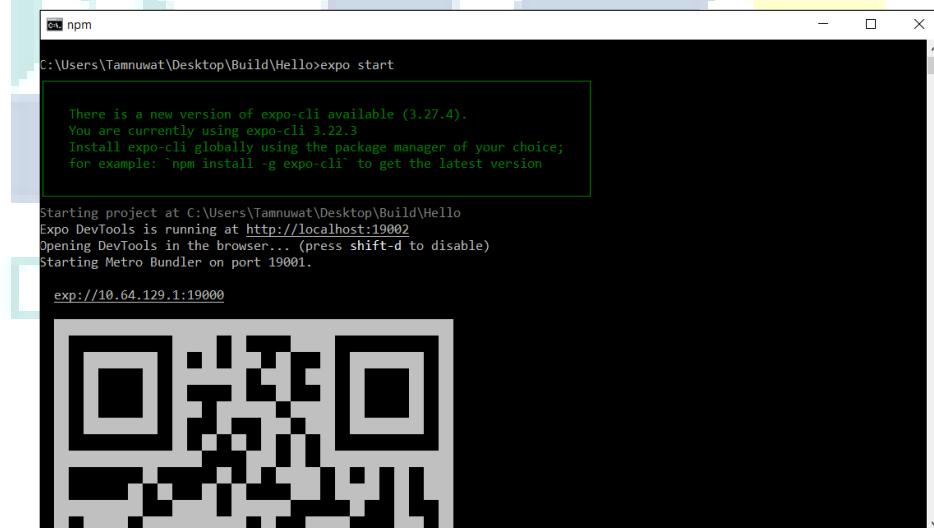
การ Build iOS แอพพลิเคชั่นและการอัปโหลดขึ้น AppStore

เมื่อทำการสมัครบัญชีนักพัฒนาและบัญชี Expo เรียบร้อยแล้ว เราจะทำการสร้างแพ็คเกจไฟล์เพื่อใช้สำหรับอัปโหลดขึ้น Apple store ซึ่งเราจะเรียกว่า IPA ไฟล์ซึ่งเป็นไฟล์โปรแกรมสำหรับระบบปฏิบัติการ iOS มีเข้ารหัสเป็นกรรมสิทธิ์ของ Apple ในการสร้าง IPA เราจำเป็นจะต้องกำหนดรายละเอียดของแพ็คเกจและเวอร์ชันของโค้ดให้เรียบร้อยในไฟล์ app.json เสียก่อนโดยมีรายละเอียดดังนี้

```
{  
  "expo": {  
    "name": "Your App Name",  
    "icon": "./path/to/your/app-icon.png",  
    "version": "1.0.0",  
    "slug": "your-app-slug",  
    "ios": {  
      "bundleIdentifier": "com.yourcompany.yourappname",  
      "buildNumber": "1.0.0"  
    }  
  }  
}
```

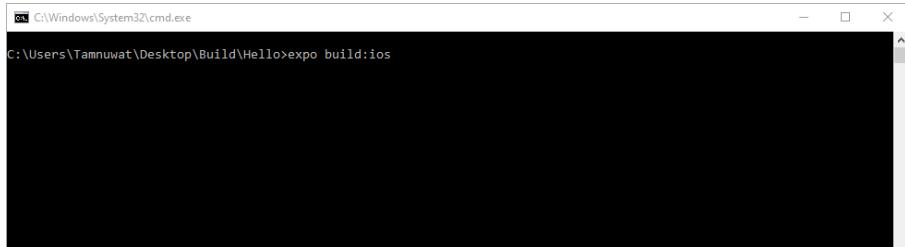
โดยวิธีการสร้างไฟล์ IPA สามารถทำได้ดังนี้

1. ทำการรัน Metro Bundler ขึ้นมาเหมือนตอนที่เราทดสอบแอพฯปกติ



2. ทำการเปิด Command Line ขึ้นมาใหม่จากนั้นพิมพ์คำสั่ง

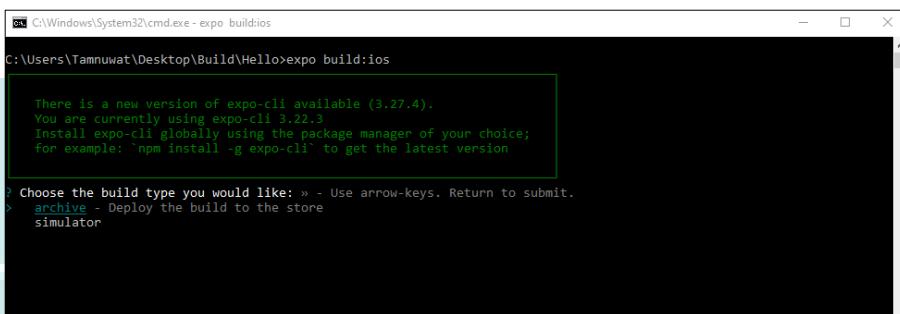
```
expo build:ios
```



```
C:\Windows\System32\cmd.exe
C:\Users\tamnuwat\Desktop\Build\Hello>expo build:ios
```

3. จากนั้นระบบจะให้เราเลือกประเภทการนำไปใช้(Deploy) โดยมี 2 แบบดังนี้

- archive คือการสร้าง(Build) เพื่ออัปโหลดขึ้น Apple store
- simulator คือการสร้าง(Build) เพื่อใช้ในการทดสอบใน Simulator



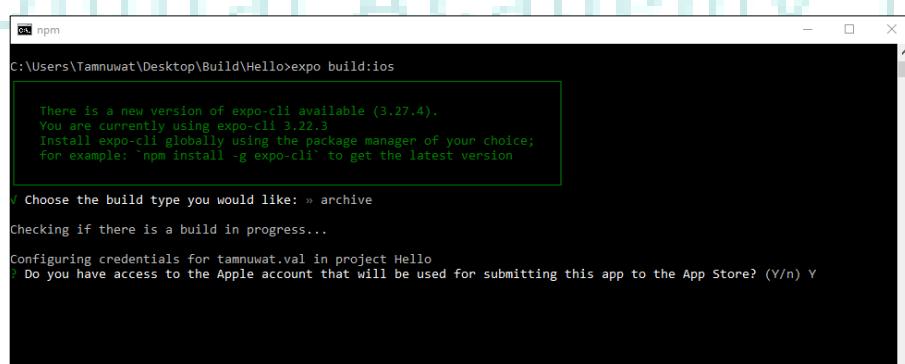
```
C:\Windows\System32\cmd.exe - expo build:ios
C:\Users\tamnuwat\Desktop\Build\Hello>expo build:ios

There is a new version of expo-cli available (3.27.4).
You are currently using expo-cli 3.22.3
Install expo-cli globally using the package manager of your choice;
for example: `npm install -g expo-cli` to get the latest version

? Choose the build type you would like: » - Use arrow-keys. Return to submit.
> archive - Deploy the build to the store
simulator
```

ในตัวอย่างนี้จะสอนทำการสร้างเพื่ออัปโหลดขึ้น Apple store โดยเลือก archive

4. ถ้ามาระบบจะถามว่าเรามี Apple Developer Account หรือไม่เนื่องจากการอัปโหลดแอปฯขึ้นสโตร์จำเป็นจะต้องใช้ Apple Developer Account ให้เราเลือก Y ระบบจะทำการสร้างสร้าง Certificates ที่ใช้ในการอัปโหลดไฟล์ให้อัตโนมัติ(หากเลือก N ต้องจัดการเองทั้งหมด) ไม่แนะนำ



```
C:\Windows\System32\cmd.exe
C:\Users\tamnuwat\Desktop\Build\Hello>expo build:ios

There is a new version of expo-cli available (3.27.4).
You are currently using expo-cli 3.22.3
Install expo-cli globally using the package manager of your choice;
for example: `npm install -g expo-cli` to get the latest version

? Choose the build type you would like: » archive
Checking if there is a build in progress...
Configuring credentials for tamnuwat.val in project Hello
? Do you have access to the Apple account that will be used for submitting this app to the App Store? (Y/n) Y
```

5. จากนั้นทำการกรอก Apple ID และรหัสผ่านคิลล์

```
cmd npm
C:\Users\Tamnuwat\Desktop\Build\Hello>expo build:ios
Choose the build type you would like: » archive
Checking if there is a build in progress...
Accessing credentials for tamnuwat.val in project Hello
Do you have access to the Apple account that will be used for submitting this app to the App Store? Yes
Please enter your Apple Developer Program account credentials. These credentials are needed to manage certificates,
keys and provisioning profiles in your Apple Developer account.
The password is only used to authenticate with Apple and never stored on Expo servers
Learn more here ( https://bit.ly/2VtawHU )
? Apple ID: tamnuwat.va@ku.th
? Password (for tamnuwat.va@ku.th): [input is hidden] -
```

6. เลือก Let Expo handle the process เพื่อจัดการการ Certificates ต่างๆให้

```
cmd npm
We can't auto-generate credentials if you don't have access to the main Apple account.
But we can still set it up if you upload your credentials.
Unable to determine validity of Distribution Certificates.
? Will you provide your own Apple Distribution Certificate? » - Use arrow-keys. Return to submit.
> Let Expo handle the process
I want to upload my own file
```

7. จากนั้นรอสักครู่ระบบจะทำการอปโหลดไฟล์โค้ดเราทั้งหมดขึ้นไปบนクラาว์นและสร้าง APK ไว้บนนั้นเราสามารถดูความคืบหน้าการสร้างได้จากลิงค์ที่แสดงไว้

```
cmd npm
Learn more about JavaScript bundle sizes: https://expo.fyi/javascript-bundle-sizes
Analyzing assets
Saving assets
No assets changed, skipped.

Processing asset bundle patterns:
- C:\Users\Tamnuwat\Desktop\Build\Hello\**\*

Uploading JavaScript bundles
Publish complete

Manifest: https://exp.host/@tamnuwat.val>Hello/index.exp?sdkVersion=38.0.0 Learn more: https://expo.fyi/manifest-url
Project page: https://expo.io/@tamnuwat.val>Hello Learn more: https://expo.fyi/project-page

Checking if this build already exists...

Build started, it may take a few minutes to complete.
You can check the queue length at https://expo.io/turtle-status

You can make this faster. ☕
Get priority builds at: https://expo.io/settings/billing

You can monitor the build at
https://expo.io/dashboard/tamnuwat.val/builds/8413f499-1193-4fcf-b13d-3e8bff8bd681

Waiting for build to complete.
You can press Ctrl+C to exit. It won't cancel the build, you'll be able to monitor it at the printed URL.
- Build queued...
```

8. เมื่อเข้าไปยังเว็บไซต์ตามลิงค์ที่ส่งมาให้จะรายละเอียดและสถานะ การดำเนินงาน

The screenshot shows the 'Build Details' page for a project named 'Hello'. The build ID is 8413f499-1193-4cfc-b13d-3e8bfff8bd681. The status is 'Queued'. The platform is iOS. The start time was Sep 17, 2020 11:03 PM. There is a 'Cancel build' button.

9. เมื่อเสร็จแล้วเราจะสามารถดาวน์โหลด APK ได้จากลิงค์ที่ส่งมาให้

The screenshot shows the 'Build Details' page for the same project 'Hello'. The status is now 'Finished'. A 'Build artifact' button is visible, with a note that the artifact is available for 29 days. The other details remain the same as in the previous screenshot.