06016323 Mobile Device Programming

CHAPTER 2 : FUNDAMENTAL MOBILE DEVICE PROGRAM

React Native คืออะไร?

- Open Source
- Cross-Platform สำหรับทำแอพพิเคชันบนอุปกรณ์เคลื่อนที่ ด้วยภาษา JavaScript และ React โดยใช้ Components ต่าง ๆ

React.js + React Native = Real Native Mobile Apps

ไลบารี่รี ของ JavaScript สำหรับสร้างส่วนติดต่อกับผู้ใช้

ใช้สำหรับการแสดงผลทางจอภาพ เมื่อทำการพัฒนาแวปแอพพิเคชัน

ReactDOM.render(...) ถูกเรียกใช้งานในบางครั้งเมื่อแอพพิเคชันมีการเรียกใช้ฟังก์ชันบางอย่างในไลบารี่รีโดยเฉพาะมีปรับปรุงการแสดงผลบนจอภาพ

ไลบรารีที่เกี่ยวกับเว็บและการวิธีแสดงผลองค์ประกอบของ HTML และอื่น ๆ ตอบสนองต่อ React เรียกว่า Platform agnostic

กลุ่มของ Components ชนิดพิเศษที่ React ทำแยกออกมาโดยเฉพาะการพัฒนา Mobile Apps

Components จะถูกแปล เพื่อให้สามารถแสดงผลไปยัง Native Widgets (iOS หรือ Android) ซึ่งเปรียบเสือนกับ ReactDOM

คุณสมบัติเพิ่มเติมของ React Native คือ สามารถใช้งาน Native Platform APIs กับ JavaScript ได้

มีเครื่องมือสำหรับเชื่อมต่อ กับ JavaScript และ Native Platform Code

iOS

Android

React Native ทำงานอย่างไร

React + Real Native Apps

```
Const App = props => {
Return (
            <View>
               <Text> Hello World!</Text>
            <View>
        );
}
```

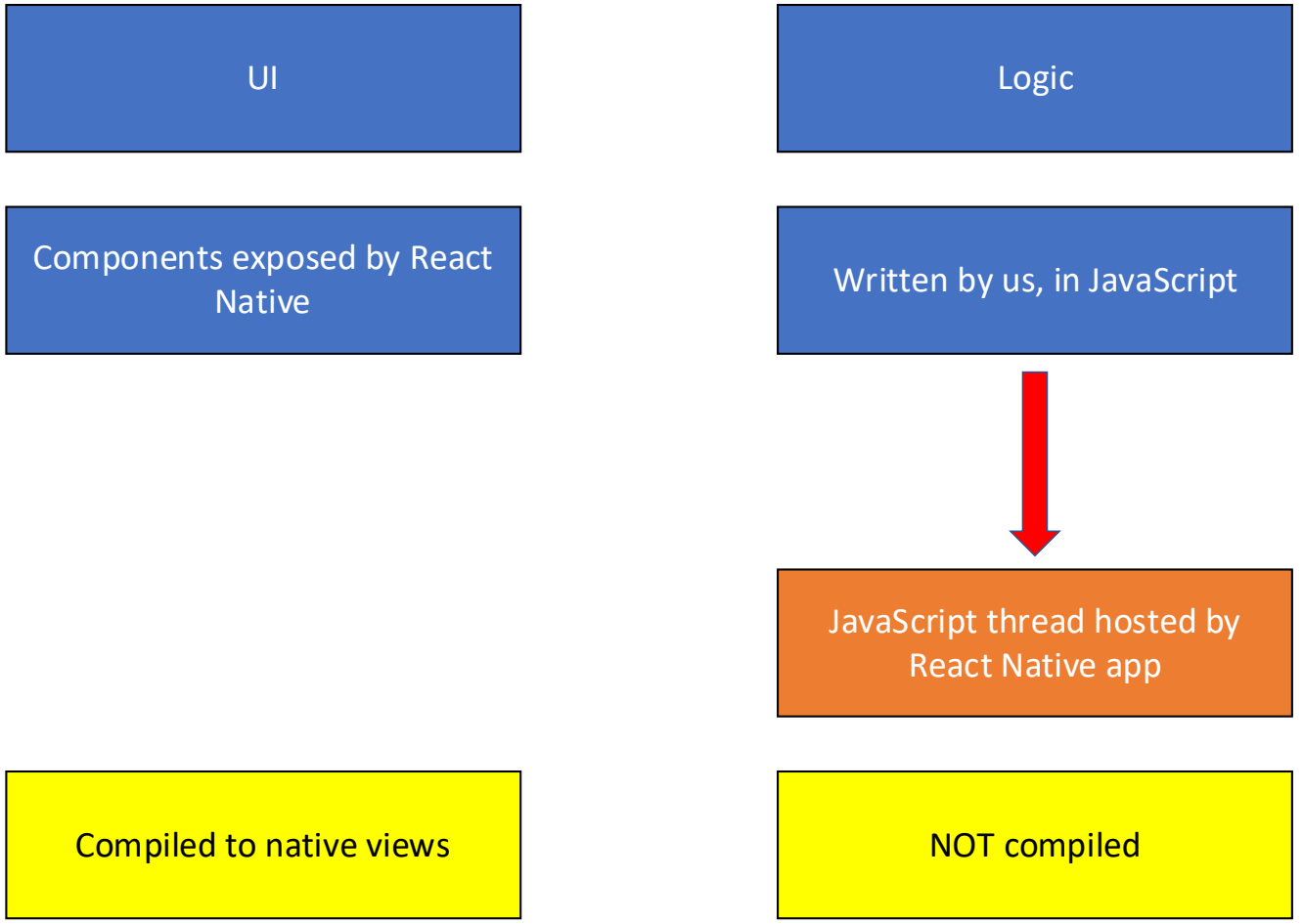
Compiled to

Views are compiled!

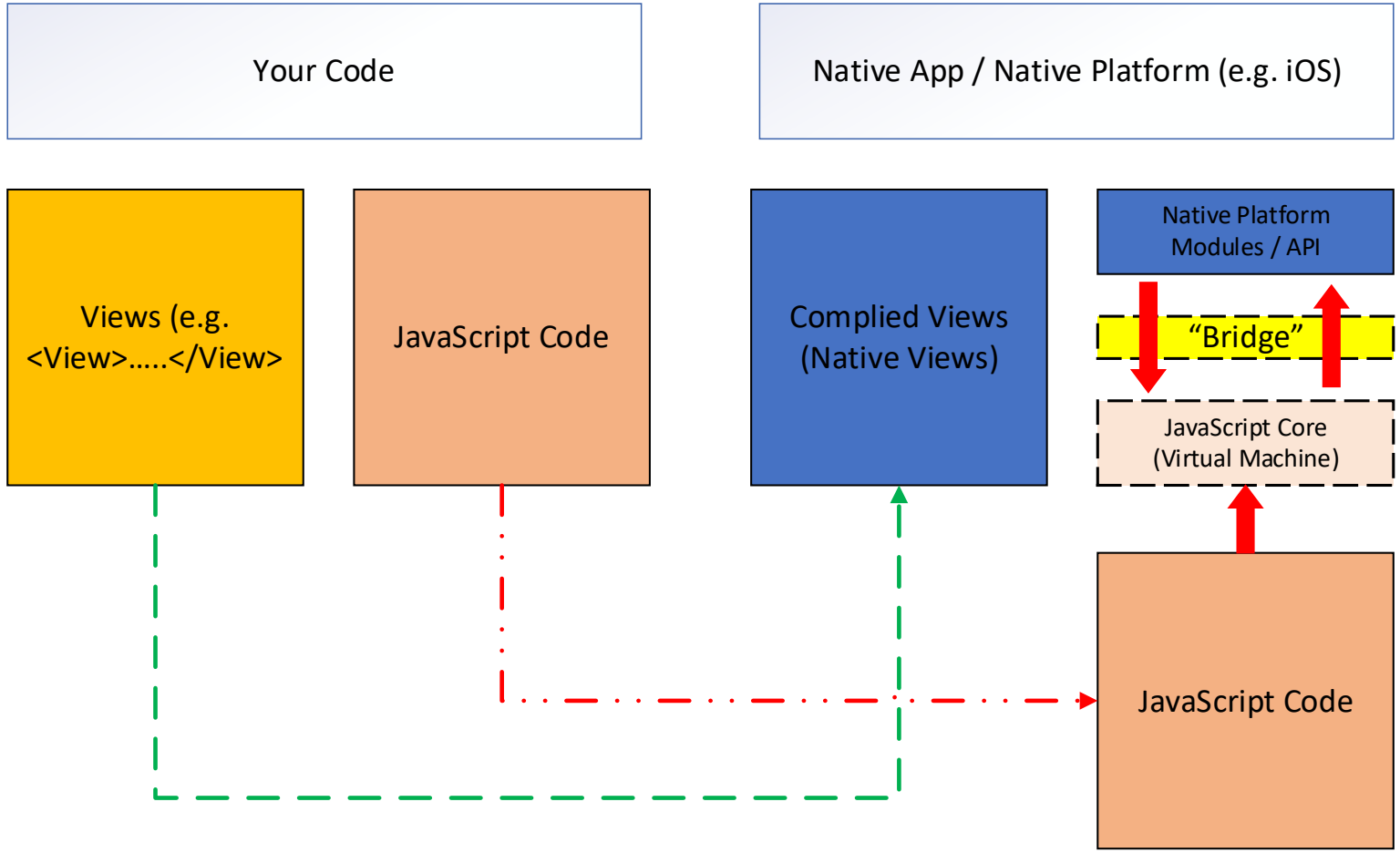
React for the Web	Native Component	Native Component	React Native
	Android.view	UIView	<View>
<input>	EditText	UITextField	<TextInput>
…	…	…	…

React Native maps re-usable components to the respective platform equivalents.

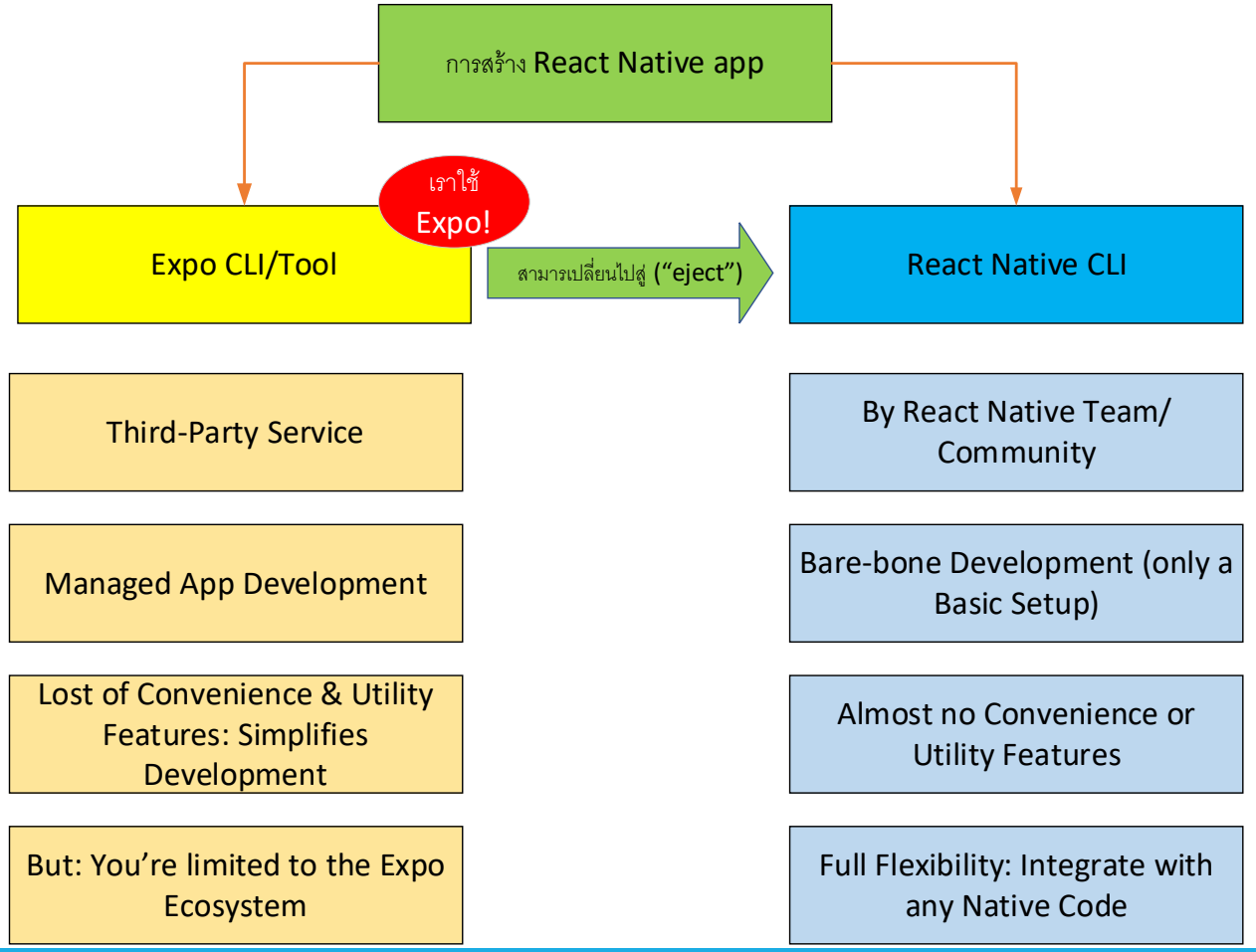
The JavaScript Part / Our Logic ?



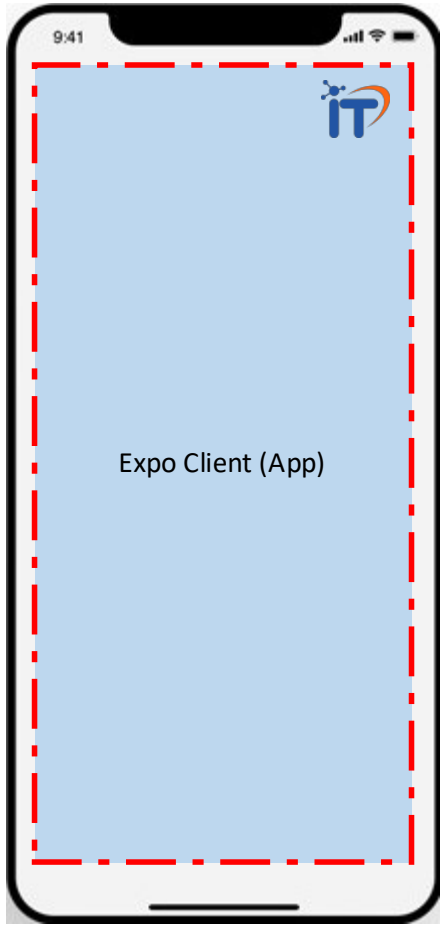
Behind the Scenes



Expo or React Native CLI?



Expo ทำงานอย่างไร



ทำการโหลดเข้าสู่ Expo App

โปรแกรมของอุปกรณ์เคลื่อนที่ และ การตั้งค่าของแอพพิเคชัน

สำหรับการพัฒนา

เราสามารถนำเสนอโมบายแอพ ที่พัฒนาจากซอฟต์แวร์สำหรับอุปกรณ์เคลื่อนที่ เช่น **Expo apps**

เราสามารถนำเสนอแอพ ที่อุปกรณ์เคลื่อนที่แบบ Standalone ได้

แอพพิเคชันสามารถเปลี่ยนการแสดงผลไปที่ Non-Expo development flow ได้

We’re going to cover the core concepts behind React:

- components
- JSX
- props
- state

Component

```
import React, { Component } from 'react';
import { Text } from 'react-native';

class Koala extends Component {
  render() {
    return (
      <Text>Hello,  Koala!</Text>
    );
  }
}

export default Koala;
```

- การกำหนด component เพื่อให้สามารถเรียกใช้งานได้ง่ายและ สะดวก
  - ทำการ Import component จาก React
  - ทำการ Extend component
  - ใช้ฟังก์ชัน Render เพื่อส่งค่ากลับไปที่ React element
  - Export ฟังก์ชันไปให้ฟังก์ชันอื่น ๆ

Core Components

โครงสร้างใน React Native

โมบายแอพ / ส่วนประกอบของ Component

“Translation” ที่อยู่ใน Native UI Widgets ถูกจัดการโดย React Native

ทำการรวบรวม องค์ประกอบหลัก(Core Components) และ องค์ปรกอบอื่น ๆ (Other built-in Components) เข้าสู่โมบายแอพพิเคชัน

<View>

<Text>

<Button> / <Touchable…>

<TextInput>

<Image>

Components

```
Import React from ‘react’;
Import {StyleSheet, Text, View } from ‘react-native’;

Const App = props => {
Return (
            <View>
                <Text> {props.title}</Text>
            <View>
        );
}
```

ทำไมต้อง JSX

- React รวบรวมลอจิกของการแสดงผลควบคู่ไปกับลอจิกของ UI อื่น ๆ ซึ่งทำให้การจัดการเหตุการณ์ สถานะที่เปลี่ยนแปลงของฟังก์ชัน และการเตรียมข้อมูลสำหรับการแสดงผลของแอพพิเคชัน แทนการแยกเทคโนโลยีที่ใส่มาร์กอัปและตรรกะไว้คนละไฟล์กัน
- ดังนั้น React ทำการแบ่งออกเป็นยูนิตและให้สามารถเชื่อมโดยถึงกันได้ โดยใช้หลักการ loosely coupled จึงทำให้เกิดเป็น “component” ที่มีหลักการทั้งสองอย่างไว้ด้วยกัน

การฝังนิพจน์ใน JSX

- ทำการประกาศตัวแปรชื่อ name และทำการเรียกใช้ใน JSX ที่ครอบด้วยเครื่องหมายปีกกา

```
const name = 'Josh Perez';
const element = <h1>สวัสดี, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Example JSX

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

const Koala = () => {
  const name = "Maru";
  return (
    <Text>Hello, I am {name}!</Text>
  );
}

export default Koala;
```

- React and React Native ใช้ JSX โดยมีไวยากรณ์ภายใน JavaScript ได้ดังนี้: <Text> สวัสดีฉันเป็น Koala ของคุณ! </Text> เนื่องจาก JSX เป็น JavaScript เราจึงใช้ตัวแปรภายในได้ ซึ่งเราได้ประกาศชื่อ Koala, {name} , และฝังด้วยวงเล็บปีกกาด้านใน <Text>

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

const getFullName = (firstName, secondName, thirdName) => {
  return firstName + " " + secondName + " " + thirdName;
}

const Koala = () => {
  return (
    <Text>
      Hello, I am {getFullName("Rum", "Tum", "Tugger")}!
    </Text>
  );
}

export default Koala;
```

Custom Components

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

const Koala = () => {
 return (
  <View>
   <Text>Hello, I am...</Text>
   <TextInput
    style={{
     height: 40,
     borderColor: 'gray',
     borderWidth: 1
    }}
    defaultValue="Name me!"
   />
  </View>
 );
}

export default Koala;
```

- ตัวอย่างการใช้ Basic Component ใน React Native Core Component
- Text Component
- TextInput Component
- View Component

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

const Koala = () => {
  return (
    <View>
      <Text>I am also a koala!</Text>
    </View>
  );
}

const Cafe = () => {
  return (
    <View>
      <Text>Welcome!</Text>
      <Koala />
      <Koala />
      <Koala />
    </View>
  );
}

export default Cafe;
```

Properties

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

const Koala = (props) => {
  return (
    <View>
      <Text>Hello, I am {props.name}!</Text>
    </View>
  );
}

const Cafe = () => {
  return (
    <View>
      <Koala name="Maru" />
      <Koala name="Jellylorum" />
      <Koala name="Spot" />
    </View>
  );
}
export default Cafe;
```

- คือการเปลี่ยนค่าของข้อมูลในตัวแปรของ Component ที่อยู่ใน React

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

const ITApp = () => {
  return (
    <View>
      <Image
        source={{uri: “c:/../IT_Logo.png"}}
        style={{width: 200, height: 200}}
      />
      <Text>Hello, I am study in Faculty of IT</Text>
    </View>
  );
}

export default ITApp;
```

State

- การเปลี่ยน สถานะของ ตัวแปร เราสามารถทำการเปลี่ยนแปลงได้จากการให้ค่าหรือ มีกำหนดค่าจากฟังก์ชันอื่น ๆ ที่เข้ามาใช้ หรือ เรียกใช้งาน Component ที่ถูกสร้างขึ้น
- การเปลี่ยนสถานะ ของ Component ของ React มี 2 รูปแบบ
  - State with Function Components
  - State with Class Components

State with Function Components

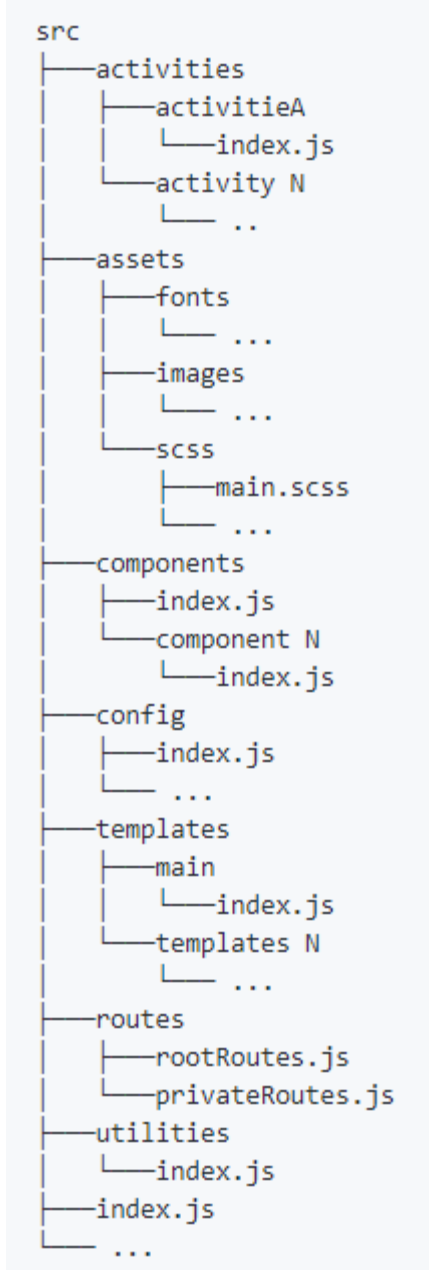
```
import React, { useState } from "react";
import { Button, Text, View } from "react-native";

const Koala = (props) => {
  const [isHungry, setIsHungry] = useState(true);

  return (
    <View>
      <Text>
        I am {props.name}, and I am {isHungry ? "hungry" : "full"}!
      </Text>
      <Button
        onPress={() => {
          setIsHungry(false);
        }}
        disabled={!isHungry}
        title={isHungry ? "Pour me some milk, please!" : "Thank you!"}
      />
    </View>
  );
}
```

```
const Cafe = () => {
  return (
    <>
      <Koala name="Munkustrap" />
      <Koala name="Spot" />
    </>
  );
}

export default Cafe;
```

State with Class Components

```
import React, { Component } from "react";
import { Button, Text, View } from "react-native";
class Koala extends Component {
  state = { isHungry: true };
  render(props) {
    return (
      <View>
        <Text>
          I am {this.props.name}, and I am
          {this.state.isHungry ? " hungry" : " full"}!
        </Text>
        <Button
          onPress={() => {
            this.setState({ isHungry: false });
          }}
          disabled={!this.state.isHungry}
          title={
            this.state.isHungry ? "Pour me some milk, please!" : "Th
ank you!"    }
        />
      </View>
    ); } }
```

```
class Cafe extends Component {
  render() {
    return (
      <>
        <Cat name="Munkustrap" />
        <Cat name="Spot" />
      </>
    );
  }
}

export default  Cafe;
```



❖ activities

แบ่งตามการทำงาน การแสดงผล หรือ แบ่งตาม route ที่เรากำหนด activity จะเป็นตัวกำหนด layout

❖ Assets

ข้างใน folder จะประกอบด้วย font image และ scss หรือ css ต่างๆ

❖ Components

ข้างในจะประกอบด้วย component ที่ต้องใช้บ่อยๆ เช่น component card หรือ component button component จะมี index.js จะเป็นตัวรวม component ทั้งหมดไว้ แล้ว export component เมื่อต้องการเรียกจะเรียกเฉพาะ index.js เราจะไม่เรียก component ตรง component จะมี component ที่ใช้ เฉพาะ component เราจะรวมไว้ในโฟรเดอร์ แล้ว export index.js

❖ Config

เก็บค่า config ต่างๆไว้ เช่น endpoint ของ service เป็นต้น

❖ Templates

เก็บโครงสร้างของหน้าเว็บ เช่น ถ้าโครงสร้างของหน้า login กับ main แตกต่างกันเราก็จะแบ่งเป็น 2 template

❖ Routes

ตัวกำหนดเส้นทางการวิ่งไปหน้าต่างๆ ภายในเว็บ จะ route เป็น 2 ประเภท route ที่มีเงื่อนไขในการเข้าถึง และไม่ต้องมีเงื่อนไขในการเข้าถึง

❖ Utilities

เก็บฟังก์ชันที่ใช้งานบ่อยหรือฟังก์ชันที่ใช้งานร่วมกันหลายๆ ที่ เช่น ฟังก์ชัน isEmpty()

JavaScript คืออะไร

- JavaScript เป็นภาษาที่ทำงานร่วมกับ HTML
- ลูกเล่นต่างๆ ของ HTML5 จะสามารถเรียกใช้ได้อย่างมีประสิทธิภาพด้วย JavaScript
- HTML5 = เนื้อหา (นิ่งๆ) + CSS3 = รูปแบบ (สวยๆ) JavaScript = สั่งให้ประมวลผล (สร้างความ Dynamic ให้กับเนื้อหา)
- เรียกใช้โดยคำสั่งง่ายๆ คือ <script language="javascript" type="text/javascript“ src="../scripts/example.js"></script>
- สามารถเขียนแทรกเข้าไปใน HTML ได้เลย แต่ใน HTML5 เน้นให้สร้างไฟล์ JS ต่างหาก

ข้อดีของ JavaScript

- เขียนโปรแกรมที่มีความซับซ้อน เช่น
  - สร้าง และเรียกดูฐานข้อมูล เช่น ระบบการสมัครเป็นสมาชิก และประวัติการเรียน เป็นต้น
  - ติดต่อเรียกข้อมูลจากเครื่องแม่ข่าย เช่น เรียกระบบการเรียนผ่านอุปกรณ์ Mobile อาจจะ เรียกข้อมูลบทเรียนใหม่ๆ
  - เชื่อมโยงระบบแผนที่ของอุปกรณ์ Mobile เพื่อเรียกดูตำแหน่ง
  - จัดเก็บข้อมูลที่อุปกรณ์ Mobile แบบ Local ทำให้ไม่ต้องเข้าอินเทอร์เน็ตทุกครั้งที่ใช้งาน
  - เขียนควบคุมการทำงานของปุ่ม เช่น ปุ่มบังคับการเคลื่อนไหวของตัวละครในเกม เป็นต้น

Embedding JavaScript in HTML

- By using the SCRIPT tag
- By specifying a file of JavaScript code
- By specifying a JavaScript expression as the value for an HTML attribute
- By using event handlers within certain other HTML tags

SCRIPT Tag

The <SCRIPT> tag is an extension to HTML that can enclose any number of JavaScript statements as shown here:

```
<SCRIPT>
    JavaScript statements...
</SCRIPT>
```

A document can have multiple SCRIPT tags, and each can enclose any number of JavaScript statements.

Hiding scripts in comment tags

```
<SCRIPT>
<!-- Begin to hide script contents from old browsers.
JavaScript statements...
// End the hiding here. -->
</SCRIPT>
```

Famous “Hello World” Program

```
<html>
<body>
  <script language="JavaScript">
    document.write(“Hello, World!”)
  </script>
</body>
</html>
```

JavaScript code in a file

- The SRC attribute of the <SCRIPT> tag lets you specify a file as the JavaScript source (rather than embedding the JavaScript in the HTML).
- This attribute is especially useful for sharing functions among many different pages.

Statements

Conditional Statement: if…else

if (*condition*) {
statements1
} **else** {
statements2
}

Example

```
// if else
function getFood(isVegetarian) {
  if (isVegetarian) {
    return 'tofu';
  } else {
    return 'fish';
  }
}

// ternary operator
function getFood(isVegetarian) {
  return isVegetarian ? 'tofu' : 'fish';
}
```

Loop Statements

- for statement:

for ([*initial-expression*]; [*condition*]; [*increment-expression*]) {
statements
}

- while statement:

while (*condition*) {
statements
}

```
const getAnimalsContent = animals => {
  let content = [];
  for (let i = 0; i < animals.length; i++) {
    const item = animals[i];
    content.push(<li key={item.id}>{item.animal}</li>);
  }
  return content;
};

return <ul>{getAnimalsContent(animals)}</ul>;
```

```
// Example with an id

const RenderList = props => {
 const animals = [
  { id: 1, animal: "Dog" },
  { id: 2, animal: "Bird" },
  { id: 3, animal: "Cat" },
  { id: 4, animal: "Mouse" },
  { id: 5, animal: "Horse" }
 ];

 return (
  <ul>
   {animals.map(item => (
    <li key={item.id}>{item.animal}</li>
   ))}
  </ul>
 );
};
```

Example

```
let i = 0

while (i<10) {

    console.log (“i”, i)

    i++

}
```

Datatype conversion

- JavaScript is a loosely typed language. That means you do not have to specify the data type of a variable when you declare it, and data types are converted automatically as needed during script execution. So, for example, you could define a variable as follows: `var answer = 42`
- And later, you could assign the same variable a string value, for example, `answer = "Thank you"`

Datatype conversion (cont'd)

- In expressions involving numeric and string values, JavaScript converts the numeric values to strings. For example, consider the following statements:

```
x = "The answer is " + 42
y = 42 + " is the answer."
```

Defining and calling Functions

- Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure--a set of statements that performs a specific task. A function definition has these basic parts:
  - The **function** keyword.
  - A function name.
  - A comma-separated list of arguments to the function in parentheses.
  - The statements in the function in curly braces.

TextInput Component

- คอมโพเนท์ พื้นฐานสำหรับจัดการป้อนข้อมูลเข้าทางคีย์บอร์ด ในโมบายแอพ ที่กำลังพัฒนา
- มีคุณสมบัติหลายประการ เช่น auto-correction, auto-capitalization, pleaceholder text และ แสดงรูปแบบคีย์บอร์ดที่หลากหลาย เช่น คีย์บอร์ดที่เป็นตัวเลข
- ตัวอย่างการใช้งาน

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

.

.

<View style={{padding: 10}}>
<TextInput
      style={{height: 40}}
      placeholder="Type here to translate!"
      onChangeText={text => setText(text)}
      defaultValue={text}
    />
 </View>
```

เอกสารเพิ่มเติม : https://reactnative.dev/docs/textinput

Layout Props

- คุณสมบัติหลักสำหรับการแสดงผลทางหน้าจอ การจัดตำแหน่งการแสดงผลของ React Native มีหลายคุณสมบัติ เช่น Flex, padding , alignContent และอื่น ๆ
- ลักษณะการใช้งาน ใส่ใน <View> tags หรือ const stytes ของ Object ต่าง ๆ เปรียบเสมือนเป็น Parameter เพื่อให้โปรแกรมทราบถึงตำแหน่งของการแสดงผล เช่น
  - Padding เป็นการระบุตำแหน่งการแสดงผลข้อมูลที่ต้องการให้ปรากฏทางหน้าจอ ในตัว Padding มีการกำหนดตำแหน่ง ทั้งหมด 4 ประเภท คือ paddingTop, paddingBottom, paddingLeft, และ paddingRight.

TYPE	REQUIRED
number, string	No

เอกสารเพิ่มเติม : https://reactnative.dev/docs/layout-props

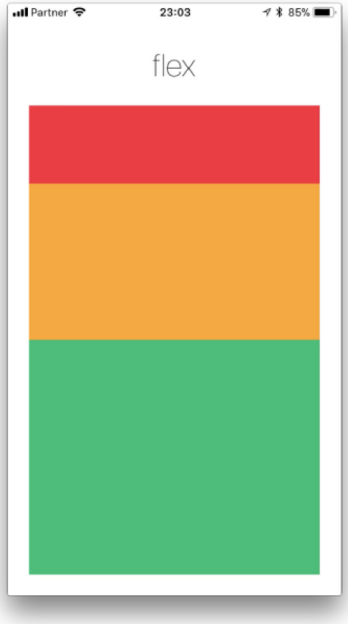
ตัวอย่าง : padding parameter

```
/* Apply to all four sides */
padding: 1em;

/* vertical | horizontal */
padding: 5% 10%;

/* top | horizontal | bottom */
padding: 1em 2em 2em;

/* top | right | bottom | left */
padding: 5px 1em 0 2em;

/* Global values */
padding: inherit;
padding: initial;
padding: unset;
```

Layout Props

- **Flex Component** มีการเรียกใช้งานจาก **Flexbox algorithm** โดยคอมโพเนทนี้ ถูกออกแบบมาเพื่อจัดการแสดงผลให้ยืดหยุ่นตามขนาดของหน้าจอของอุปกรณ์เคลื่อนที่

- หลักการทำงานของ **Flex** จะแสดงผลตามขนาดที่ระบุ เช่น **flex: 1** แสดงสีแดง **flex: 2** แสดงสีเหลือง และ **flex: 3** แสดงสีแดง เมื่อกำหนดแบบนี้ ตัว **Flexbox algorithm** จะทำการนำตัวเลขไปรวมกัน แล้วคำนวณ **1+2+3 = 6** เพื่อหาขนาดการแสดงผล ทำให้การแสดงผลของสีแดง เท่ากับ **1/6** ของหน้าจอ, สีเหลือง เท่ากับ **2/6** และสีเขียว เท่ากับ **3/6** ตามลำดับ



เอกสารอ้างอิง : **https://reactnative.dev/docs/flexbox**

Flex Direction (flexDirection)

- คือการกำหนดทิศทางการแสดงของผลลัพธ์ทางจอภาพ ซึ่งมีทิศทางดังนี้
  - row แสดงจากซ้ายไปขวาต่อกันจนจบ โดยบรรทัดถัดไปจะทำการแสดงใต้บรรทัดก่อนหน้า

```
<View style={{flex: 1, flexDirection: 'row'}}>
```

  - column (default value) แสดงผลลัพธ์จากบนลงล่าง หากขึ้นบรรทัดใหม่จะทำการแสดงจากทางซ้าย
  - row-reverse แสดงผลกลับทิศ กับ row parameter
  - column-reverse แสดงผลกลับทิศ กับ column parameter

ตัวอย่าง การใช้งาน Flexbox Algorithm