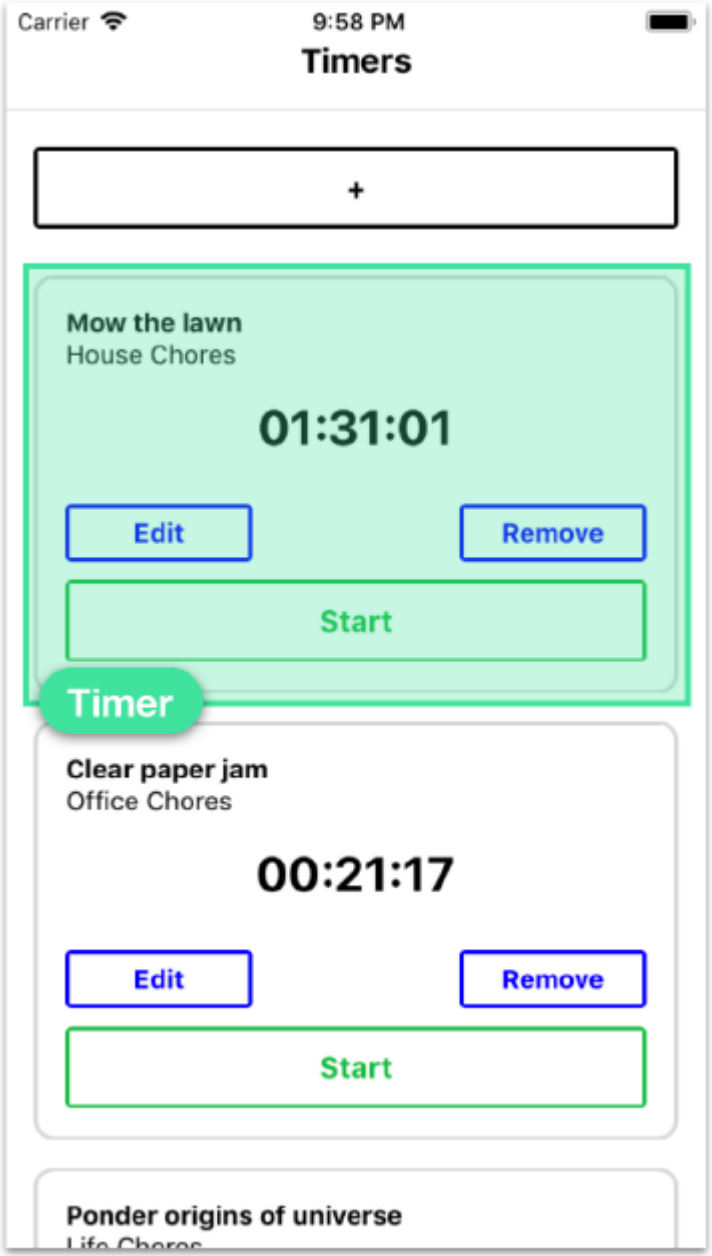
06016323 Mobile Device Programming

CHAPTER 4 : COMPONENTS COMMUNICATION



คอมโพเนนต์ (Component)

- แอปพลิเคชันประกอบไปด้วยคอมโพเนนต์ต่างๆ ที่ทำหน้าที่เป็นส่วนประกอบย่อยๆ ในแอปพลิเคชัน
- App ถือเป็น รูทคอมโพเนนต์ (ระดับบนสุด)
- การสร้างคอมโพเนนต์ ทำได้ 2 วิธี
  - Class Component
  - Function Component

Class Component

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

class Cat extends Component {
  render() {
    return (
      <Text>Hello, I am your cat!</Text>
    );
  }
}
export default Cat;
```

Function Component

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

const Cat = () => {
  return (
    <Text>Hello, I am your cat!</Text>
  );
}

export default Cat;
```

คุณสมบัติพื้นฐานของคอมโพเนนต์ (Component)

- สามารถซ้อนกันได้ (Nesting)
  - คอมโพเนนต์หลัก StudentList อาจประกอบด้วยคอมโพเนนต์ย่อย Student
- สามารถนำมาใช้งานซ้ำได้ (Reusable)
  - เราสามารถนำคอมโพเนนต์เดิมมาใช้ใหม่ได้
- สามารถกำหนดค่าเริ่มต้นได้ (Configuration)

Multiple Components

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

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

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

export default Cafe;
```

Multiple Components (Separate files)

Parent Component

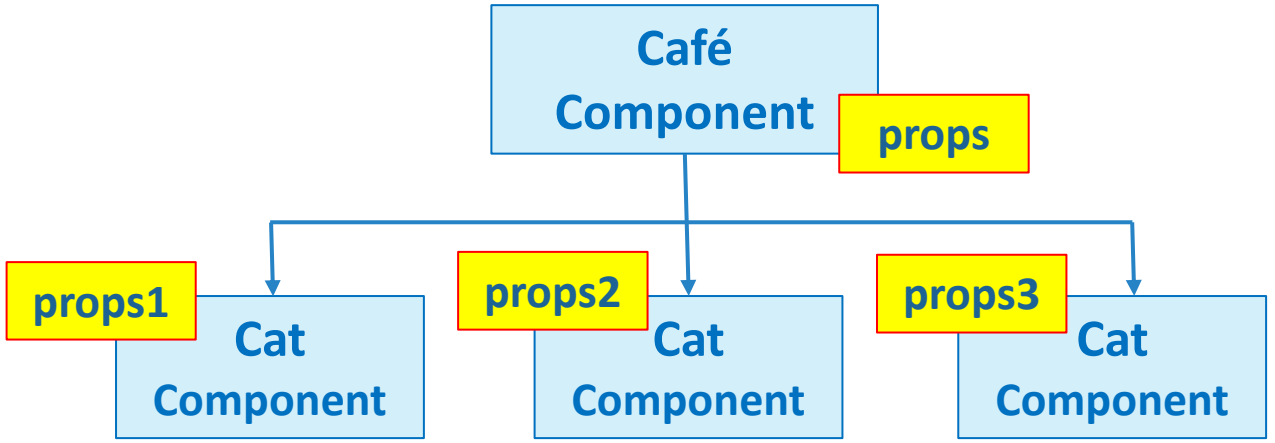
```
import React from 'react';
import { Text, TextInput, View } from 'react-native’;
import Cat from './Cat'

const Cafe = () => {
  return (
    <View>
      <Text>Welcome!</Text>
      <Cat />
      <Cat />
      <Cat />
</View>
  );
}
export default Cafe;
```

Child Component

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

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

export default Cat;
```

การส่งค่าจาก Parent -> Child Component

- ใช้ props (properties)
  - ใช้กำหนดคุณสมบัติของคอมโพเนนต์ลูก (Child component) ได้
- รูปแบบ : <ChildComponent pName1=pValue1 pName2=pValue2 ...>



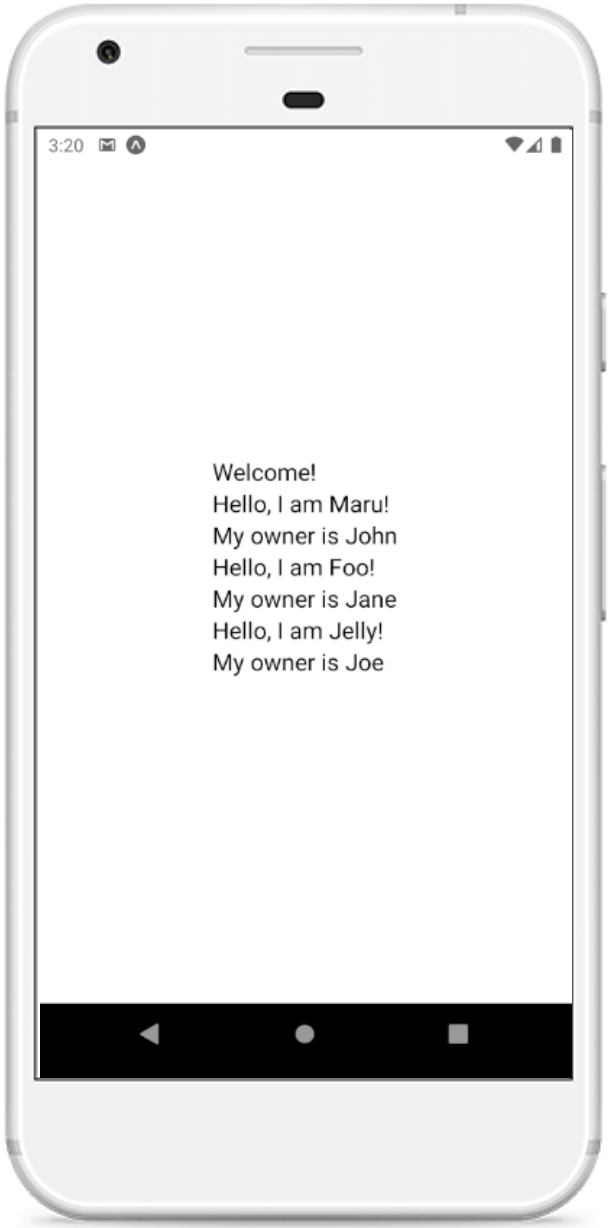
การส่งค่าจาก Parent -> Child Component

Parent Component (/components/Café.js)

```
import React from 'react';
import { Text, TextInput, View } from 'react-native’;
import Cat from './Cat'

const Cafe = () => {
  return (
    <View>
      <Text>Welcome!</Text>
      <Cat name ="Maru" owner="John" />
      <Cat name ="Foo" owner="Jane" />
      <Cat name ="Jelly" owner="Joe" />
</View>
  );
}
export default Cafe;
```

Child Component (/components/Cat.js)

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

const Cat = (props) => {
  return (
    <View>
      <Text>Hello, I am {props.name} !</Text>
      <Text>My owner is {props.owner}</Text>
    </View>
  );
}
export default Cat;
```


3:20
Welcome!
Hello, I am Maru!
My owner is John
Hello, I am Foo!
My owner is Jane
Hello, I am Jelly!
My owner is Joe

ผลลัพธ์

State

- State คือ ข้อมูลที่เก็บอยู่ในคอมโพเนนต์
- เมื่อค่าของ State มีการเปลี่ยนแปลง React Native จะทำการเรนเดอร์เนื้อหาที่อยู่ในคอมโพเนนต์นั้นใหม่ ซึ่งสามารถเปลี่ยนแปลงได้ตามค่าของ State ได้ทันที
- การกำหนด State สามารถทำได้ทั้งใน
  - Class Component
  - Function Component

State: Class Component

- ตัวอย่างการกำหนด state ให้กับคอมโพเนนต์
  - state = {
    name: ‘James’,
    gender: ‘male
    }
- การเรียกใช้ค่าใน state
  - this.state.name
  - this.state.gender
- การกำหนดค่าใหม่ใน state -> ใช้เมธอด setState()
  - this.setStage({name: ‘Jane’, gender: ‘female’})

State : Class Component (Cat.js)

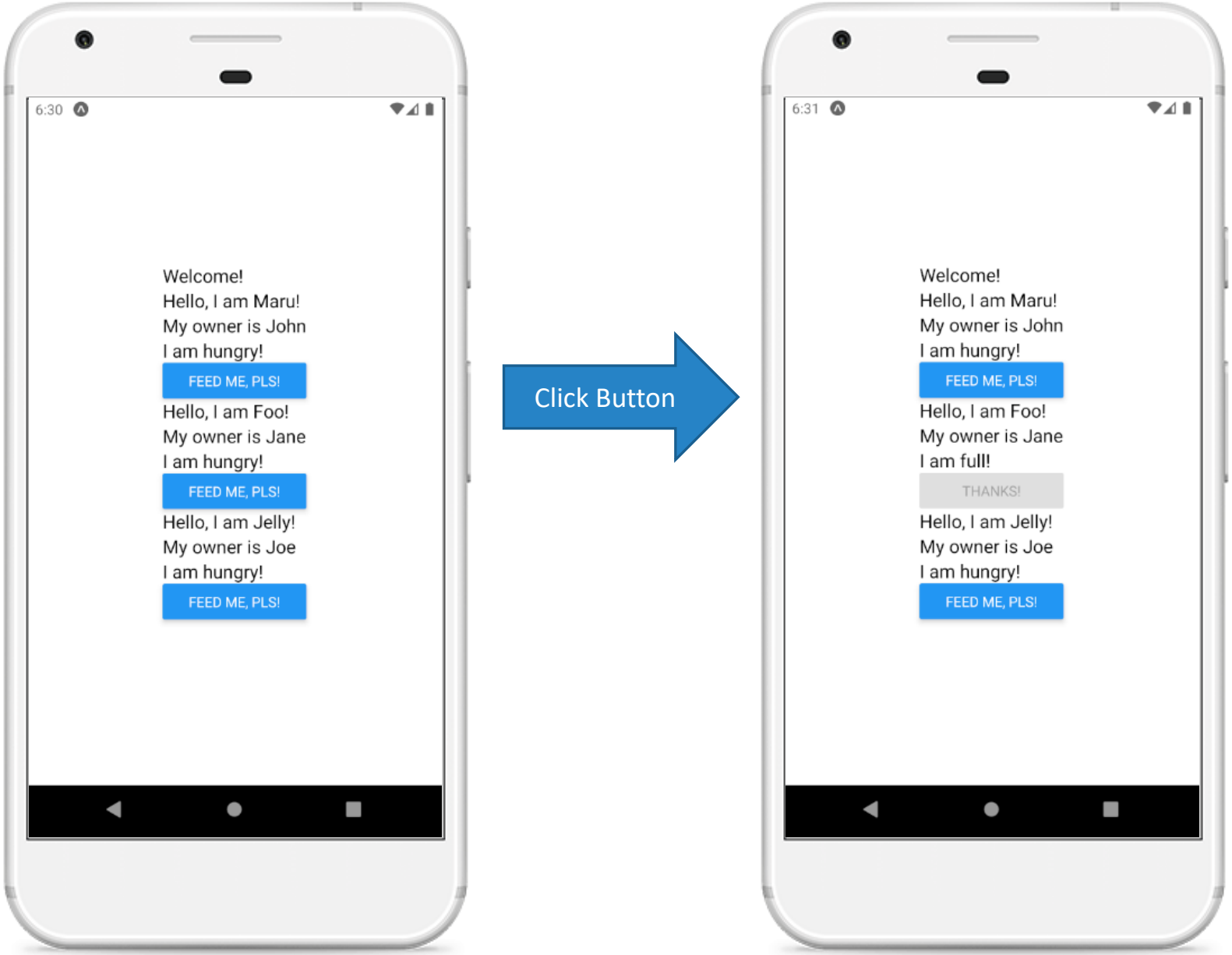
```
import React, { Component } from "react";
import { Button, Text, View } from "react-native";

class Cat extends Component {
 state = { isHungry: true };

 render(props) {
  return (
   <View>
    <Text>Hello, I am {this.props.name} !</Text>
    <Text>My owner is {this.props.owner}</Text>
    <Text>I am {this.state.isHungry ? " hungry" : " full"}!
    </Text>
    <Button
     onPress={() => {
      this.setState({ isHungry: false });
     }}
     disabled={!this.state.isHungry}
     title={
      this.state.isHungry ? "Feed me, pls!" : "Thanks!"
     }
    />
   </View>
  );
 }
}

export default  Cat;
```

State: Function Component

- ต้องทำการ import {useState} from ‘react’
- การกำหนด state ให้กับคอมโพเนนต์ -> ใช้ useState()
  - รูปแบบ : [*ตัวแปรข้อมูลใน state*, *ฟังก์ชันอัพเดทค่าข้อมูลใน state*] = useState(*ค่าเริ่มต้น*)
  - const [name, setName] = useState(‘James’);
  - const [gender, setGender] = useState(‘male’);
- การเรียกใช้ค่าใน state สามารถเรียกผ่านชื่อตัวแปรข้อมูลได้เลย
  - name หรือ gender
- การกำหนดค่าใหม่ใน state -> ใช้ฟังก์ชันที่กำหนดไว้ก่อนหน้านี้
  - setName(‘Jane’);
  - setGender(‘female’);

States: Function Component (Cat.js)

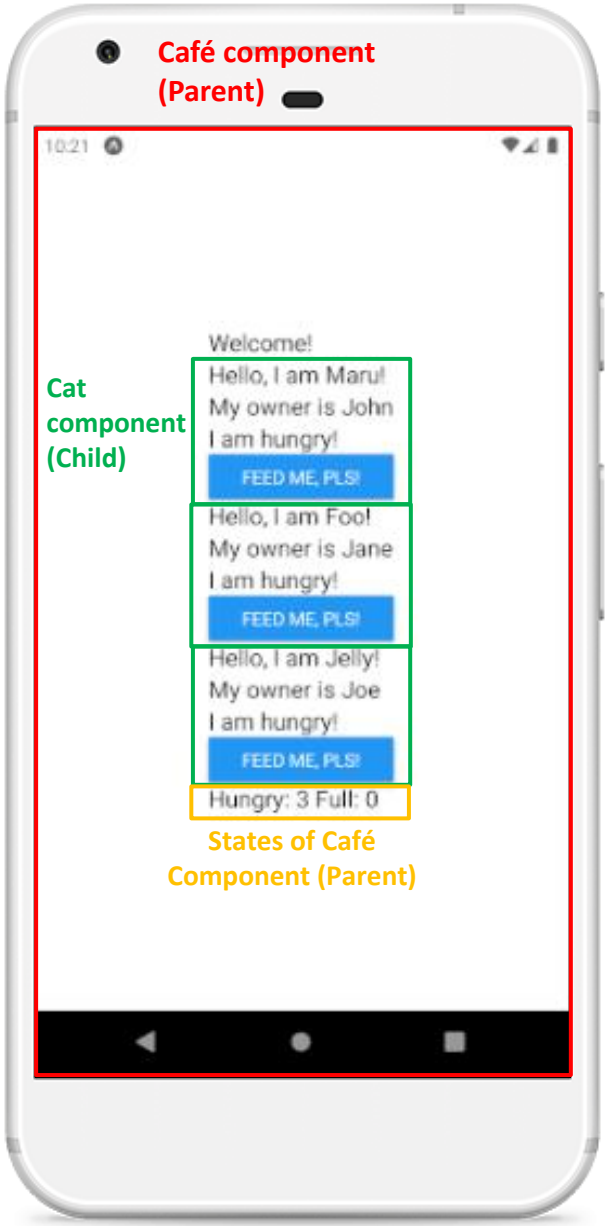
```
import React, { useState } from "react";
import { Button, Text, View } from "react-native";

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

 return (
  <View>
   <Text>Hello, I am {props.name} !</Text>
   <Text>My owner is {props.owner}</Text>
   <Text>I am {isHungry ? "hungry" : "full"}!</Text>
   <Button
    onPress={() => { setIsHungry(false);}}
    disabled={!isHungry}
    title={isHungry ? “Feed me, pls!" : "Thanks!"}
   />
  </View>
 );
}
export default Cat;
```


6:30
Welcome!
Hello, I am Maru!
My owner is John
I am hungry!
FEED ME, PLS!
Hello, I am Foo!
My owner is Jane
I am hungry!
FEED ME, PLS!
Hello, I am Jelly!
My owner is Joe
I am hungry!
FEED ME, PLS!
Click Button
6:31
Welcome!
Hello, I am Maru!
My owner is John
I am hungry!
FEED ME, PLS!
Hello, I am Foo!
My owner is Jane
I am full!
THANKS!
Hello, I am Jelly!
My owner is Joe
I am hungry!
FEED ME, PLS!

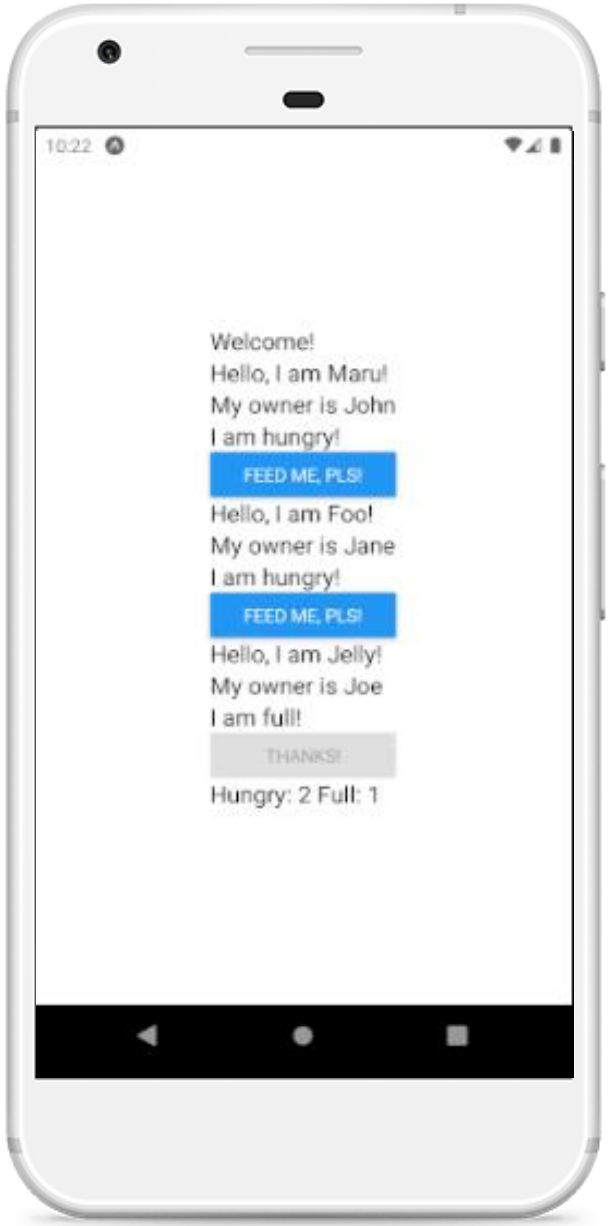
ผลลัพธ์

การติดต่อจาก Child -> Parent Component

- คอมโพเนนต์ลูกสามารถทำงานและติดต่อกลับไปยังคอมโพเนนต์แม่ได้ โดย
  - สร้างฟังก์ชันในคอมโพเนนต์แม่
  - ส่งฟังก์ชันนั้น เป็น props ให้กับคอมโพเนนต์ลูก
  - คอมโพเนนต์ลูกสามารถเรียกฟังก์ชันผ่าน props ซึ่งอาจมีการแก้ไขค่า state ของคอมโพเนนต์แม่ได้



เมื่อกดปุ่ม FEED ME, PLS! ในคอมโพเนนต์ลูก (Cat) จะทำการอัพเดทค่า state ใน คอมโพเนนต์แม่ (Cafe)



การติดต่อจาก Child -> Parent : Café component

- สร้าง state ใน Café component
  - const [hungryNum, setHungryNum] = useState(3);
  - const [fullNum, setFullNum] = useState(0);
- แสดงค่า state ใน <View> ของ Café component
  - <Text>Hungry: {hungryNum} Full: {fullNum}</Text>

การติดต่อจาก Child -> Parent : Café component

- สร้างฟังก์ชันใน Café component (เพื่อให้ Cat component เรียกใช้ภายหลัง)
  - const incFullHandler = () => {

        setFullNum(fullNum + 1);

        setHungryNum(hungryNum - 1);

    };
- ทำการส่งฟังก์ชันนั้น ผ่าน props เมื่อมีการเรียกคอมโพเนนต์ลูก (Cat component)
  - <Cat name="Maru" owner="John" onIncFull={incFullHandler} />

การติดต่อจาก Child -> Parent : Café component

- สร้างฟังก์ชันใน Café component (เพื่อให้ Cat component เรียกใช้ภายหลัง)
  - const incFullHandler = () => {

        setFullNum(fullNum + 1);

        setHungryNum(hungryNum - 1);

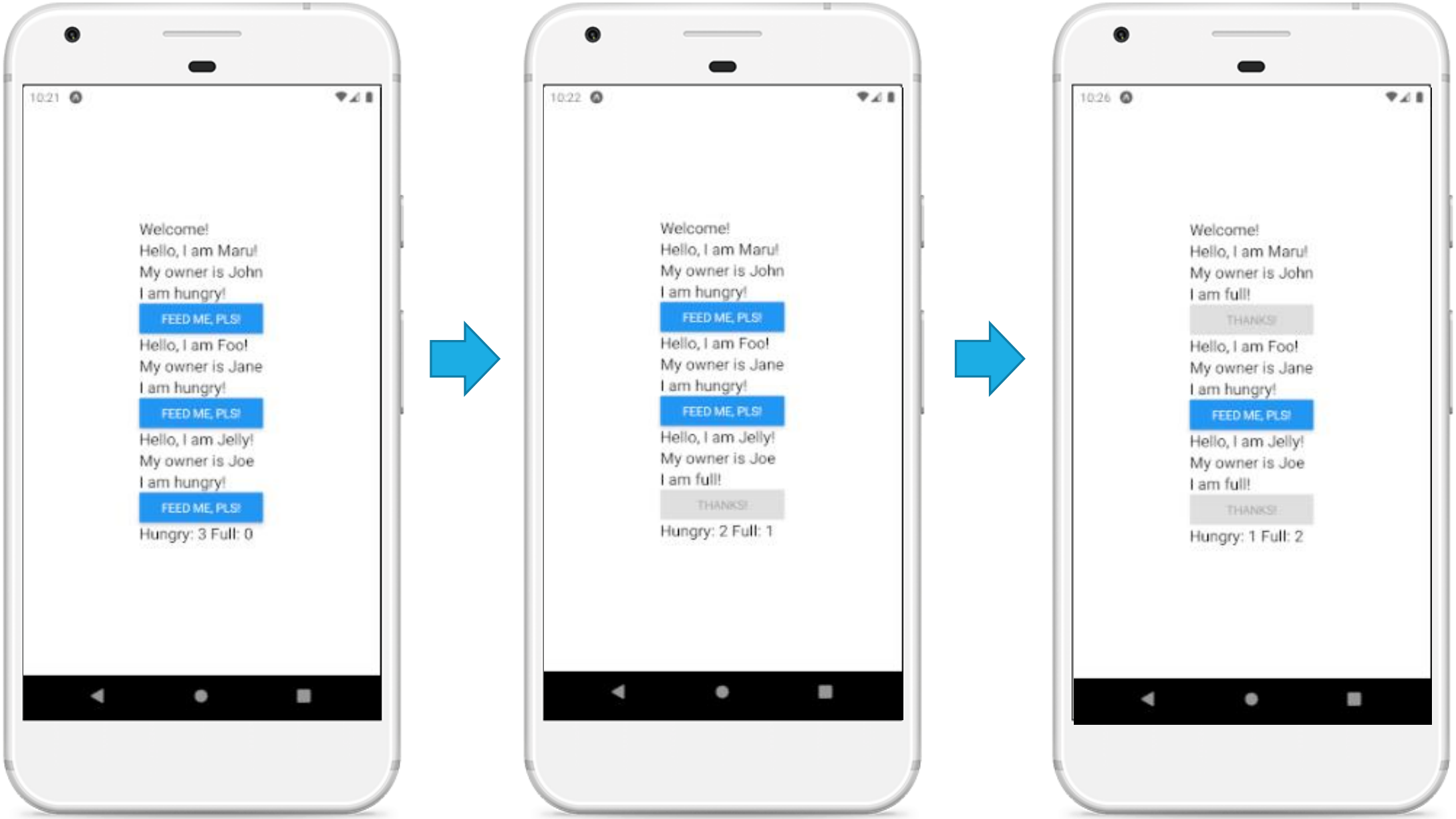
    };
- ทำการส่งฟังก์ชันนั้น ผ่าน props เมื่อมีการเรียกคอมโพเนนต์ลูก (Cat component)
  - <Cat name="Maru" owner="John" onIncFull={incFullHandler} />

การติดต่อจาก Child -> Parent : Cat component

- ทำการเรียกฟังก์ชันของคอมโพเนนต์แม่ (Cafe) ผ่าน props ได้
  - <Button

```
onPress={() => {
    setIsHungry(false);
    props.onIncFull();
}}
disabled={!isHungry}
title={isHungry ? "Feed me, pls!" : "Thanks!"}
```

/>


10:21
Welcome!
Hello, I am Maru!
My owner is John
I am hungry!
FEED ME, PLS!
Hello, I am Foo!
My owner is Jane
I am hungry!
FEED ME, PLS!
Hello, I am Jelly!
My owner is Joe
I am hungry!
FEED ME, PLS!
Hungry: 3 Full: 0
10:22
Welcome!
Hello, I am Maru!
My owner is John
I am hungry!
FEED ME, PLS!
Hello, I am Foo!
My owner is Jane
I am hungry!
FEED ME, PLS!
Hello, I am Jelly!
My owner is Joe
I am full!
THANKS!
Hungry: 2 Full: 1
10:26
Welcome!
Hello, I am Maru!
My owner is John
I am full!
THANKS!
Hello, I am Foo!
My owner is Jane
I am hungry!
FEED ME, PLS!
Hello, I am Jelly!
My owner is Joe
I am full!
THANKS!
Hungry: 1 Full: 2