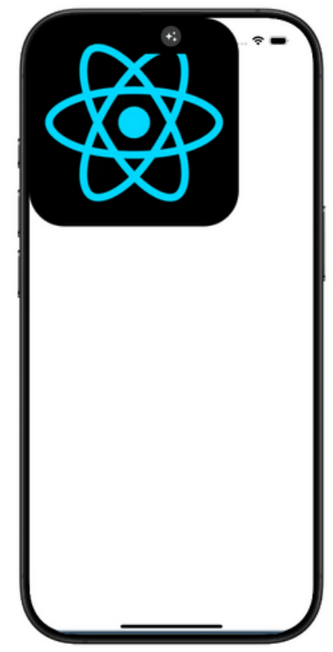
# Image

React Native การแสดงรูปภาพสามารถทำได้
โดยใช้คอมโพเนนต์ <Image>
ซึ่งเป็นส่วนหนึ่งของคอมโพเนนต์หลักที่ React Native มีให้คอมโพเนนต์นี้รองรับการโหลดรูปภาพจากทั้งแหล่งข้อมูล ภายในแอป และจาก URL

## <Image>:

ในโค้ดด้านบน คอมโพเนนต์ <Image> จะโหลดรูปภาพจาก URL ที่กำหนด และแสดงผล

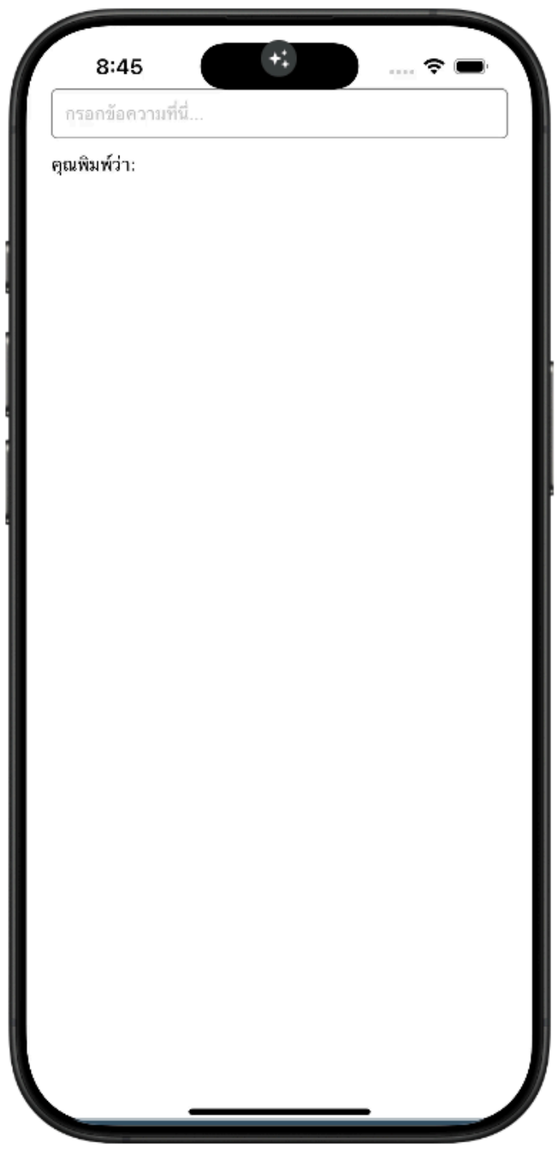



## การโหลดรูปภาพจากไฟล์ภายในโปรเจกต์

# TextInput

React Native สามารถใช้คอมโพเนนต์ <TextInput> เพื่อรับข้อมูลจากผู้ใช้ได้

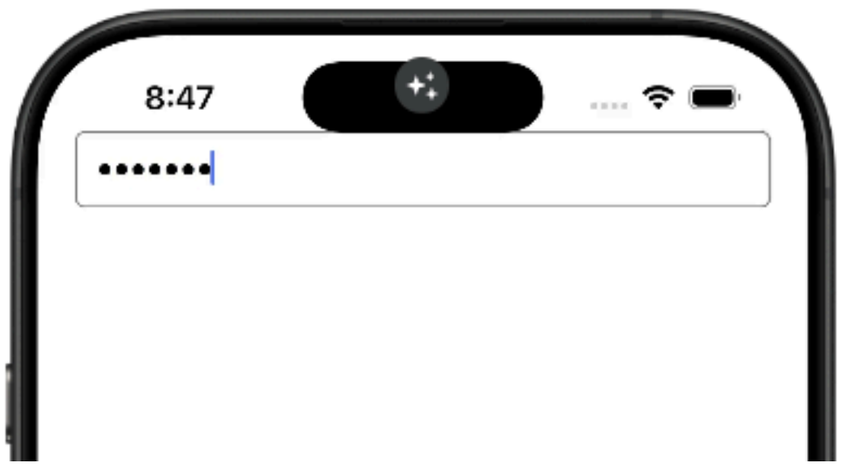
```jsx
import React, { useState } from 'react';
import { TextInput, View, Text, StyleSheet } from 'react-native';

const MyTextInput = () => {
  const [text, setText] = useState('');

  return (
    <View style={styles.container}>
      <TextInput
        style={styles.input}
        placeholder="กรอกข้อความที่นี่..."
        value={text}
        onChangeText={(value) => setText(value)}
      />
      <Text>คุณพิมพ์ว่า: {text}</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 20,
    paddingTop:50,
  },
  input: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,
    paddingHorizontal: 10,
    marginBottom: 10,
    borderRadius: 5,
  },
});

export default MyTextInput;
```




- placeholder="กรอกข้อความที่นี่..." → แสดงข้อความแนะนำในช่องป้อนข้อมูล
- value={text} → กำหนดค่าข้อความจาก state
- onChangeText={(value) => setText(value)} → เมื่อมีการพิมพ์ข้อมูล จะอัปเดต state text
- Text → แสดงข้อความที่ผู้ใช้พิมพ์

(secureTextEntry)ใช้สำหรับซ่อนรหัสผ่านในการป้อนข้อมูล
หากต้องการให้ช่องอินพุตใช้สำหรับรหัสผ่าน สามารถใช้ secureTextEntry={true} ได้:

```jsx
return (
  <View style={styles.container}>
    <TextInput
      style={styles.input}
      placeholder="กรอกรหัสผ่าน"
      secureTextEntry={true}
    />
  </View>
);
```



# กำหนดประเภทของคีย์บอร์ด (keyboardType)

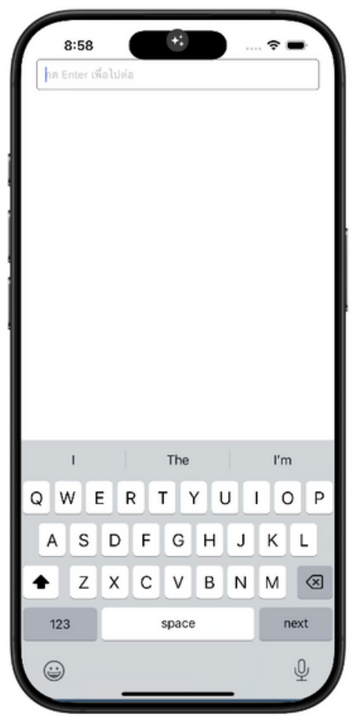
สามารถกำหนดประเภทของคีย์บอร์ดได้ เช่น
keyboardType="email-address" → สำหรับอีเมล
keyboardType="numeric" → สำหรับตัวเลข
keyboardType="phone-pad" → สำหรับเบอร์โทรศัพท์

```
return (
  <View style={styles.container}>
    <TextInput
      style={styles.input}
      placeholder="กรอกอีเมล"
      keyboardType="email-address"
    />
  </View>
```


(returnKeyType)
สามารถกำหนดปุ่มที่ใช้ปิดคีย์บอร์ดได้ เช่น:

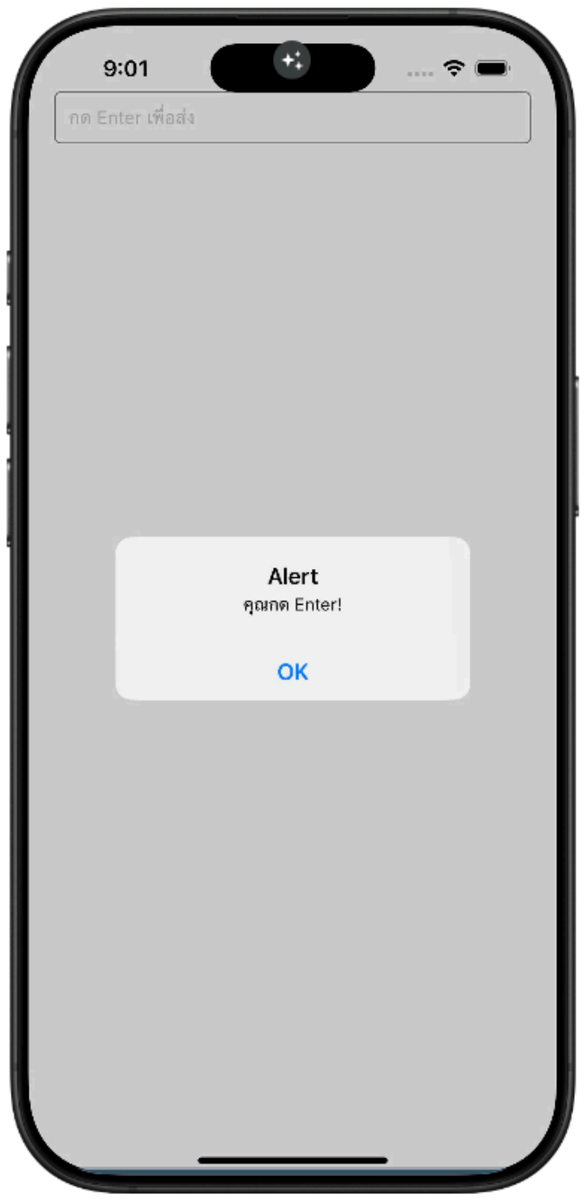
1. returnKeyType="done"
2. returnKeyType="next"

```
<View style={styles.container}>
  <TextInput
    style={styles.input}
    placeholder="กด Enter เพื่อไปต่อ"
    returnKeyType="next"
  />
</View>
```



การจัดการ Event (onSubmitEditing)
หากต้องการให้แอปทำงานเมื่อกดปุ่ม Enter หรือ Done บนคีย์บอร์ด ใช้ onSubmitEditing ได้:

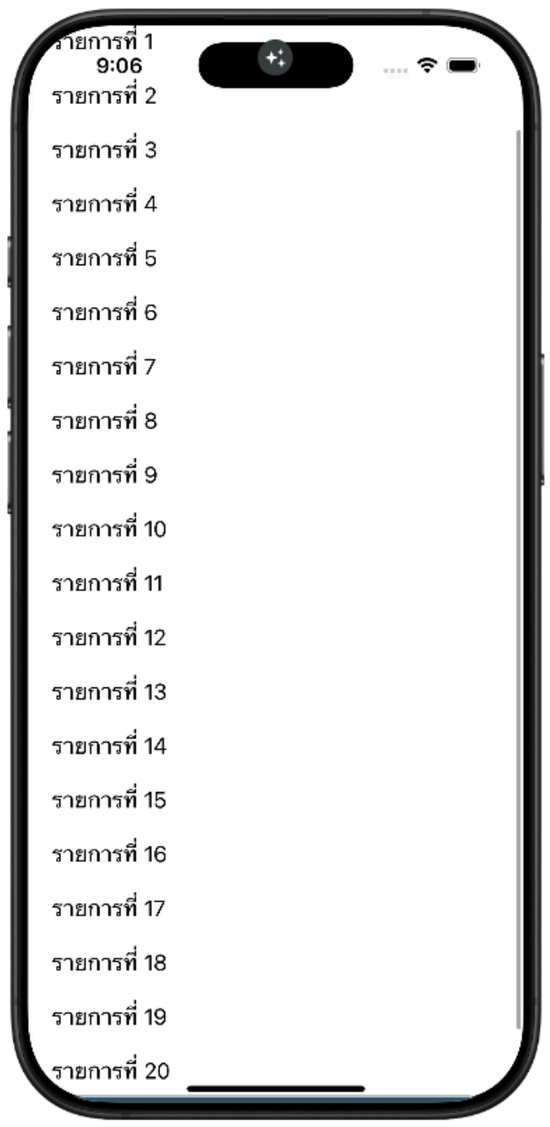
```
return (
  <View style={styles.container}>
    <TextInput
      style={styles.input}
      placeholder="กด Enter เพื่อส่ง"
      onSubmitEditing={() => alert('คุณกด Enter!')}
    />
  </View>
```



# คอมโพเนนต์ <ScrollView>

ใช้สำหรับเลื่อนเนื้อหาภายในแอปเมื่อข้อมูลมีขนาดใหญ่เกินกว่าที่จะแสดงในหน้าจอได้

```jsx
import React from 'react';
import { ScrollView, Text, StyleSheet } from 'react-native';

const MyScrollView = () => {
  return (
    <ScrollView style={styles.container}>
      {Array.from({ length: 20 }, (_, i) => (
        <Text key={i} style={styles.item}>
          รายการที่ {i + 1}
        </Text>
      ))}
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 20,
  },
  item: {
    fontSize: 18,
    marginVertical: 10,
  },
});

export default MyScrollView;
```




- สร้าง <ScrollView> ห่อรายการ <Text>
- ใช้ Array.from() เพื่อสร้างรายการจำลอง 20 รายการ
- style={styles.container} → กำหนดสไตล์ให้กับ <ScrollView>

การเลื่อนแนวนอน (horizontal={true})
โดยปกติ <ScrollView> เลื่อนในแนวตั้ง แต่หากต้องการเลื่อนในแนวนอน สามารถกำหนด horizontal={true} ได้:

```jsx
import React from 'react';
import { ScrollView, Text, StyleSheet } from 'react-native';

const MyScrollView = () => {
  return (
    <ScrollView horizontal={true} style={{ padding: 10 }}>
      {Array.from({ length: 10 }, (_, i) => (
        <Text key={i} style={{ fontSize: 18, marginHorizontal: 20 }}>
          ไอเท็ม {i + 1}
        </Text>
      ))}
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 20,
  },
  item: {
    fontSize: 18,
    marginVertical: 10,
  },
});

export default MyScrollView;
```


ใช้ horizontal={true} เพื่อให้ ScrollView เลื่อนไปทางขวาแทน

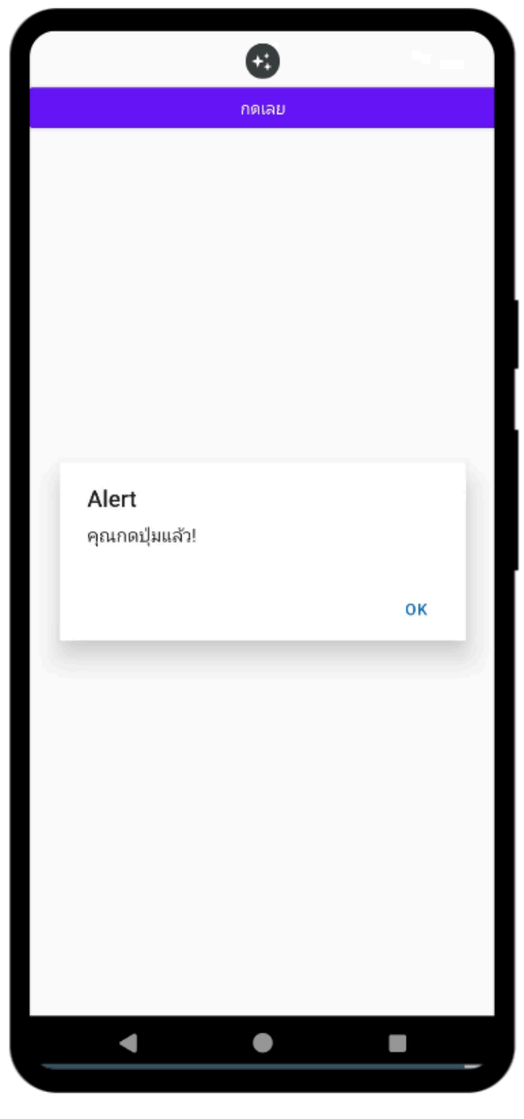
Scroll Indicator (showsVerticalScrollIndicator & showsHorizontalScrollIndicator)
หากต้องการซ่อนแถบเลื่อน สามารถตั้งค่า false ได้:
<ScrollView showsVerticalScrollIndicator={false} showsHorizontalScrollIndicator={false}>

การรีเฟรชเมื่อเลื่อนลง (refreshControl)
สามารถใช้ RefreshControl เพื่อทำให้ผู้ใช้สามารถเลื่อนลงเพื่อโหลดข้อมูลใหม่ได้:

เมื่อผู้ใช้เลื่อนลง (pull to refresh) → จะโหลดข้อมูลใหม่หลังจาก 2 วินาที

# Button

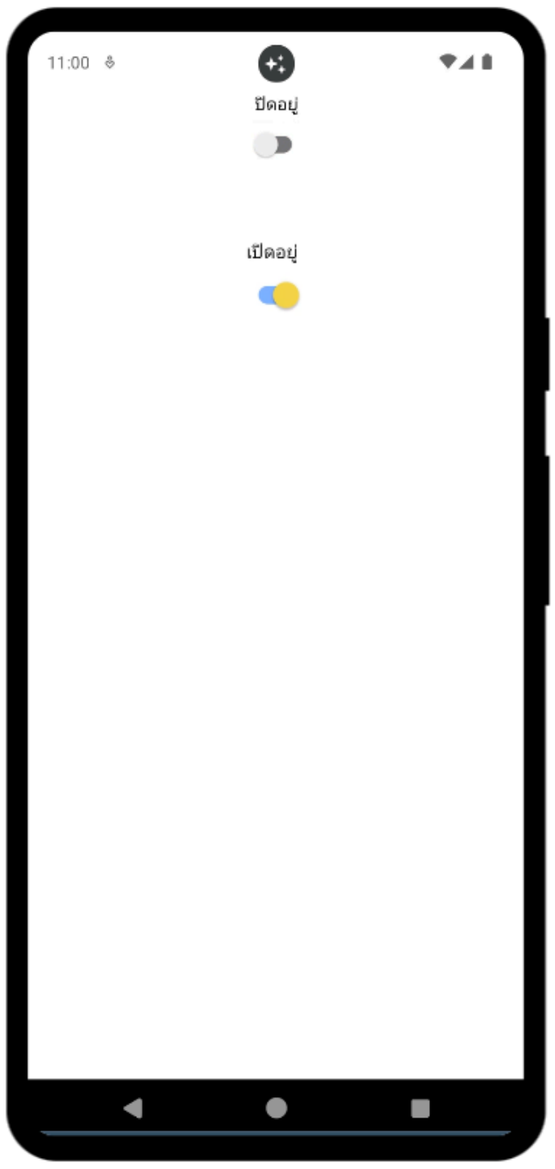
```jsx
import React from 'react';
import { Button, View } from 'react-native';

export default function App() {
  return (
    <View style={{ marginTop: 50 }}>
      <Button
        title="กดเลย"
        onPress={() => alert('คุณกดปุ่มแล้ว!')}
        color="#6200EE"
      />
    </View>
  );
}
```




- title: ข้อความบนปุ่ม
- onPress: ฟังก์ชันที่เรียกเมื่อกด
- color: สีของปุ่ม (เฉพาะ Android)

# Switch

```jsx
import React, { useState } from 'react';
import { View, Text, Switch, StyleSheet } from 'react-native';

export default function App() {
  const [isEnabled, setIsEnabled] = useState(false);

  const toggleSwitch = () => setIsEnabled(previousState => !previousState);

  return (
    <View style={styles.container}>
      <Text>{isEnabled ? 'เปิดอยู่' : 'ปิดอยู่'}</Text>
      <Switch
        trackColor={{ false: '#767577', true: '#81b0ff' }}
        thumbColor={isEnabled ? '#f5dd4b' : '#f4f3f4'}
        onValueChange={toggleSwitch}
        value={isEnabled}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    marginTop: 50,
    alignItems: 'center',
  },
});
```




- value: กำหนดสถานะเปิด/ปิด (true/false)
- onValueChange: ฟังก์ชันที่ทำงานเมื่อมีการเปลี่ยนสถานะ
- thumbColor: สีของปุ่มวงกลม
- trackColor: สีของพื้นหลังของ switch (เปิด/ปิด)