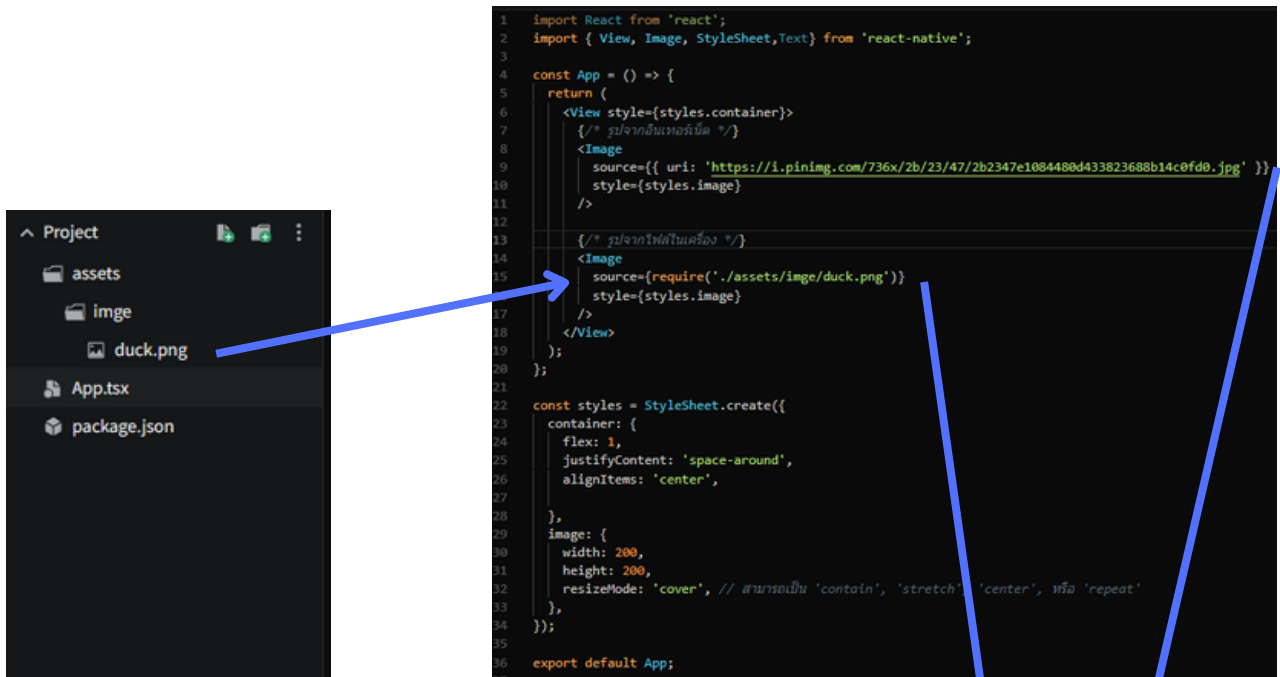


Image

เป็นคอมโพเนนต์ที่ใช้ในการแสดงผลรูปภาพในแอปพลิเคชันที่รองรับการโหลดรูปภาพจากหลากหลายแหล่ง เช่น ไฟล์ในเครื่อง (local files) หรือ URL บนอินเทอร์เน็ต นอกจากนี้ยังมีพรีโพรเซสเซอร์และคุณสมบัติที่ช่วยในการปรับแต่งภาพได้หลากหลาย เช่น การปรับขนาด (resize), การใส่สไตล์, การเพิ่ม placeholder และอื่นๆ

วิธีการใช้งาน Image ใน React Native



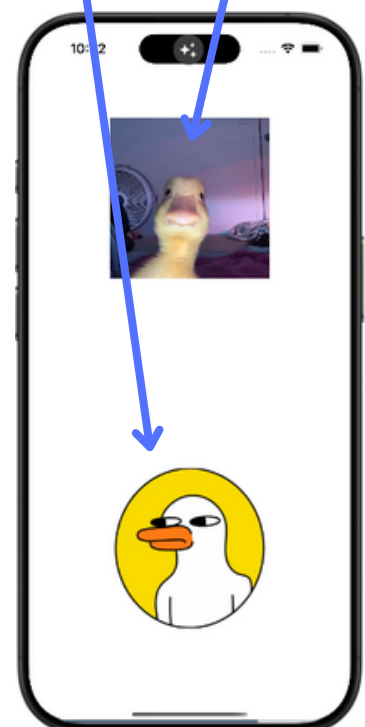
กำหนดแหล่งของรูปภาพ

รูปจากไฟล์ในเครื่อง:

- `require('./path-to-image.png')`

รูปจาก URL:

- `{ uri: 'https://example.com/image.jpg' }`



Props สำคัญของ Image

source

- กำหนดแหล่งของรูปภาพ เช่น:
 - รูปจากไฟล์ในเครื่อง: `require('./path-to-image.png')`
 - รูปจาก URL: `{ uri: 'https://example.com/image.jpg' }`

style

- ใช้กำหนดขนาดและรูปแบบ เช่น กำหนดความกว้าง (width), ความสูง (height), และอื่น ๆ

resizeMode

- ใช้สำหรับควบคุมวิธีการปรับขนาดของภาพในกรอบ เช่น:
 - 'cover': เติบโตเต็มพื้นที่โดยไม่ยืดรูป
 - 'contain': ให้อยู่ในกรอบโดยรักษาสัดส่วน
 - 'stretch': ยืดรูปให้เต็มพื้นที่
 - 'center': แสดงรูปในตำแหน่งกลาง

onLoad / onError

- ใช้จัดการ event เช่น ตรวจสอบเมื่อรูปโหลดสำเร็จหรือมีข้อผิดพลาด

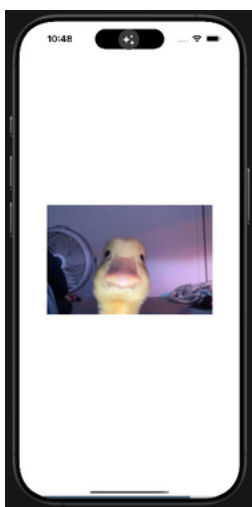
defaultSource

- (เฉพาะ iOS) ใช้แสดงรูป placeholder ขณะรอโหลดรูปหลัก

code

`defaultSource={require('./placeholder-image.png')}`

ตัวอย่างการใช้งาน Props



```
1 import React from 'react';
2 import { View, Image, StyleSheet } from 'react-native';
3
4
5 const App = () => {
6   return (
7     <View style={styles.container}>
8       <Image
9         source={{ uri: 'https://i.pinimg.com/736x/2b/23/47/2b2347e1884488d433823688b14c0fd0.jpg' }}
10         style={styles.image}
11         resizeMode="cover"
12         blurRadius={2}
13         onLoad={() => console.log('Image loaded!')}
14         onError={(e) => console.log('Error loading image', e)}
15       />
16     </View>
17   );
18 };
19
20 const styles = StyleSheet.create({
21   container: {
22     flex: 1,
23     justifyContent: 'center',
24     alignItems: 'center',
25   },
26   image: {
27     width: 300,
28     height: 200,
29   },
30 });
31
32 export default App;
33
```

PROBLEMS LOGS

iPhone 16 Pro: source.uri should not be an empty string

iPhone 16 Pro: Image loaded! 5

✓ No errors

TextInput

TextInput เป็นคอมโพเนนต์สำหรับสร้างช่องป้อนข้อมูล (Input Field) ในแอปพลิเคชัน React Native โดยสามารถใช้เพื่อให้ผู้ใช้งานป้อนข้อความ เช่น ชื่อ, รหัสผ่าน, อีเมล หรือข้อมูลอื่น ๆ ได้

```
1 import React, { useState } from 'react';
2 import { View, TextInput, StyleSheet } from 'react-native';
3
4 const App = () => {
5   const [text, setText] = useState('');
6
7   return (
8     <View style={styles.container}>
9       <TextInput
10         style={styles.input}
11         placeholder="Type here..."
12         value={text}
13         onChangeText={setText}
14       />
15     </View>
16   );
17 };
18
19 const styles = StyleSheet.create({
20   container: {
21     flex: 1,
22     justifyContent: 'center',
23     alignItems: 'center',
24   },
25   input: {
26     height: 40,
27     borderColor: 'gray',
28     borderWidth: 1,
29     padding: 10,
30     width: '80%',
31     borderRadius: 5,
32   },
33 });
34
35 export default App;
```



ตัวอย่างการใช้งาน Props เพิ่มเติม

```
1 import React, { useState } from 'react';
2 import { View, TextInput, StyleSheet } from 'react-native';
3
4 const App = () => {
5   const [text, setText] = useState('');
6
7   return (
8     <View style={styles.container}>
9       <TextInput
10         style={styles.input}
11         placeholder="Enter your email"
12         placeholderTextColor="gray"
13         value={text}
14         onChangeText={setText}
15         keyboardType="email-address"
16         autoCapitalize="none"
17         returnKeyType="done"
18         secureTextEntry={false}
19         maxLength={50}
20         onSubmitEditing={() => console.log('Submitted:', text)}
21       />
22     </View>
23   );
24 };
25
26 const styles = StyleSheet.create({
27   container: {
28     flex: 1,
29     justifyContent: 'center',
30     alignItems: 'center',
31   },
32   input: {
33     height: 50,
34     borderColor: 'gray',
35     borderWidth: 1,
36     padding: 10,
37     width: '80%',
38     borderRadius: 5,
39   },
40 });
41
42 export default App;
```



กำหนดชนิดของคีย์บอร์ด เป็น

- 'email-address': คีย์บอร์ดอีเมล

Props สำคัญของ TextInput

1. value

- กำหนดค่าเริ่มต้นของข้อความในช่องป้อนข้อมูล

2. onChangeText

- ฟังก์ชันที่ทำงานเมื่อผู้ใช้งานพิมพ์ข้อความ
- `onChangeText={(text) => console.log(text)}`

3. placeholder

- ข้อความที่แสดงในช่องป้อนข้อมูลเมื่อยังไม่มีข้อความ
- `placeholder="Enter your name"`

4. placeholderTextColor

- กำหนดสีของข้อความ placeholder
- `placeholderTextColor="gray"`

5. keyboardType

- กำหนดชนิดของคีย์บอร์ด เช่น ตัวเลข, อีเมล, หรือข้อความทั่วไป
- 'default': คีย์บอร์ดปกติ
- 'numeric': ตัวเลข
- 'email-address': คีย์บอร์ดอีเมล
- 'phone-pad': คีย์บอร์ดเบอร์โทรศัพท์
- ตัวอย่าง:
- `keyboardType="email-address"`

6. secureTextEntry

- ใช้สำหรับช่องป้อนข้อมูลรหัสผ่าน (ซ่อนข้อความที่พิมพ์)
- `secureTextEntry={true}`

7. multiline

- กำหนดให้ช่องป้อนข้อมูลรองรับหลายบรรทัด
- `multiline={true}`

8. maxLength

- กำหนดจำนวนตัวอักษรสูงสุดที่สามารถพิมพ์ได้
- `maxLength={50}`

9. editable

- กำหนดให้ช่องป้อนข้อมูลสามารถแก้ไขได้หรือไม่ (ค่าเริ่มต้น: true)
- `editable={false}` // ช่องจะไม่สามารถแก้ไขได้

10. autoCapitalize

- กำหนดการใช้ตัวพิมพ์ใหญ่เมื่อเริ่มต้นพิมพ์ข้อความ
- 'none': ไม่แปลงเป็นตัวพิมพ์ใหญ่
- 'sentences': ตัวพิมพ์ใหญ่เฉพาะคำแรกในประโยค
- 'words': ตัวพิมพ์ใหญ่ทุกคำ
- 'characters': ตัวพิมพ์ใหญ่ทุกตัวอักษร
- ตัวอย่าง:
- `autoCapitalize="none"`

11. returnKeyType

- กำหนดปุ่ม Return/Done ในคีย์บอร์ด เช่น 'done', 'next', 'search', 'go'
- `returnKeyType="done"`

12. onSubmitEditing

- ฟังก์ชันที่ทำงานเมื่อผู้ใช้กดปุ่ม Return/Enter
- `onSubmitEditing={() => console.log('Submitted!')}`

ScrollView

React Native, ScrollView คือคอมโพเนนต์ที่ใช้สร้างพื้นที่ที่สามารถเลื่อนดูเนื้อหาที่มีขนาดเกินหน้าจอได้ ใช้ได้ดีกับข้อมูลขนาดเล็กหรือเลย์เอาต์ที่ต้องการการเลื่อนในแนวตั้งหรือแนวนอน

ScrollView:

การเลื่อนในแนวตั้งและแนวนอน:

- ใช้ horizontal เพื่อเลื่อนในแนวนอน

ปรับการเลื่อน:

- เช่น `contentContainerStyle`, `onScroll`, `scrollEnabled`,