國立高雄科技大學 智慧商務系
National Kaohsiung University of Science and Technology Department of Intelligent Commerce

# React Native 基本介紹 I

廖奕雯
yiwen923@nkust.edu.tw

# 大綱

- 簡介
- JSX
- 組件 Components

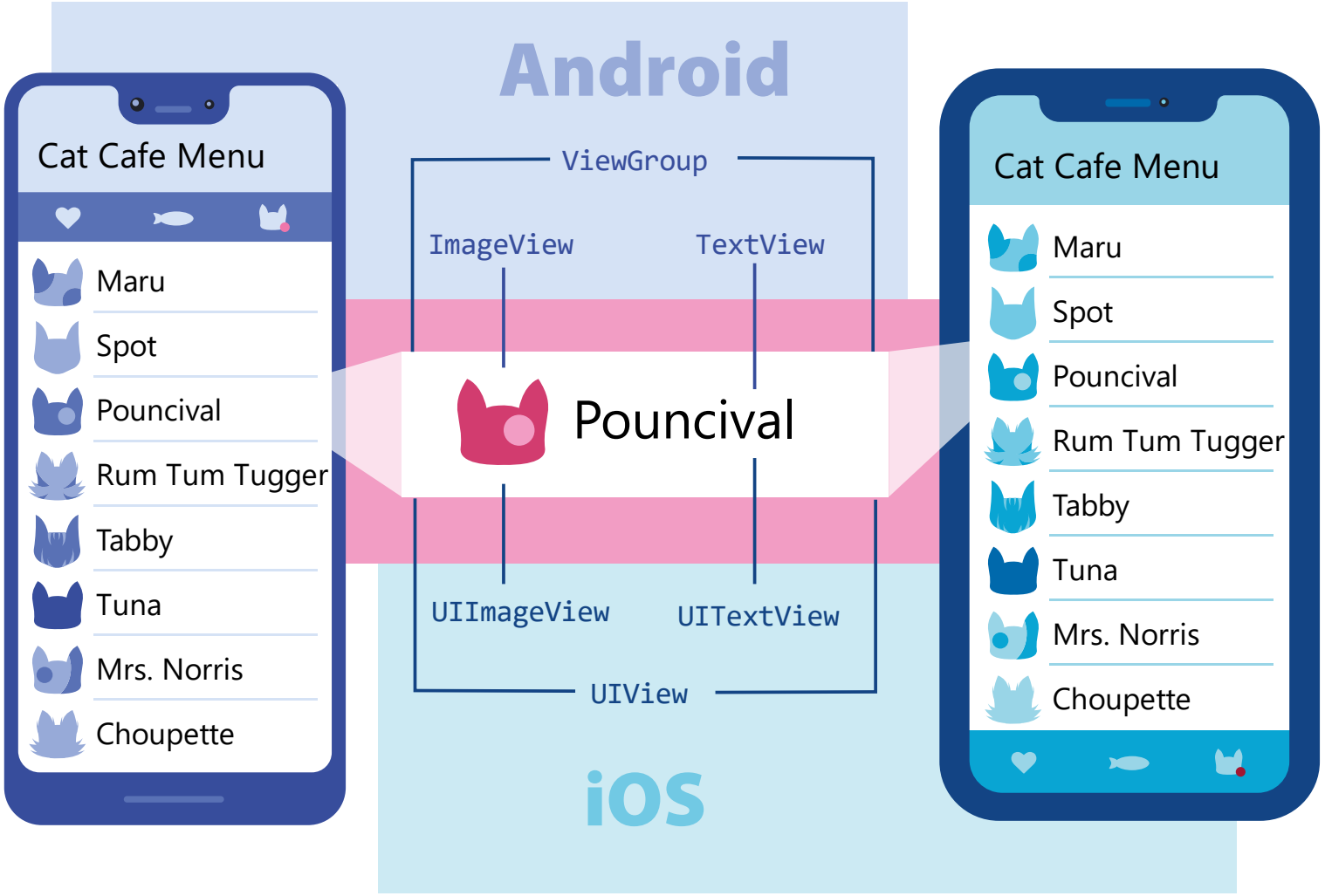
# React Native

- React Native 是一個使用React和app平臺的原生功能來構建 Android 和 iOS 應用的open source framework。
- 透過 React Native，您可以使用 JavaScript 來訪問app平臺的 API，以及使用 React components(組件) 來描述 UI 的外觀和行為：一系列可重用、可嵌套 (nested) 的代碼。

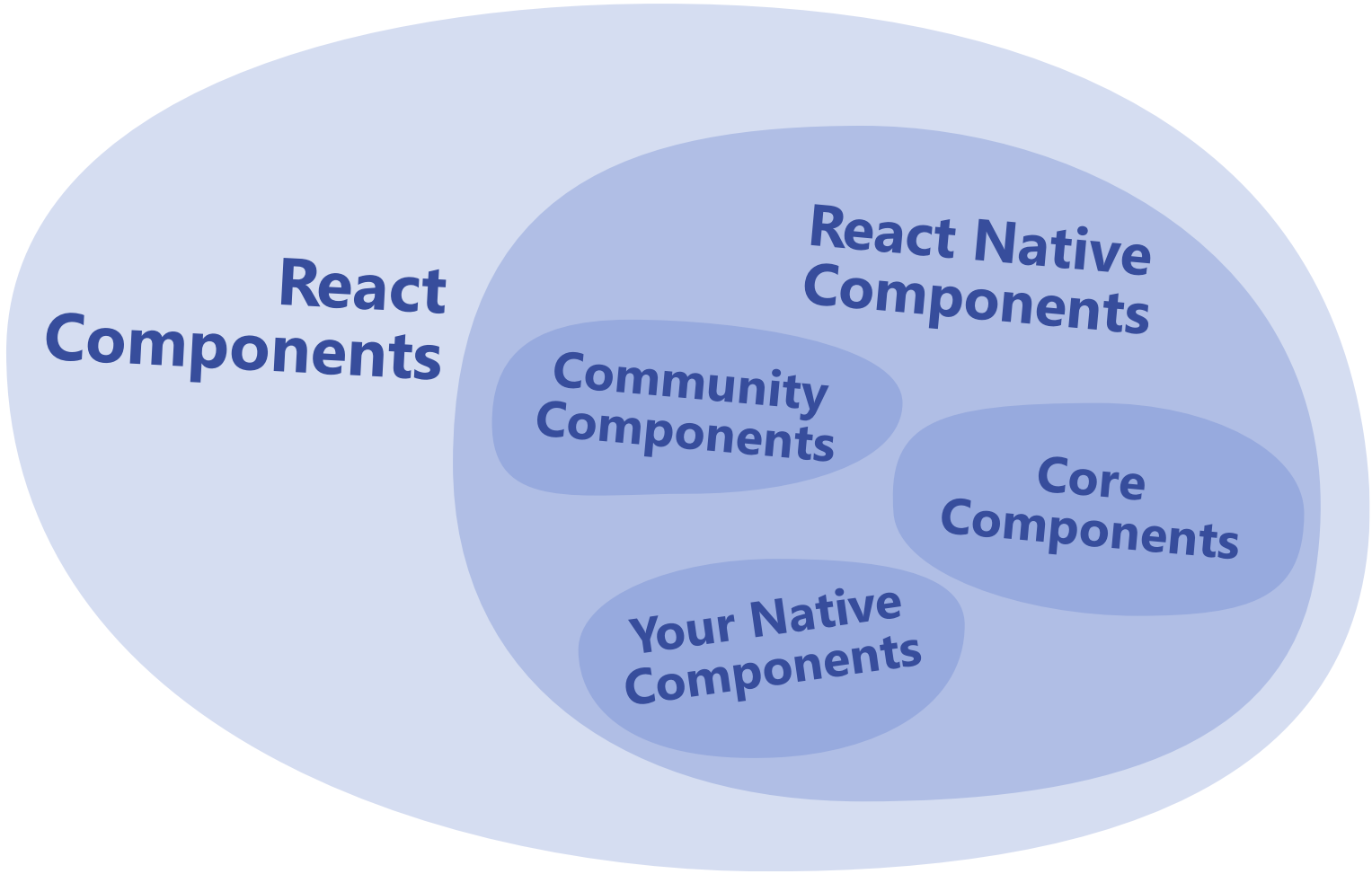
# 原生組件與核心組件

- 原生組件(Native Components)
  - 在 Android 開發中是使用 Kotlin 或 Java 來編寫 view；在 iOS 開發中是使用 Swift 或 Objective-C 來編寫 view。
  - 在 React Native 中，則使用 React 元件通過 JavaScript 來建立 view。
  - 在 app 運行時，React Native 會為這些組件建立相應的 Android 和 iOS view。
  - 由於 React Native 組件就是對原生 view 的封裝，因此使用 React Native 編寫的應用外觀、感覺和性能與其他任何原生應用一樣。我們將這些平臺支援的組件稱為**原生組件**。
- 核心組件 (Core Components)
  - React Native 具有許多核心元件，從controls 到 activity indicators，應有盡有。

# A sampling of the many views used in Android and iOS apps.



# React Native and React components



# App.js

File: App.js

```
01: import { StatusBar } from 'expo-status-bar';
02: import { StyleSheet, Text, View } from 'react-native';
03:
04: export default function App() {
05:   return (
06:     <View style={styles.container}>
07:       <Text>Open up App.js to start working on your app!</Text>
08:       <StatusBar style="auto" />
09:     </View>
10:   );
11: }
12:
13: const styles = StyleSheet.create({
14:   container: {
15:     flex: 1,
16:     backgroundColor: '#fff',
17:     alignItems: 'center',
18:     justifyContent: 'center',
19:   },
20: });
21:
```

一開始先引入 react native 和 expo 的 components

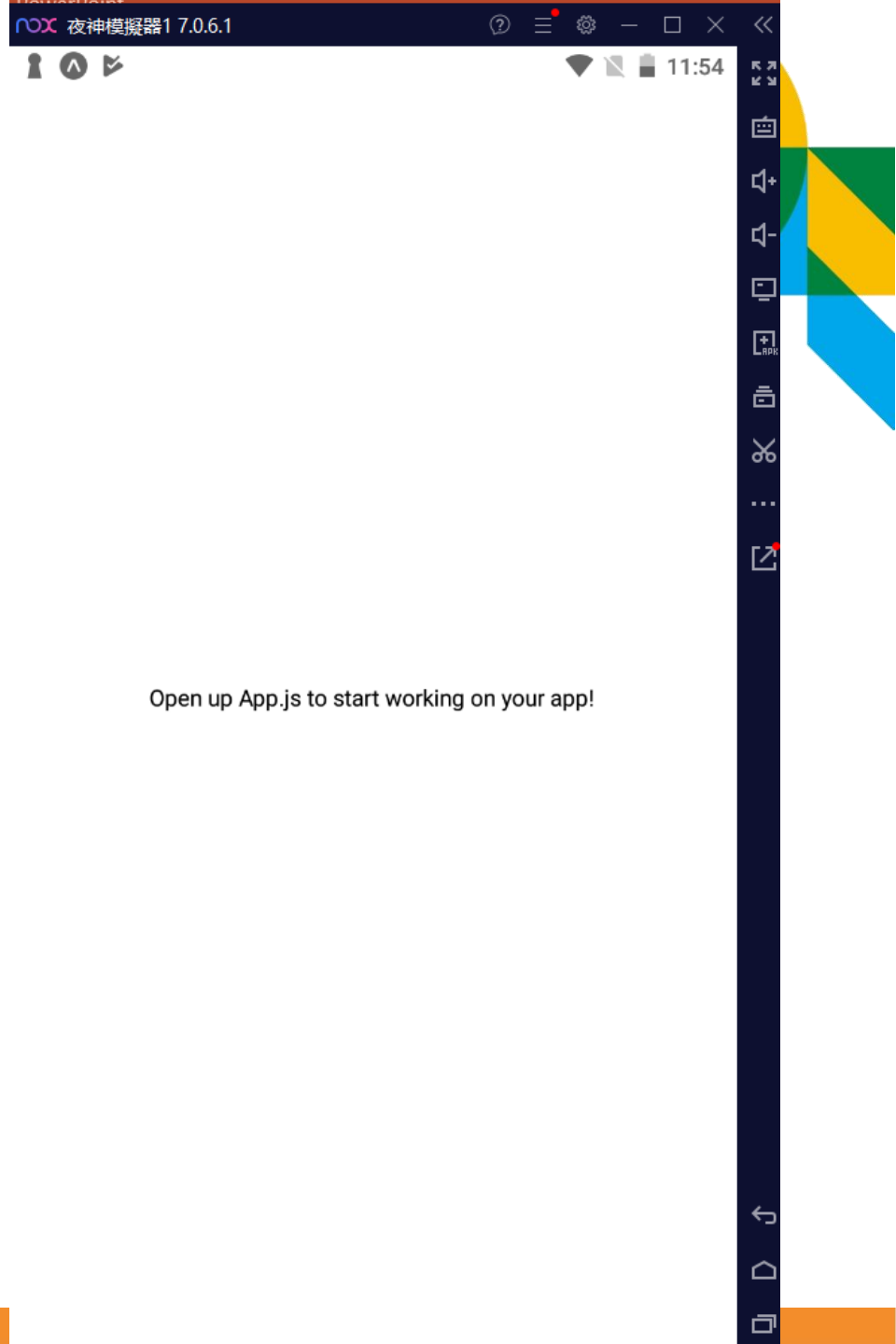
App view 的程式碼。
UI 的程式碼要寫在 return(); 當中。

App 的 style 的程式碼

# native05_01

JS App.js > ...

```
1   File: App.js
2   import { StatusBar } from 'expo-status-bar';
3   import { StyleSheet, Text, View } from 'react-native';
4
5   export default function App() {
6    return (
7      <View style={styles.container}>
8        <Text>Open up App.js to start working on your app!</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  });
```



# Class Component vs. Functional Component

## Class-based Component

```
class App extends React.component {
    render () { return </> }
}
```

- 需繼承React.Component
- 具有生命週期，可以針對某些情境決定是否渲染，例如: shouldComponentUpdate()
- 具有state (Stateful component)
- 需要實作render方法
- 擁有this
- 每次都可以拿到最新的this.props，因為this隨時都在變化，

## Function Component

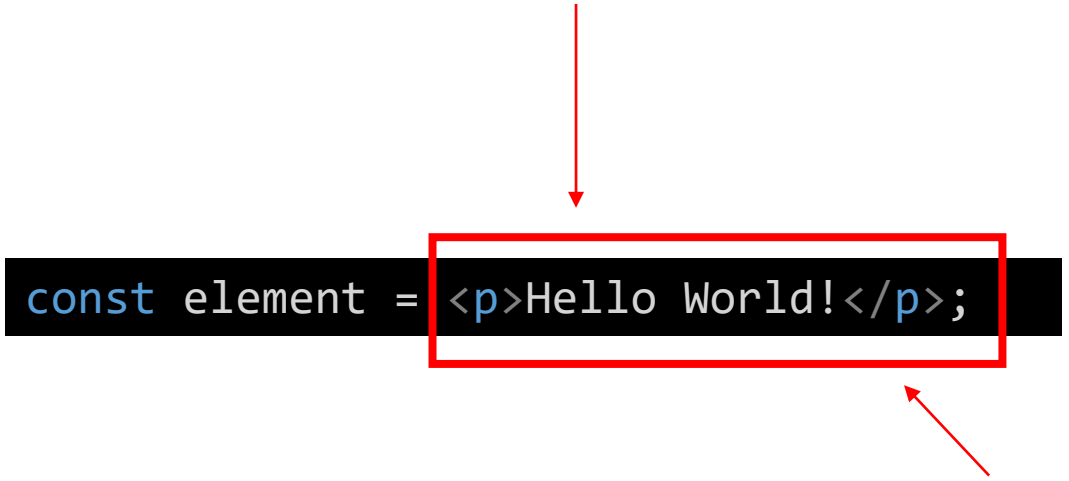
```
function App () {
}
```

- 沒有生命週期（React Hook userEffect 出現後，就有生命週期了！）
- 沒有state（Stateless），所以被稱為無狀態組件（但React Hook useState出現後就可以有state了！）
- 可以用arrow function 宣告或是一般的function
- 沒有this
- 編譯更快（因為不用將class轉換成es5)
- props會一直是原本傳進來的那個，而不會跟著更新，閉包的概念

# JSX

# JSX

- 在 React 中，我們這樣寫，就是 JSX




- 沒錯，JSX可視為加強版的HTML，除了讓你把原來的 HTML放到JavaScript中，還可以讓你在HTML裡面呼叫 JavaScript。

# 在 JSX 中帶入變數

- 在JSX 中如果你要帶入變數，就用 { } 把 JavaScript的變數包起來。
- 像這樣:

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

# 在 JSX 中帶入表達式(expression)

- 首先，什麼是表達式，就是可以一段有回傳值的程式碼
- 例如: 2+2 ，5<10...等，有回傳值的程式碼；呼叫一個有回傳值的函式也可以。

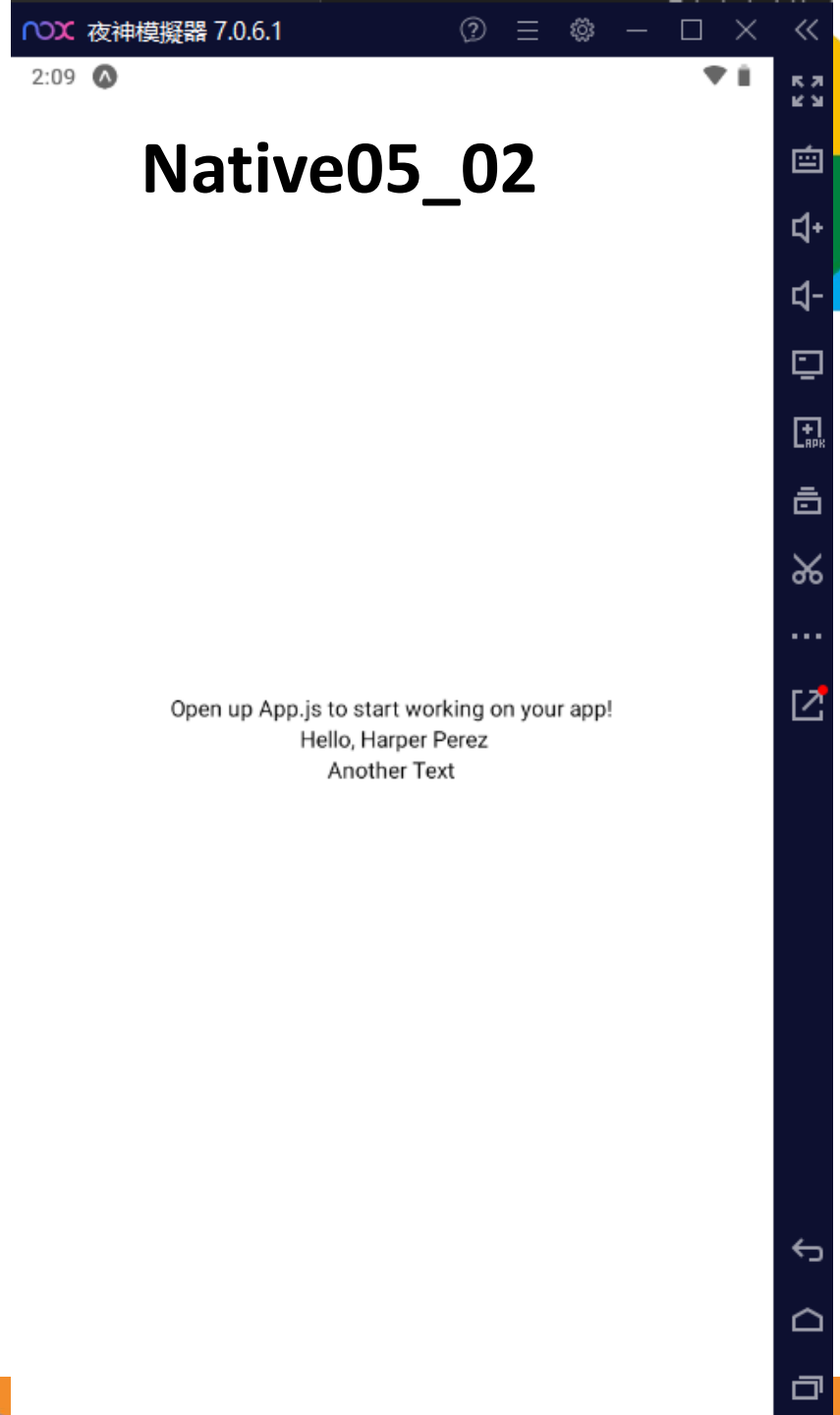
- if...else，while 就不是一個表達式，叫做陳述式。
- 一定要有回傳值是因為JSX是要在網頁顯示的，沒有回傳值，代表沒有內容可以顯示呀....

# 在 JSX 中帶入表達式(expression)

- 例如

```
function formatName(user) {
    return user.firstName+ ' ' + user.lastName;
  }

  const user = {
    firstName: 'Harper',
    lastName: 'Perez'
  };

  const element = (
    <h1>
      Hello, {formatName(user)}!
    </h1>
  );
```

```jsx
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

function formatName(user) {
    return user.firstName+ ' ' + user.lastName;
}

const user = {
firstName: 'Harper',
lastName: 'Perez'
};

export default function App() {
  return (
      <View style={styles.container}>
          <Text>Open up App.js to start working on your app!</Text>
          <Text> Hello, {formatName(user)}</Text>
          <Text>Another Text</Text>
          <StatusBar style="auto" />
      </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



# 在 JSX 中使用 CSS

- 同樣可以在 JSX 中使用 CSS，但是因為 class 是 React.js 關鍵字，所以使用 className 取代 CSS 中的 class

**普通的 CSS**　　把 class改成 className　　**JSX裡面的 CSS**

```
<h1 class="big">NKUST</h1>
```

```
<h1 className="big">NKUST</h1>
```

# 如果 JSX 的程式碼太長

- 可以把 JSX 的部分指令寫成一個變數，然後代入。

```
var myStyle = {
    fontSize: 100,
    color: '#FF0000'
};

const element = <h1 style = {myStyle}>直接寫 JSX 太長了</h1>
```

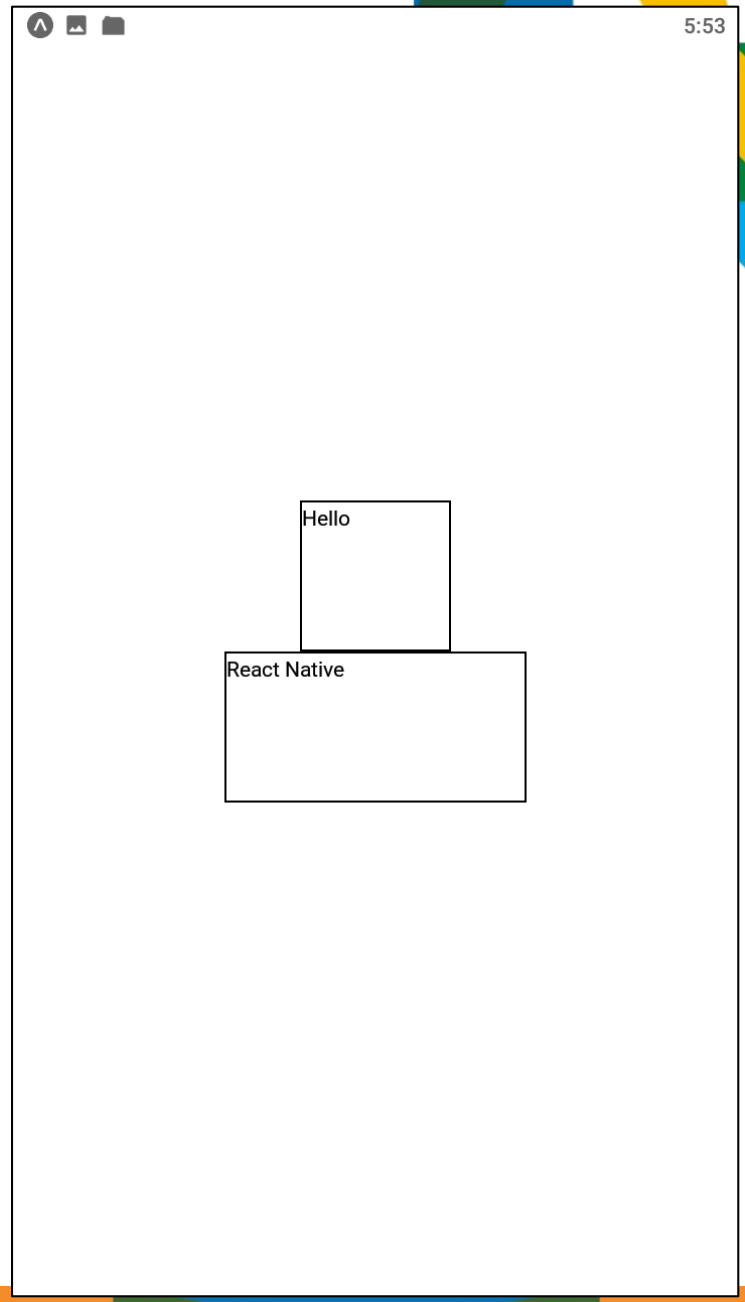
# View

- View 是 React native 建立 UI 介面時，最基礎的 component，就像 HTML 中的 div 一樣，主要負責：排版、樣式設計、事件處理...等。

# 組件 Components

# View

```
06:     <View style={styles.container}>
07:       <View style={{ height:100, width:100, borderWidth: 1}}>
08:         <Text>Hello</Text>
09:       </View>
10:       <View style={{ height:100, width:200, borderWidth: 1}}>
11:         <Text>React Native</Text>
12:       </View>
13:     </View>
```




- 一個 View component 包覆兩個 View components
- 外層的 View 使用之前就設定好的 style；另外兩個 View 分別設定 style。
- 並且在 View 中顯示 Text

```
File: App.js
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

function formatName(user) {
    return user.firstName+ ' ' + user.lastName;
}

const user = {
firstName: 'Harper',
lastName: 'Perez'
};

export default function App() {
  return (
    <View style={styles.container}>
      <View style={{ height:100, width:100, borderWidth: 1}}>
        <Text> React Native </Text>
      </View>
      <View style={{ height:100, width:200, borderWidth: 1}}>
        <Text> Hello, {formatName(user)}! </Text>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

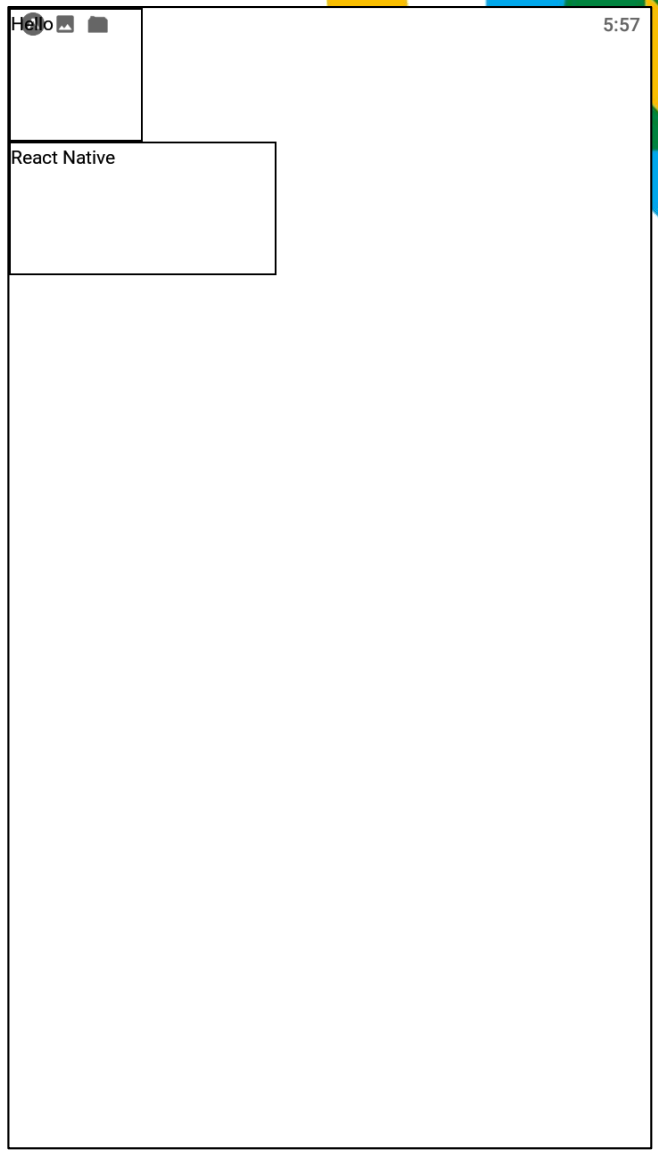
夜神模擬器 7.0.6.1

2:34

# Native05_05

# View

```
File: App.js
06:      <View style={{flex: 1}}>
07:        <View style={{ height:100, width:100, borderWidth: 1}}>
08:          <Text>Hello</Text>
09:        </View>
10:        <View style={{ height:100, width:200, borderWidth: 1}}>
11:          <Text>React Native</Text>
12:        </View>
13:      </View>
```

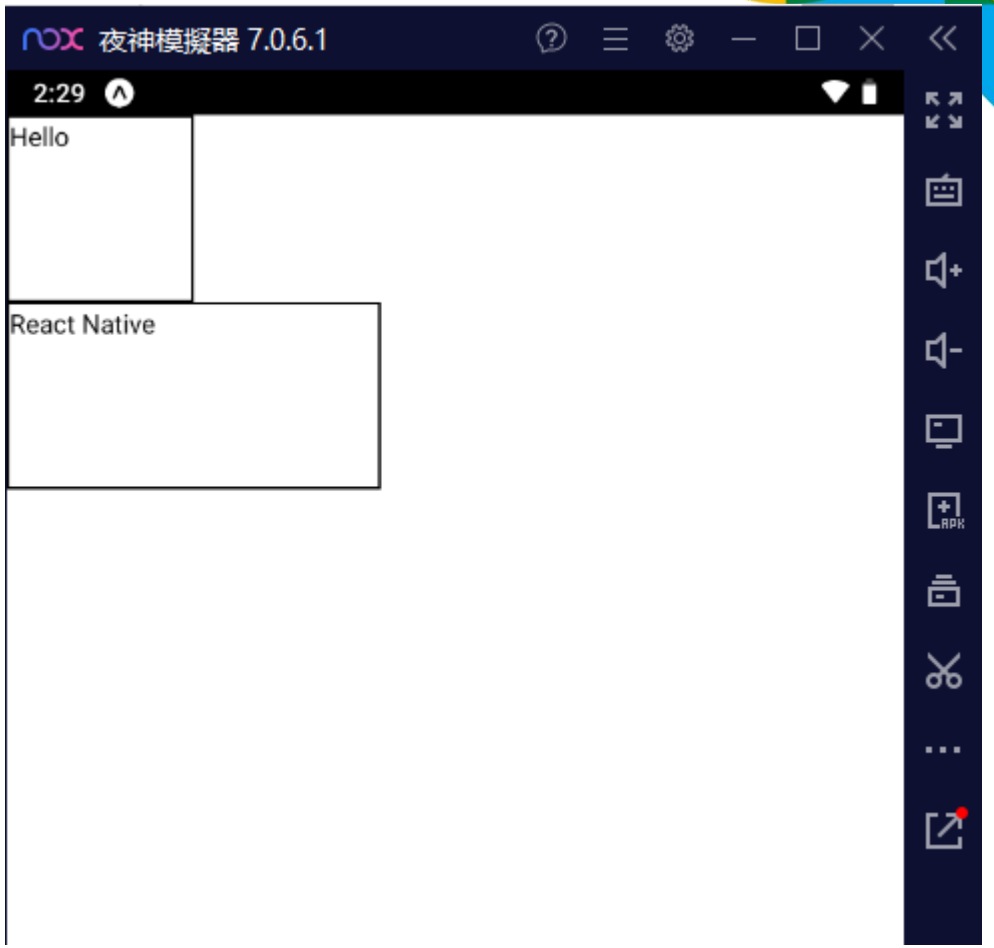



- 將外層 View 的 style 修改為 {{flex: 1}} 。
- 因為 外層 View 沒有設定置中，所以另外兩個 View就對齊左上角。

- flex:1 代表全螢幕。

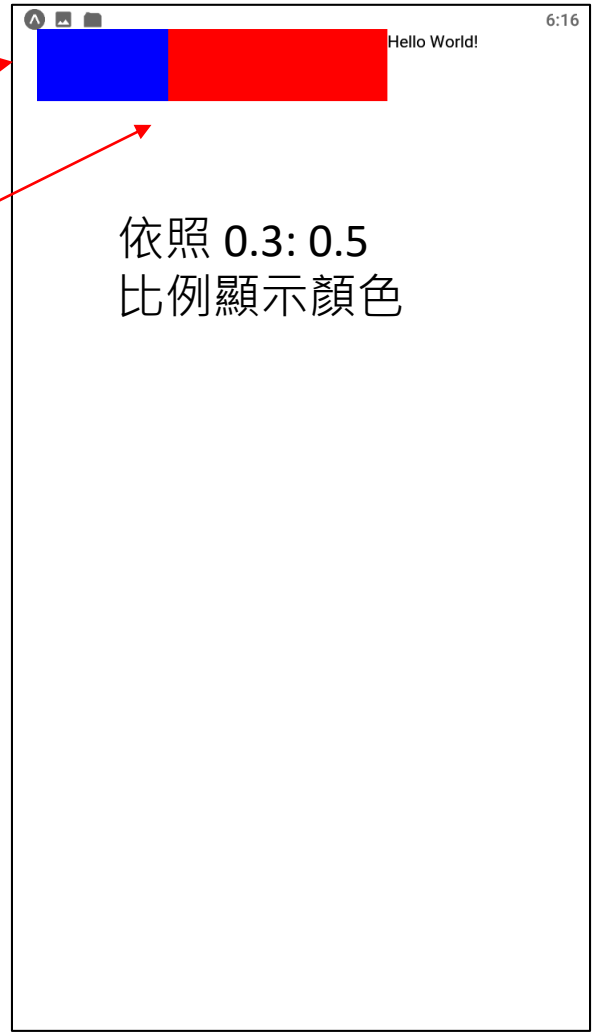
# Native05_04

```
1
2   import { StatusBar } from 'expo-status-bar';
3   import { StyleSheet, Text, View } from 'react-native';
4
5   function formatName(user) {
6       return user.firstName+ ' ' + user.lastName;
7   }
8
9   const user = {
10  firstName: 'Harper',
11  lastName: 'Perez'
12  };
13
14
15  export default function App() {
16    return (
17      <View style={{flex: 1}}>
18        <View style={{ height:100, width:100, borderWidth: 1}}>
19          <Text>Hello</Text>
20        </View>
21        <View style={{ height:100, width:200, borderWidth: 1}}>
22          <Text>React Native</Text>
23        </View>
24      </View>
25    );
26  }
```



# 如何避免 View 與 status bar 重疊

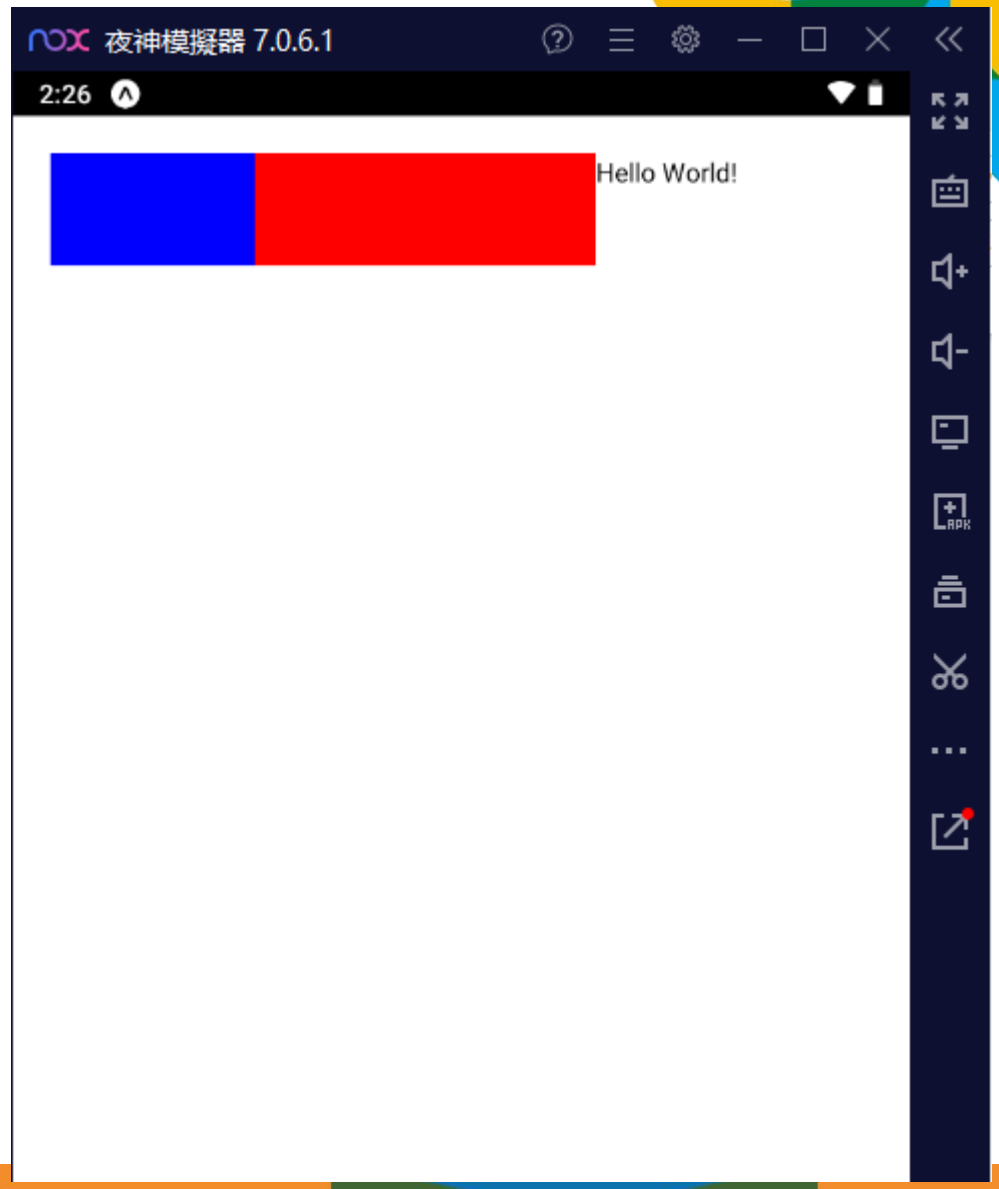
```
File: App.js
06:      <View
07:        style={{
08:          flexDirection: "row",
09:          height: 100,
10:          padding: 20
11:        }}
12:      >
13:        <View style={{ backgroundColor: "blue", flex: 0.3 }} />
14:        <View style={{ backgroundColor: "red", flex: 0.5 }} />
15:        <Text>Hello World!</Text>
16:      </View>
```




- 在 iOS 11 以上，可以使用 SafeAreaView 來避免 View 和 上方工具列重疊。
- 在 Android 上，可以自行使用 padding 來避免重疊。

# Native05_03

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

function formatName(user) {
    return user.firstName+ ' ' + user.lastName;
}

const user = {
firstName: 'Harper',
lastName: 'Perez'
};

export default function App() {
  return (
    <View
      style={{
      flexDirection: "row",
      height: 100,
        padding: 20
      }}
    >
    <View style={{ backgroundColor: "blue", flex: 0.3 }} />
      <View style={{ backgroundColor: "red", flex: 0.5 }} />
      <Text>Hello World!</Text>
    </View>

  );
}
```

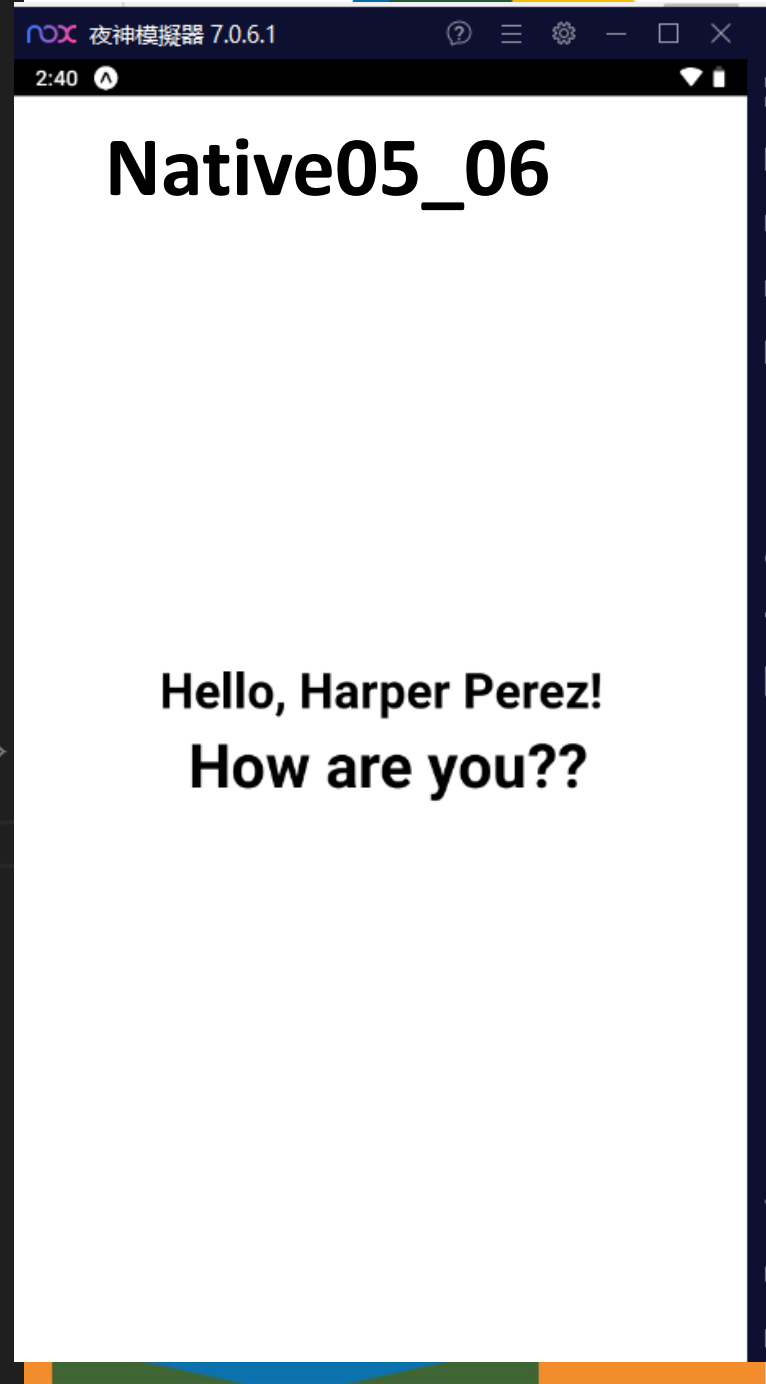


# Text

- 主要負責在畫面上顯示文字

```
File: App.js
6:      <View style={styles.container}>
7:        <Text style={{fontSize: 40,fontWeight: "bold"}}>How are you??</Text>
8:      </View>
```

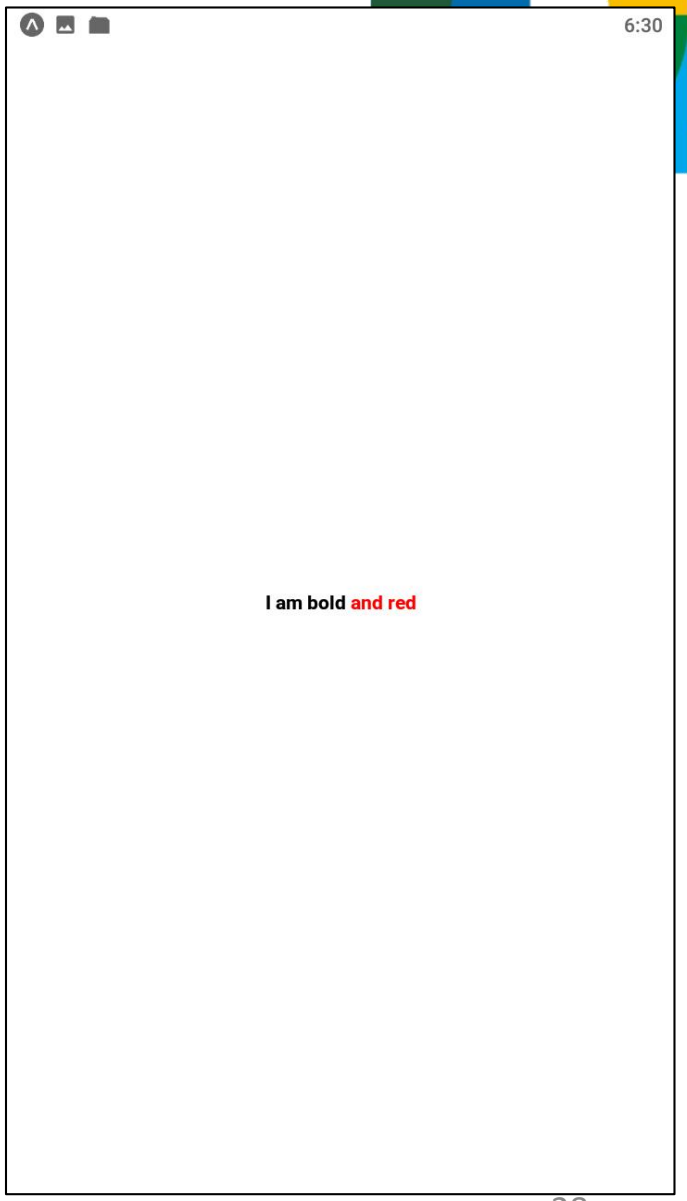
```
File: App.js
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

function formatName(user) {
    return user.firstName+ ' ' + user.lastName;
}

const user = {
firstName: 'Harper',
lastName: 'Perez'
};

export default function App() {
  return (
    <View style={styles.container}>
      <Text style={{fontSize: 32,fontWeight: "bold"}}>Hello, {formatName(user)}!</Text>
      <Text style={{fontSize: 40,fontWeight: "bold"}}> How are you??</Text>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



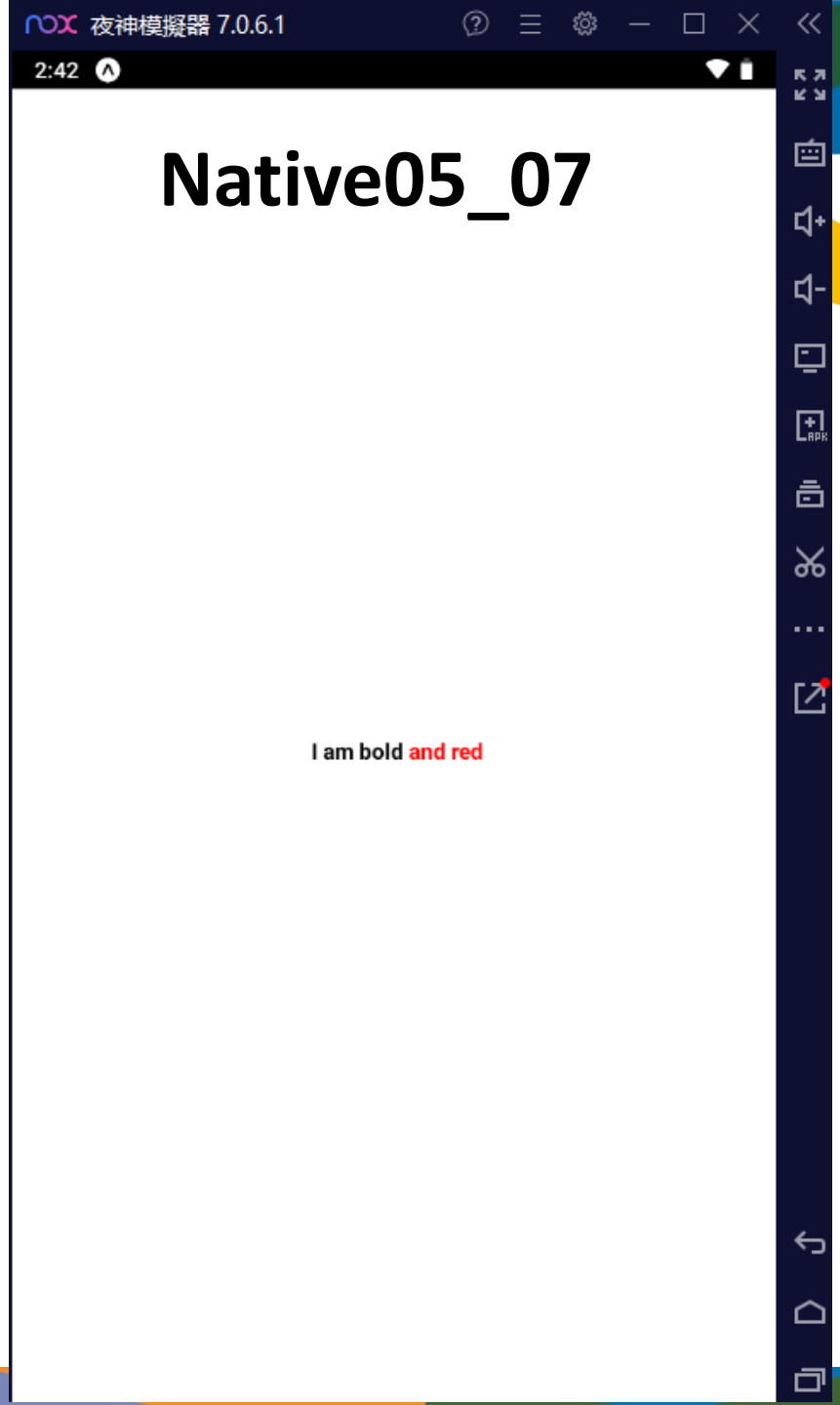
# Text

- 巢狀 Text

```
File: App.js
06:     <View style={styles.container}>
07:       <Text style={{ fontWeight: "bold" }}>
08:         I am bold
09:         <Text style={{ color: 'red' }}> and red</Text>
10:       </Text>
11:     </View>
```



```
File: App.js
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

function formatName(user) {
    return user.firstName+ ' ' + user.lastName;
}

const user = {
firstName: 'Harper',
lastName: 'Perez'
};

export default function App() {
  return (
    <View style={styles.container}>
     <Text style={{ fontWeight: "bold" }}> I am bold
      <Text style={{ color: 'red' }}> and red</Text>
     </Text>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



# Text

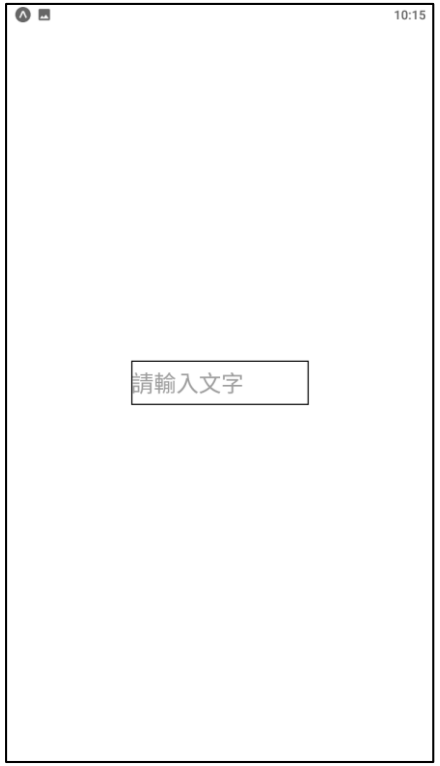
- 在 React Native 中，所有的文字都必須放在 Text 或其他組件中，不能單獨放在 View 組件內。

```
// BAD: will raise exception, can't have a text node as child of a <View>
<View>
Some text
</View>

// GOOD
<View>
<Text>
Some text
</Text>
</View>
```

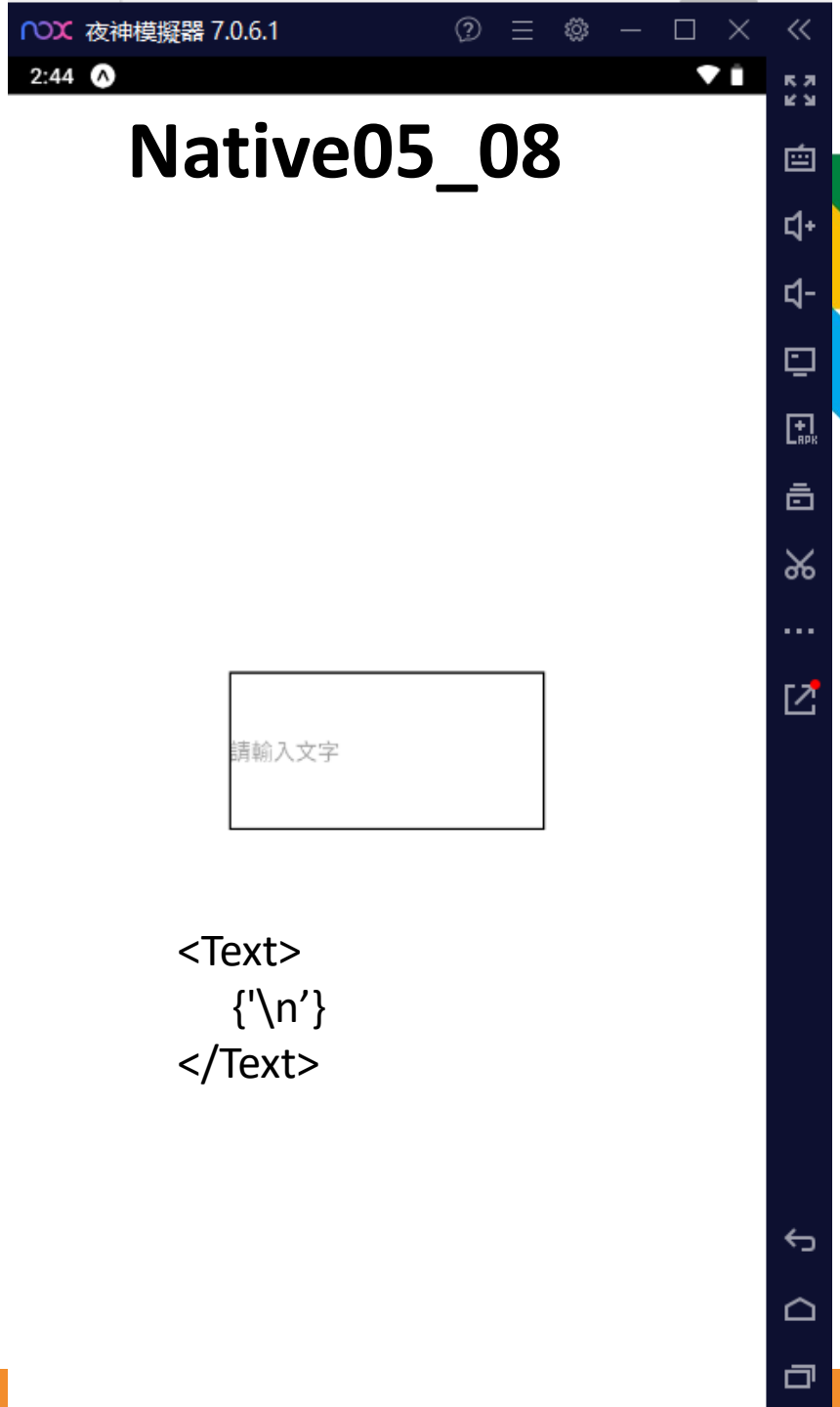
# TextInput

- TextInput 是 React Native 的文字輸入框，主要負責文字的輸入及操作。

```
File: App.js
06:     <View style={styles.container}>
07:
08:       <TextInput
09:         style={{height:40, width:200,borderWidth:1}}
10:         placeholder="請輸入文字">
11:
12:       </TextInput>
13:     </View>
```



```
File: App.js
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, TextInput, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <TextInput
        style={{height:100, width: 200, borderWidth:1}}
        placeholder="請輸入文字"
        multiline={true} >
      </TextInput>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```




```
<Text>
    {'\n'}
</Text>
```

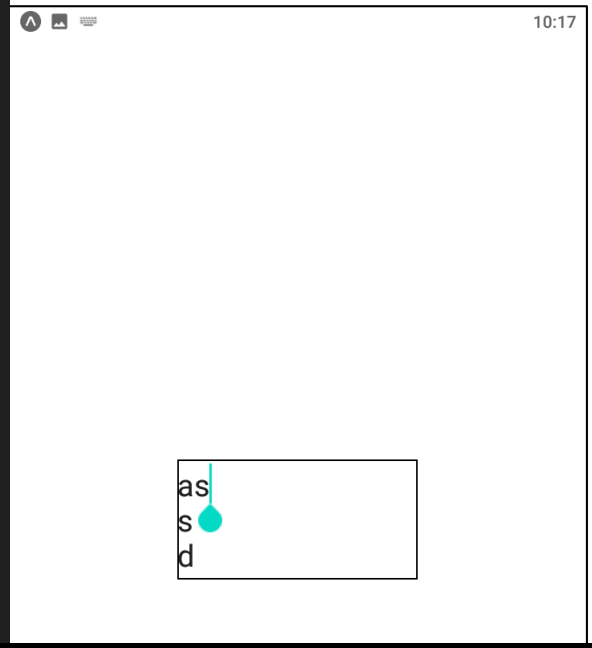
# TextInput 屬性

| 屬性名稱 | 功能 |
| --- | --- |
| autoCapitalize | 自動切換大小寫 |
| autoCorrect | 自動拼寫校正 |
| defaultValue | 初始值 |
| editable | 是否可以修改文字 |
| maxLength | 可輸入字數 |
| multiline | 是否可以輸入多行文字 |
| placeholder | 提示訊息 |
| secureTextEntry | 是否隱藏文字內容 |

https://reactnative.dev/docs/textinput

# TextInput - multiline

```
3   import { StyleSheet, TextInput, View } from 'react-native';
4
5   export default function App() {
6     return (
7       <View style={styles.container}>
8         <TextInput
9           style={{height:100, width: 200, borderWidth:1}}
10          placeholder="請輸入文字"
11          multiline={true} >
12        </TextInput>
13      </View>
14    );
15  }
16
17  const styles = StyleSheet.create({
18   container: {
19      flex: 1,
20      backgroundColor: '#fff',
21      alignItems: 'center',
22      justifyContent: 'center',
23    },
24  });
```

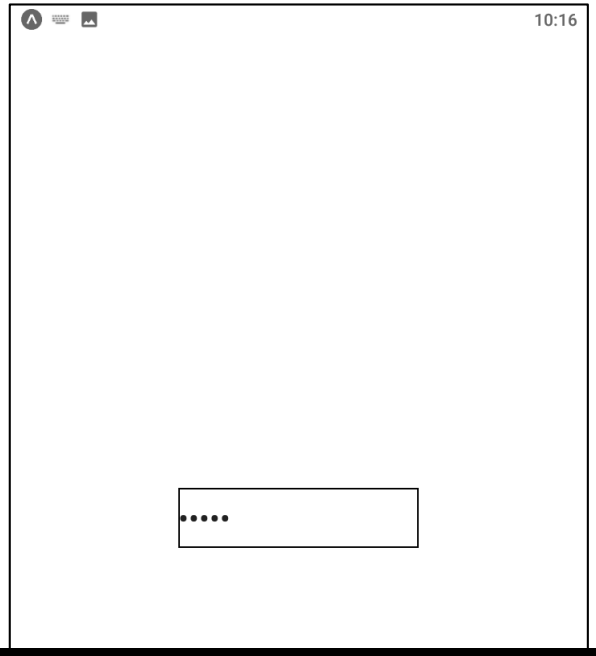



File: App.js

```
06:    <View style={styles.container}>
07:      <TextInput
08:        style={{height:100, width: 200, borderWidth:1}}
09:        placeholder="請輸入文字"
10:        multiline={true} />
11:    </View>
```

# TextInput - secureTextEntry

```
3   import { StyleSheet, TextInput, View } from 'react-native';
4
5   export default function App() {
6     return (
7       <View style={styles.container}>
8         <TextInput
9           style={{height:100, width: 200, borderWidth:1}}
10          placeholder="請輸入文字"
11          secureTextEntry={true}  >
12        </TextInput>
13      </View>
14    );
15  }
16
17  const styles = StyleSheet.create({
18   container: {
19      flex: 1,
20      backgroundColor: '#fff',
21      alignItems: 'center',
22      justifyContent: 'center',
23    },
24  });
```




File: App.js

```
06:    <View style={styles.container}>
07:      <TextInput
08:        style={{height:40, width: 200, borderWidth:1}}
09:        placeholder="請輸入文字"
10:        secureTextEntry={true} />
11:    </View>
```

# TextInput - keyboardType

File: App.js

```
06:    <View style={styles.container}>
07:      <TextInput
08:        style={{height:100, width: 200, borderWidth:1, fontSize: 25}}
09:        placeholder="請輸入文字"
10:        keyboardType="numeric"/>
11:    </View>
```

- default
- number-pad
- decimal-pad
- numeric
- email-address
- phone-pad

numeric


email-address

# TextInput Callback function

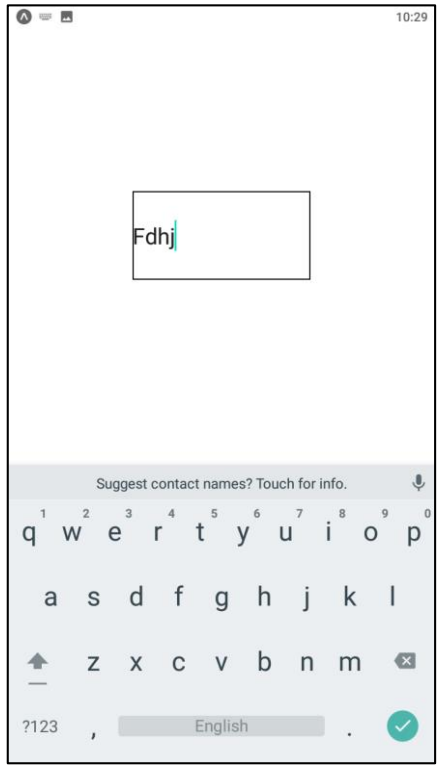
- TextInput 具有 callback function，以下是常用的 callback functions:

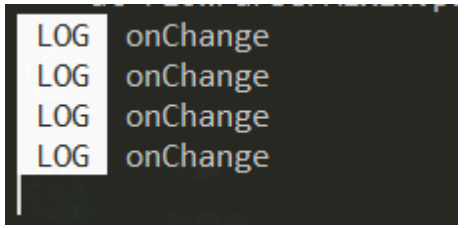
| 屬性 | 功能 |
| --- | --- |
| onChange | 當TextInput內容變化時，會呼叫此Callback function |
| onChangeText | 當TextInput內容變化時，會呼叫此Callback function |
| onFocus | 當TextInput獲得焦點 (focus) 時，會呼叫此Callback function |
| onBlur | 當TextInput失去焦點(focus) 時，呼叫此Callback function |

# TextInput Callback function - onChange

```
File: App.js
06:    <View style={styles.container}>
07:      <TextInput
08:        style={{height:100, width: 200, borderWidth:1, fontSize: 25}}
09:        placeholder="請輸入文字"
10:        onChange={()=>{console.log('onChange')}}
11:        />
12:    </View>
```

可以取得額外的資料




**onChange**

Callback that is called when the text input's text changes.

| TYPE |
| --- |
| `({nativeEvent: {eventCount, target, text}}) => void` |

練習: 嘗試加入function參數，看看結果。

# TextInput Callback function - onChangeText

File: App.js

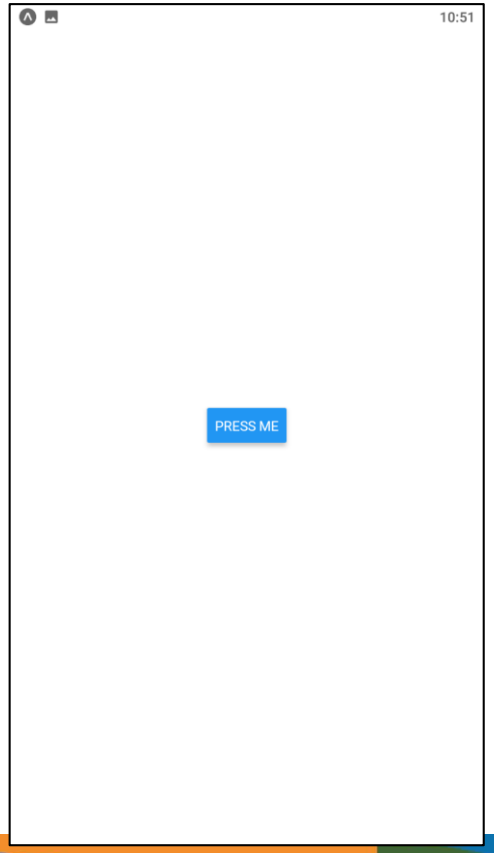
```
06:    <View style={styles.container}>
07:      <TextInput
08:        style={{height:100, width: 200, borderWidth:1, fontSize: 25}}
09:        placeholder="請輸入文字"
10:        // onChange={()=>{console.log('onChange')}
11:        onChangeText={(text)=>(console.log(text))}
12:        />
13:    </View>
```

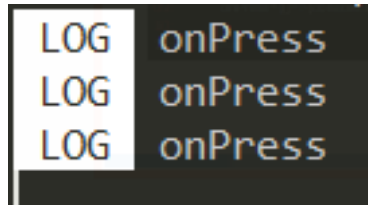
text 可以取得 TextInput 內的文字訊息

# Button

- 在 React Native 中，為了讓使用者可以跟頁面互動，因此提供了許多可以觸摸的 Component， Button是其中一項。

```
File: App.js
06:    <View style={styles.container}>
07:      <Button
08:        title= "Press Me"
09:        onPress={()=>{console.log('onPress')}}
10:      />
11:    </View>
```

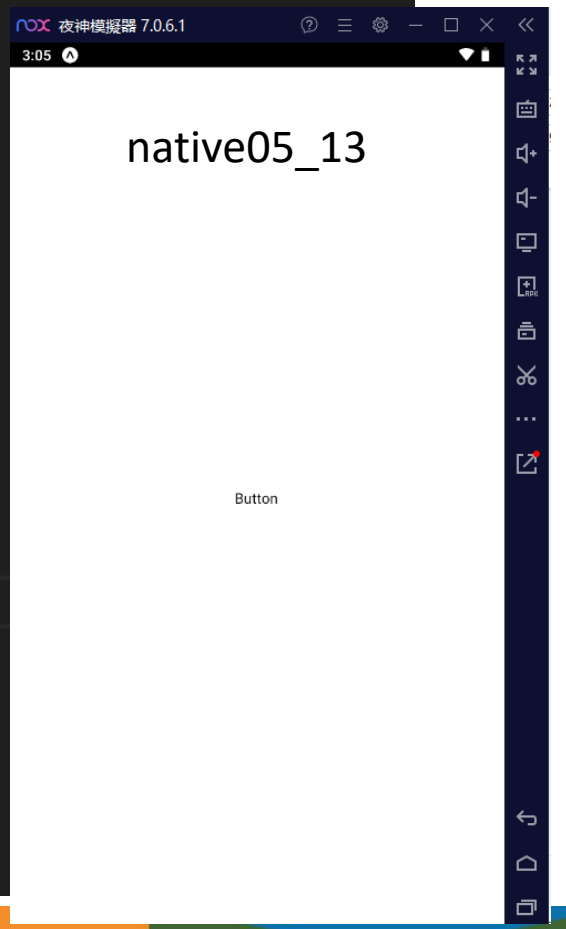


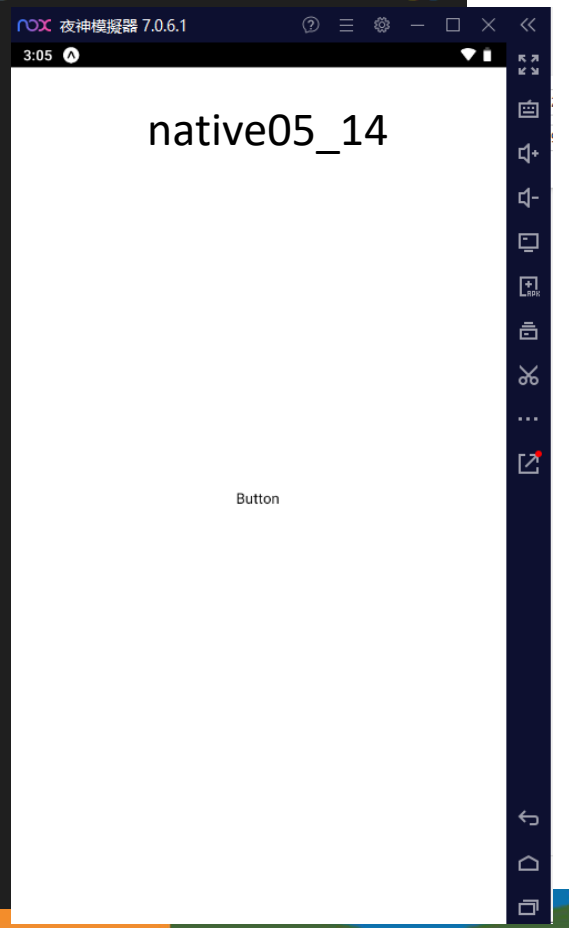
# Button 屬性

| 屬性 | 描述 |
| --- | --- |
| title | 必要，Button 上顯示的文字 |
| onPress | 必要，處理按下 Button 後續的行為 |
| disabled | True; Button 不可按<br>False; Button 可按 |
| color | 背景顏色 |

```
File: App.js

import { StyleSheet, Text, View, Button, TouchableHighlight } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <TouchableHighlight
        style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
        underlayColor="gray"
        activeOpacity={0.7}
        onPress={() => { console.log("TouchableHighlight") }}>
      <Text>Button</Text>
      </TouchableHighlight>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

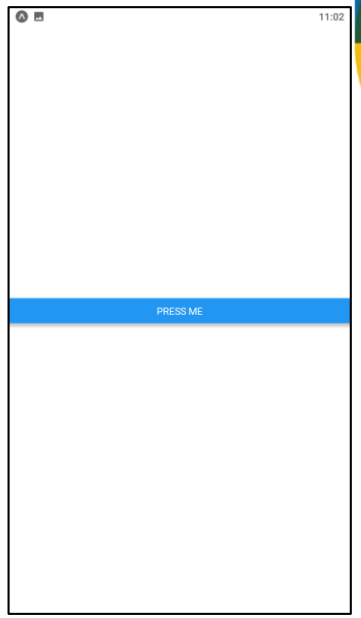


```
File: App.js

import { StyleSheet, Text, View, Button, TouchableHighlight } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <TouchableHighlight
        style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
        underlayColor="gray"
        activeOpacity={0.3}
        onPress={() => { console.log("TouchableOpacity") }}>
        <Text style={{fontSize: 30}}>Button</Text>
      </TouchableHighlight>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

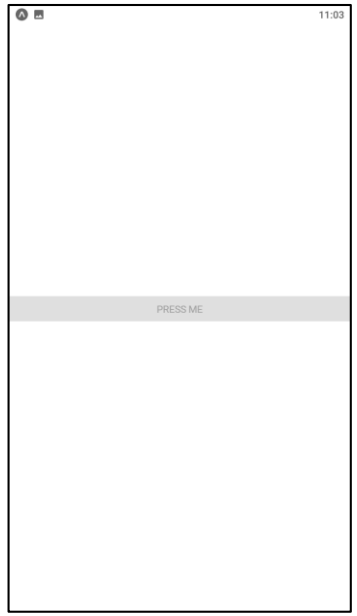


調整 Button size

```
File: App.js
06:    <View style={styles.container}>
07:      <View style={{width:500}}>
08:       <Button
09:         title= "Press Me"
10:         onPress={()=>{console.log('onPress')}}
11:         disabled={false}
12:       />
13:      </View>
14:    </View>
```




```
File: App.js
06:    <View style={styles.container}>
07:      <View style={{width:500}}>
08:      <Button
09:         title= "Press Me"
10:         onPress={()=>{console.log('onPress')}}
11:         disabled={true}
12:       />
13:      </View>
14:    </View>
```



# Touchables

- 除了 Button以外，還可以使用 Touchables 來實作按鈕的效果。
- Touchables 包含四種 components:
  - TouchableHighlight
  - TouchableOpacity
  - TouchableWithoutFeedback
  - TouchableNativeFeedback (Android 限定)

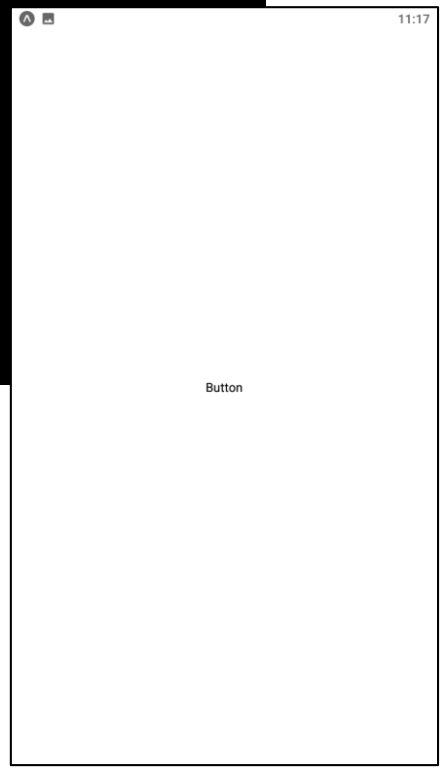
# TouchableHighlight

File: App.js

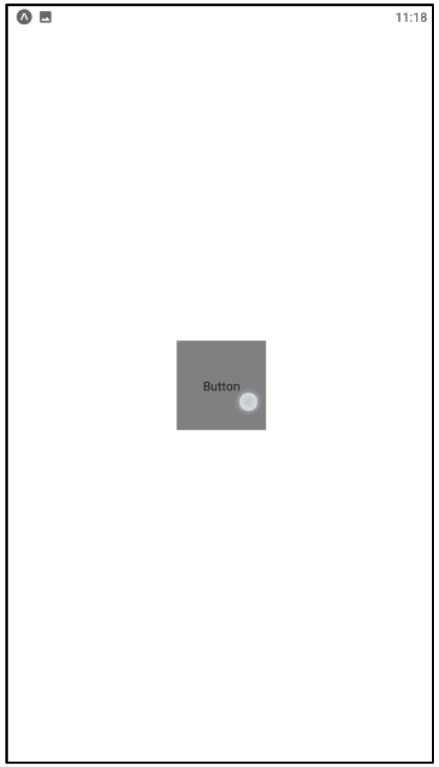
```
06:     <View style={styles.container}>
07:       <TouchableHighlight
08:         style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
09:         underlayColor="gray"
10:         activeOpacity={0.7}
11:         onPress={() => { console.log("TouchableHighlight") }}>
12:         <Text>Button</Text>
13:       </TouchableHighlight>
14:     </View>
```

- TouchableHighlight 按下去後，背景顏色會有變化

normal

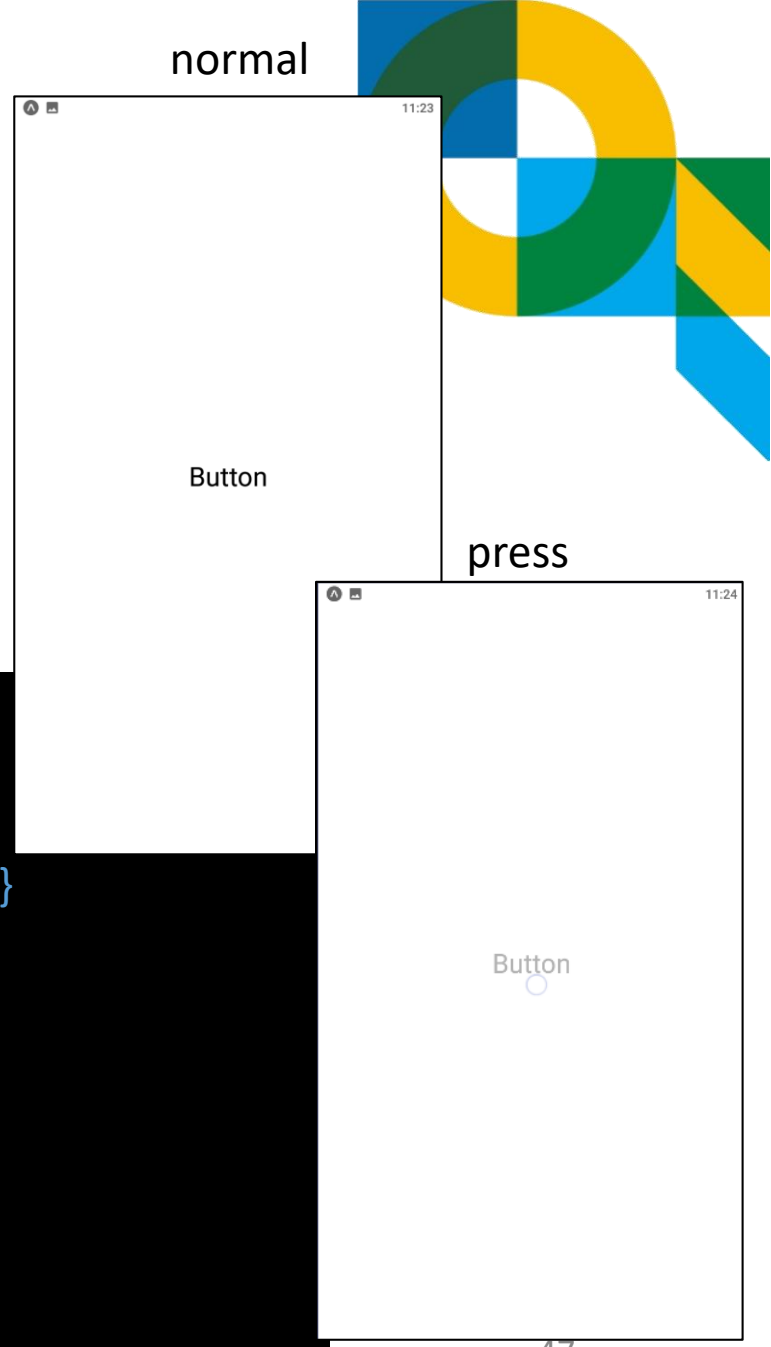



press



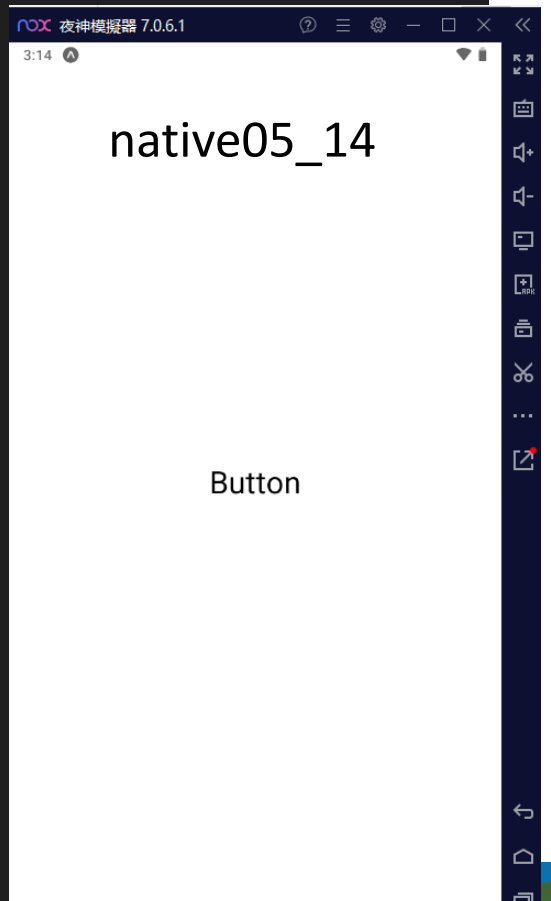
# TouchableOpacity

- 底色不會變化，文字會變成透明的。



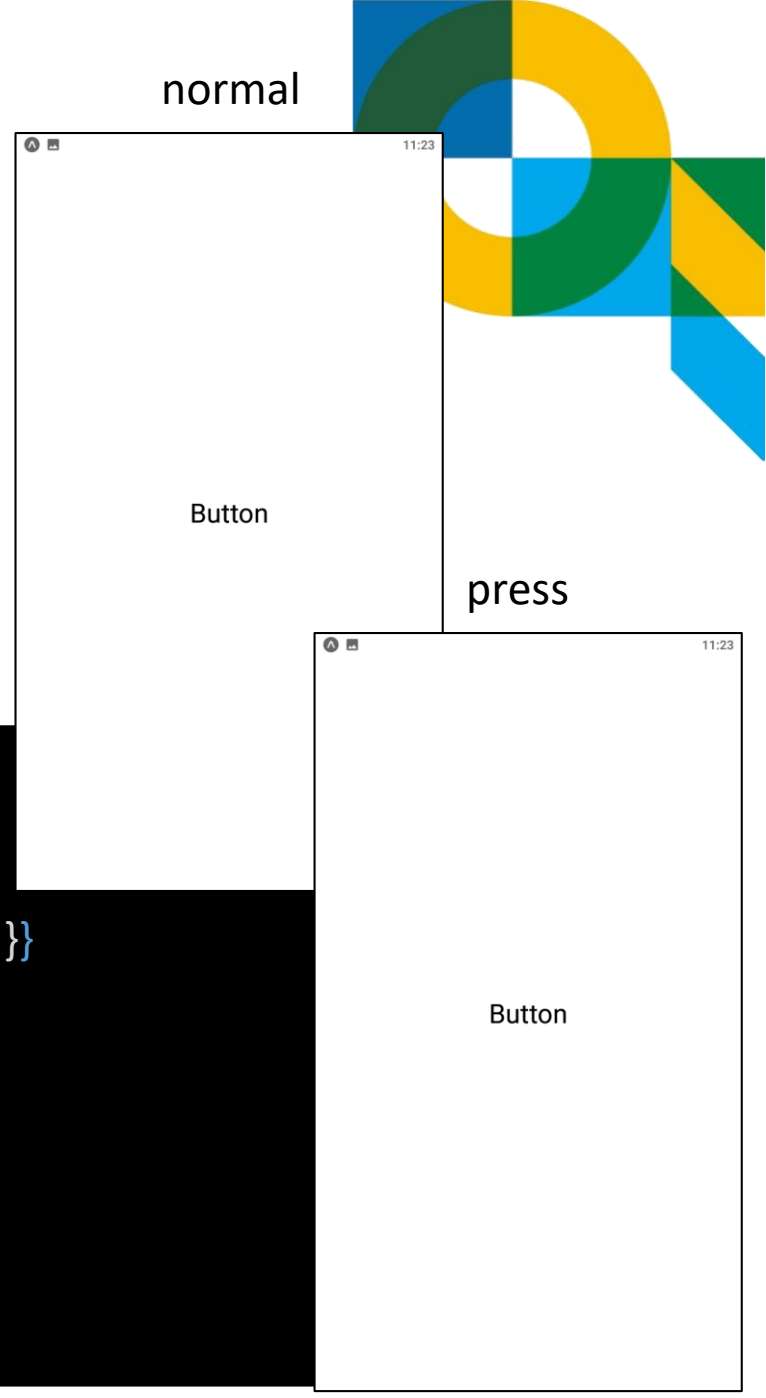

```
File: App.js
06:     <View style={styles.container}>
07:       <TouchableOpacity
08:         style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
09:         activeOpacity={0.3}
10:         onPress={() => { console.log("TouchableOpacity") }}>
11:         <Text style={{fontSize: 30}}
12:         >Button</Text>
13:       </TouchableOpacity>
14:     </View>
```

```
File: App.js

import { StyleSheet, Text, View, Button, TouchableHighlight } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <TouchableHighlight
        style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
        underlayColor="gray"
        activeOpacity={0.3}
        onPress={() => { console.log("TouchableOpacity") }}>
      <Text style={{fontSize: 30}}>Button</Text>
      </TouchableHighlight>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



# TouchableWithoutFeedback

- 按下去，不會有任何變化，
  適合不需要有反應的情況。



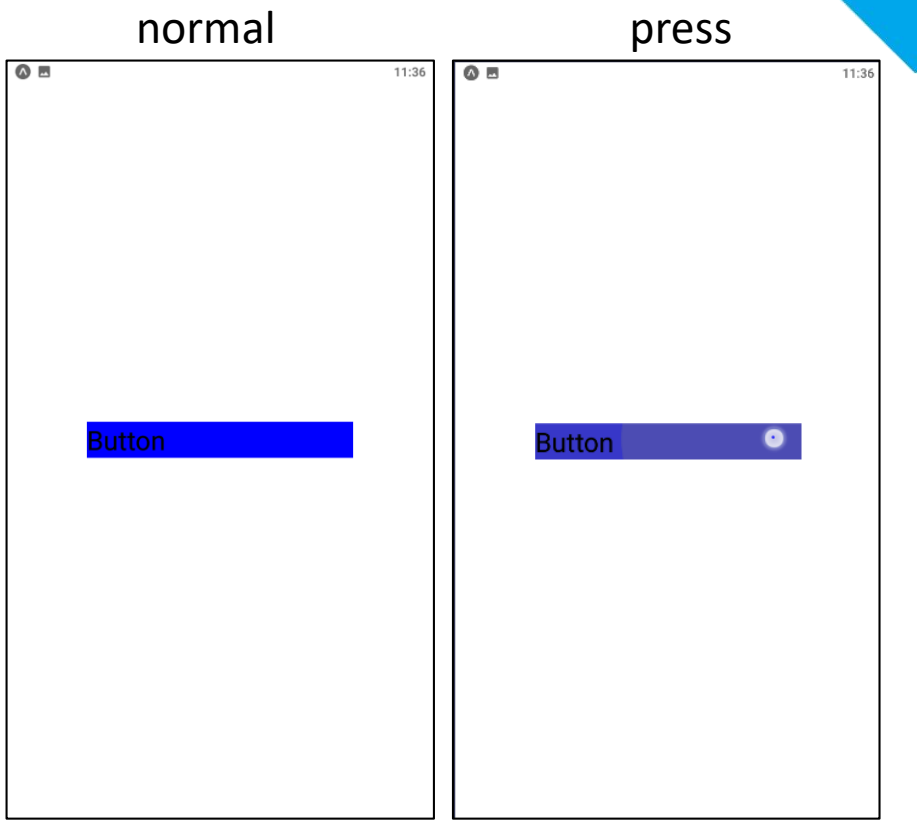

File: App.js

```
06:     <View style={styles.container}>
07:       <TouchableWithoutFeedback
08:         style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
09:         activeOpacity={0.3}
10:         onPress={() => { console.log("TouchableWithoutFeedback") }}>
11:         <Text style={{fontSize: 30}}>
12:           Button
13:         </Text>
14:       </TouchableWithoutFeedback>
15:     </View>
```

# TouchableNativeFeedback

File: App.js

```
06:     <View style={styles.container}>
07:       <TouchableNativeFeedback
08:         style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
09:         background={TouchableNativeFeedback.Ripple('gray', false)}
10:         onPress={() => { console.log("TouchableNativeFeedback") }}>
11:         <View style={{ backgroundColor: 'blue', width: 300 }}>
12:           <Text style={{ fontSize: 30 }}>
13:             Button
14:           </Text>
15:         </View>
16:       </TouchableNativeFeedback>
17:     </View>
```




- Android 限定
- 按下後，會有水波的動畫

# Touchables 通用屬性

| 屬性 | 描述 | 範例 |
| --- | --- | --- |
| hipSlop | 在距離 Touchables component 外多遠可以觸發按鈕 | hipSlop={{top:10, left: 10, bottom:10, right:10}} |
| disabled | 設定是否可以點擊按鈕 | |

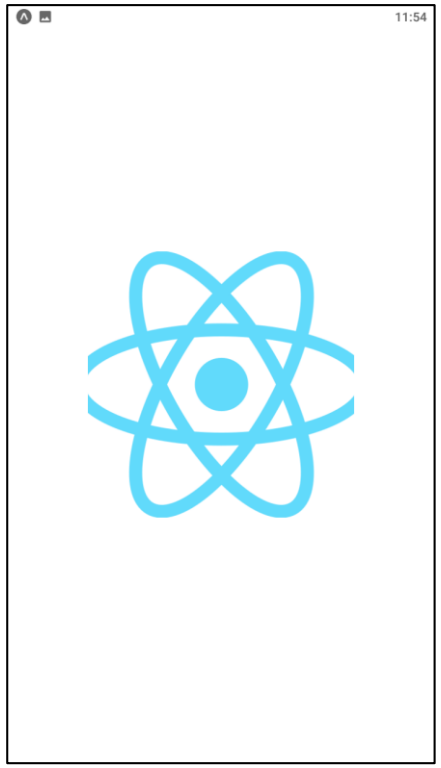
| Callback 函數 | 描述 |
| --- | --- |
| onLongPress | 長按按鈕時觸發 |
| onPress | 按鈕被按下時觸發 |
| onPressIn | 按鈕按下前觸發 |
| onPressOut | 按鈕放開後觸發 |

# Image

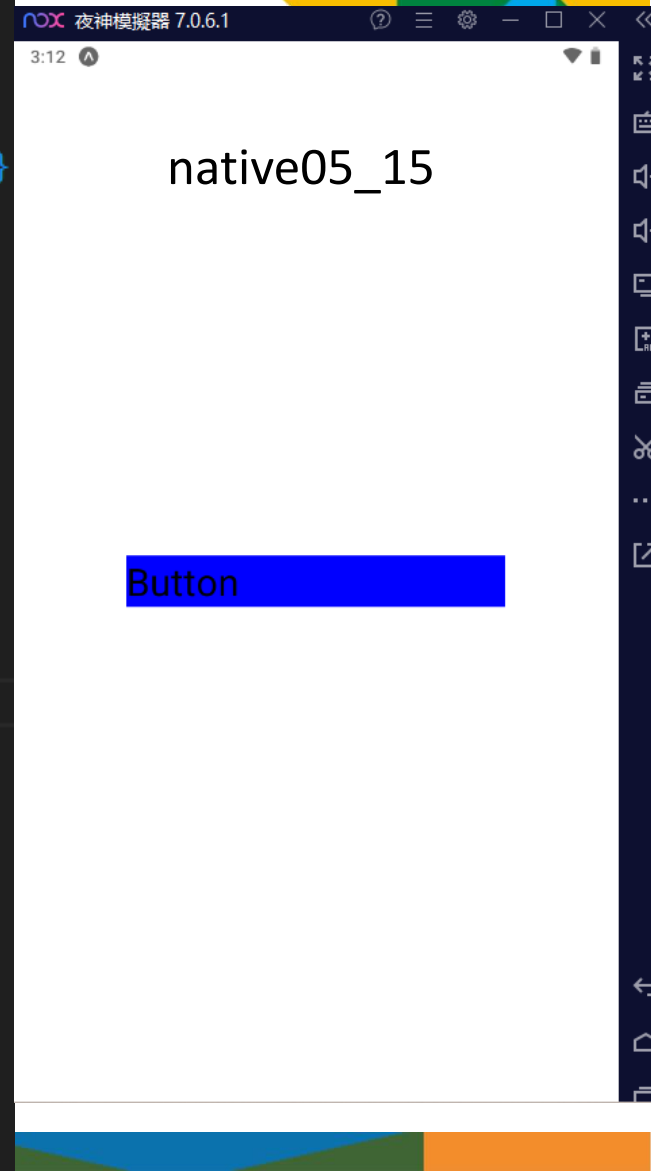
- 顯示圖片用的 component。

File: App.js

```
06:      <View style={styles.container}>
07:        <Image
08:          style={{width:300, height:300}}
09:          source={require('./React-icon.png')}
10:          />
11:      </View>
```

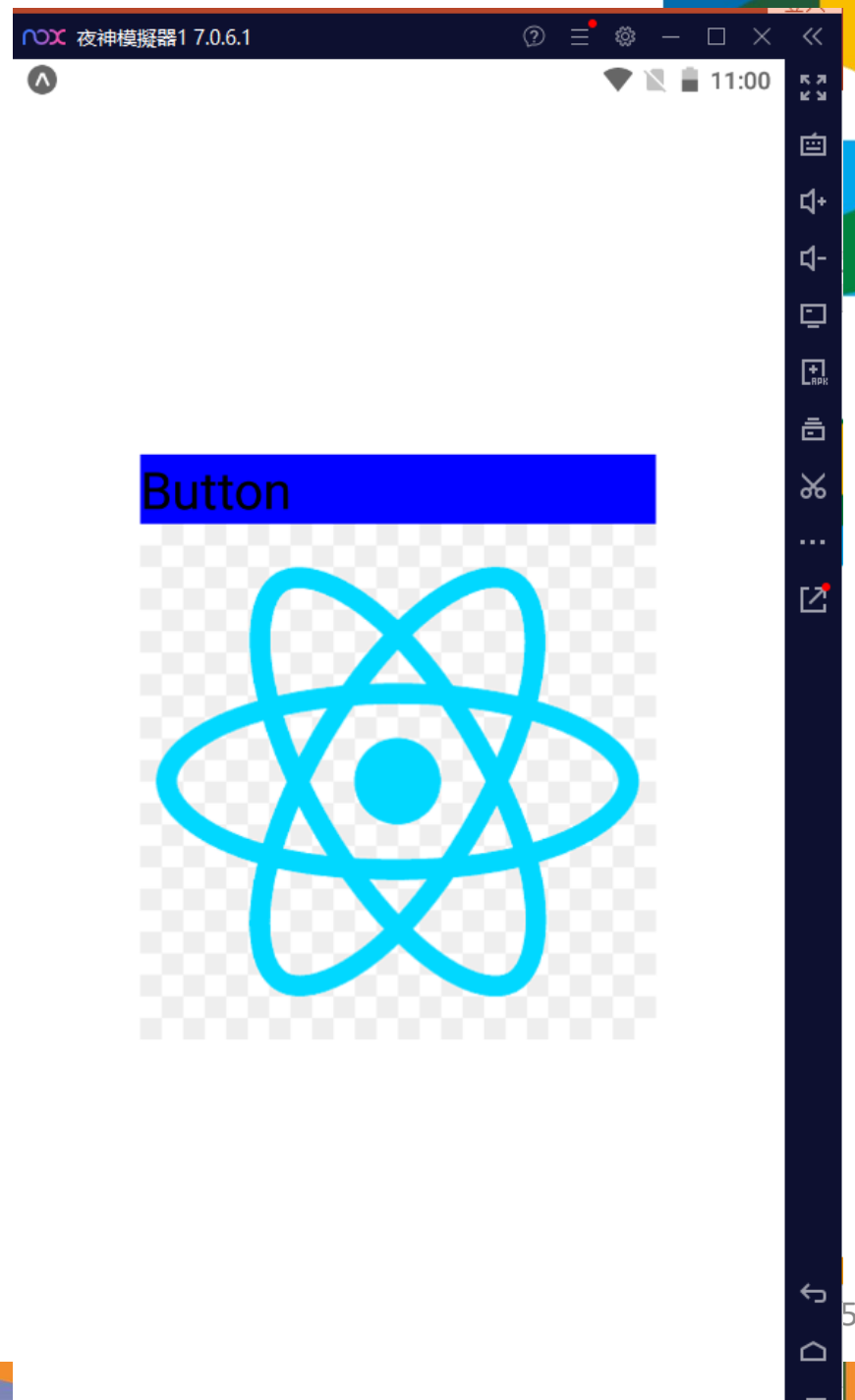


```
File: App.js

import { StyleSheet, Text, View, TouchableNativeFeedback } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <TouchableNativeFeedback
        style={{ height: 100, width: 100, alignItems: 'center', justifyContent: 'center' }}
        underlayColor="gray"
        activeOpacity={0.3}
        background={TouchableNativeFeedback.Ripple('gray', false)}
        onPress={() => { console.log("TouchableNativeFeedback") }}>
        <View style={{ backgroundColor: 'blue', width: 300 }}>
          <Text style={{fontSize: 30}}>Button</Text>
        </View>
      </TouchableNativeFeedback>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



JS App.js > ...

```
File: App.js
import { StyleSheet, Text, View, Image} from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <View style={{ backgroundColor: 'blue', width: 300 }}>
        <Text style={{fontSize: 30}}>Button</Text>
        <Image
          style={{width:300, height:300}}
          source={require('./React-icon.png')}
        />
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



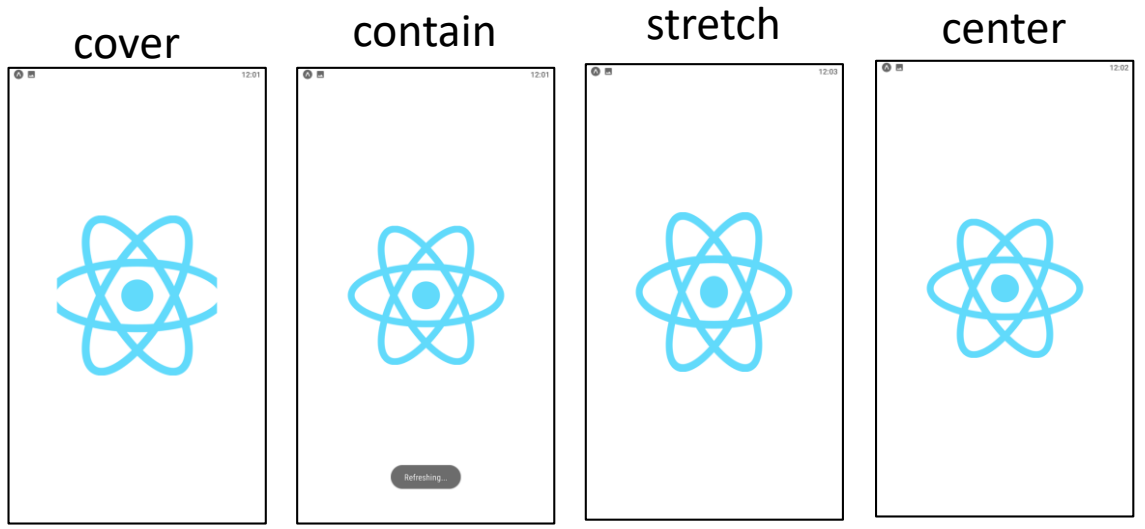
# Image

練習一下: 不同的resizeMode的效果。

| 屬性 | 描述 |
| --- | --- |
| source | 圖片位置 |
| resizeMode | 如果圖片跟顯示的區塊大小不同時使用。<br>● cover: 保持圖片比例，放大到長和寬都填滿父組件。<br>● contain: 保持圖片比例，放大到長或寬等於父組件<br>● stretch:不保持比例，填滿父組件。<br>● repeat: 維持圖片比例，重複放置，直到填滿父組件，iOS 限定。<br>● center:維持圖片比例，並且置中顯示。 |

File: App.js

```
06:     <View style={styles.container}>
07:       <Image
08:         style={{width:300, height:300}}
09:         source={require('./React-icon.png')}
10:         resizeMode="cover"
11:         />
12:     </View>
```



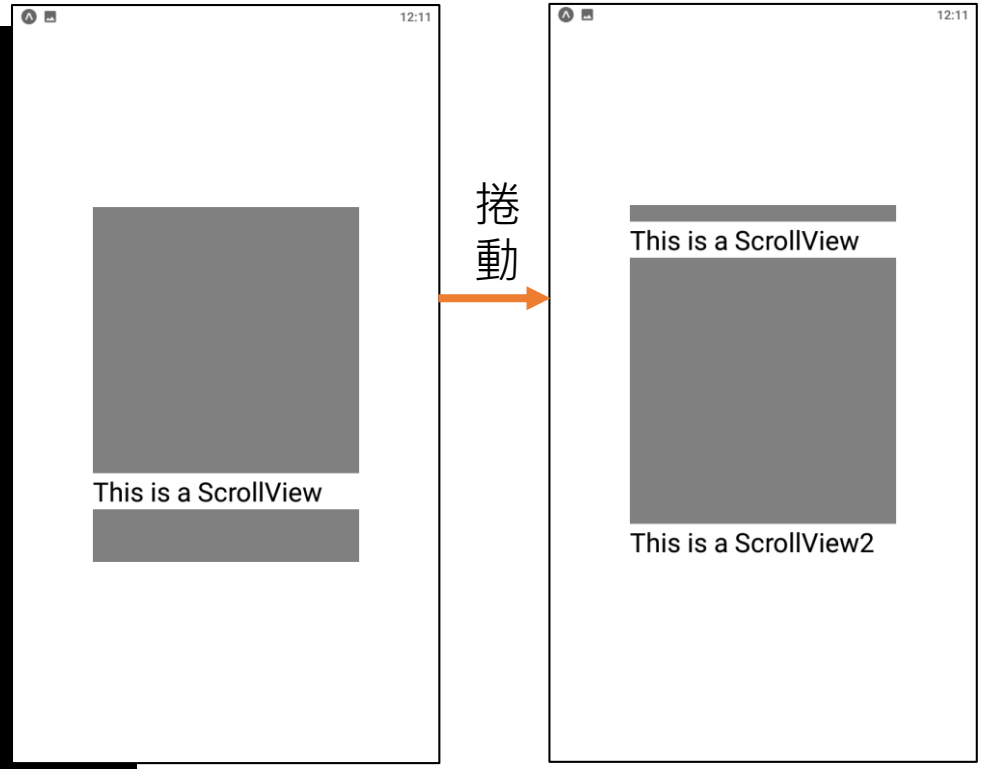
# ScrollView

練習一下: 試著加入更多的元件，了解 ScrollView 的效果

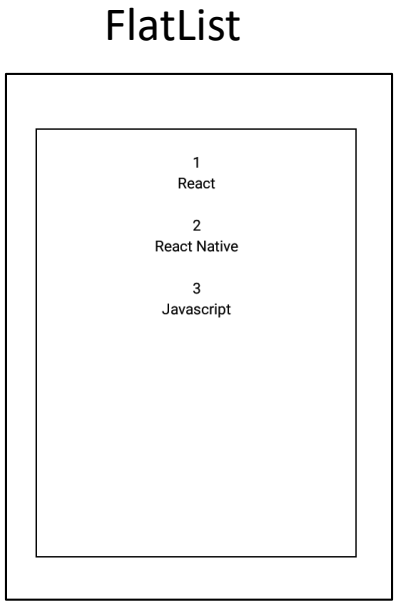
- 當頁面資料超過螢幕大小時，可以讓使用者捲動畫面。

File: App.js

```
06:     <View style={styles.container}>
07:       <View style={{ width: 300, height: 400, }}>
08:         <ScrollView>
09:           <View style={{ backgroundColor: 'gray', height: 300 }} />
10:           <Text style={{ fontSize: 30 }}>
11:             This is a ScrollView
12:           </Text>
13:           <View style={{ backgroundColor: 'gray', height: 300 }} />
14:           <Text style={{ fontSize: 30 }}>
15:             This is a ScrollView2
16:           </Text>
17:         </ScrollView>
18:       </View>
19:     </View>
```



# FlatList



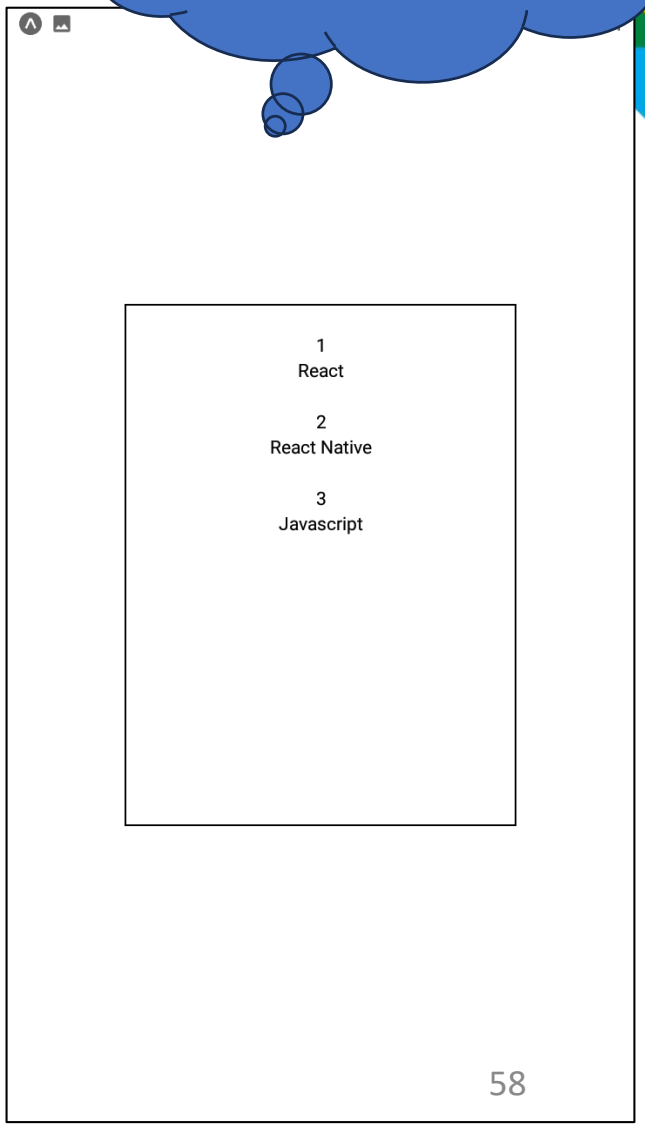

- FlatList 適合簡單的資料列表
  - https://reactnative.dev/docs/flatlist
- 主要包含三個元件:
  - data: 設定要傳入畫面渲染的資料，以 Json 陣列格式表示。
  - renderItem: 用來設定資料的渲染樣式；將 data 陣列中的每一筆資料，轉換成 {item} 物件，然後設定渲染的樣式。
  - keyExtractor: 用來設定每一筆 data 的特徵值，當單筆 data 發生改變時，可以透過檢查特徵值來避免重新渲染整個 data 陣列，提高效能。

# FlatList

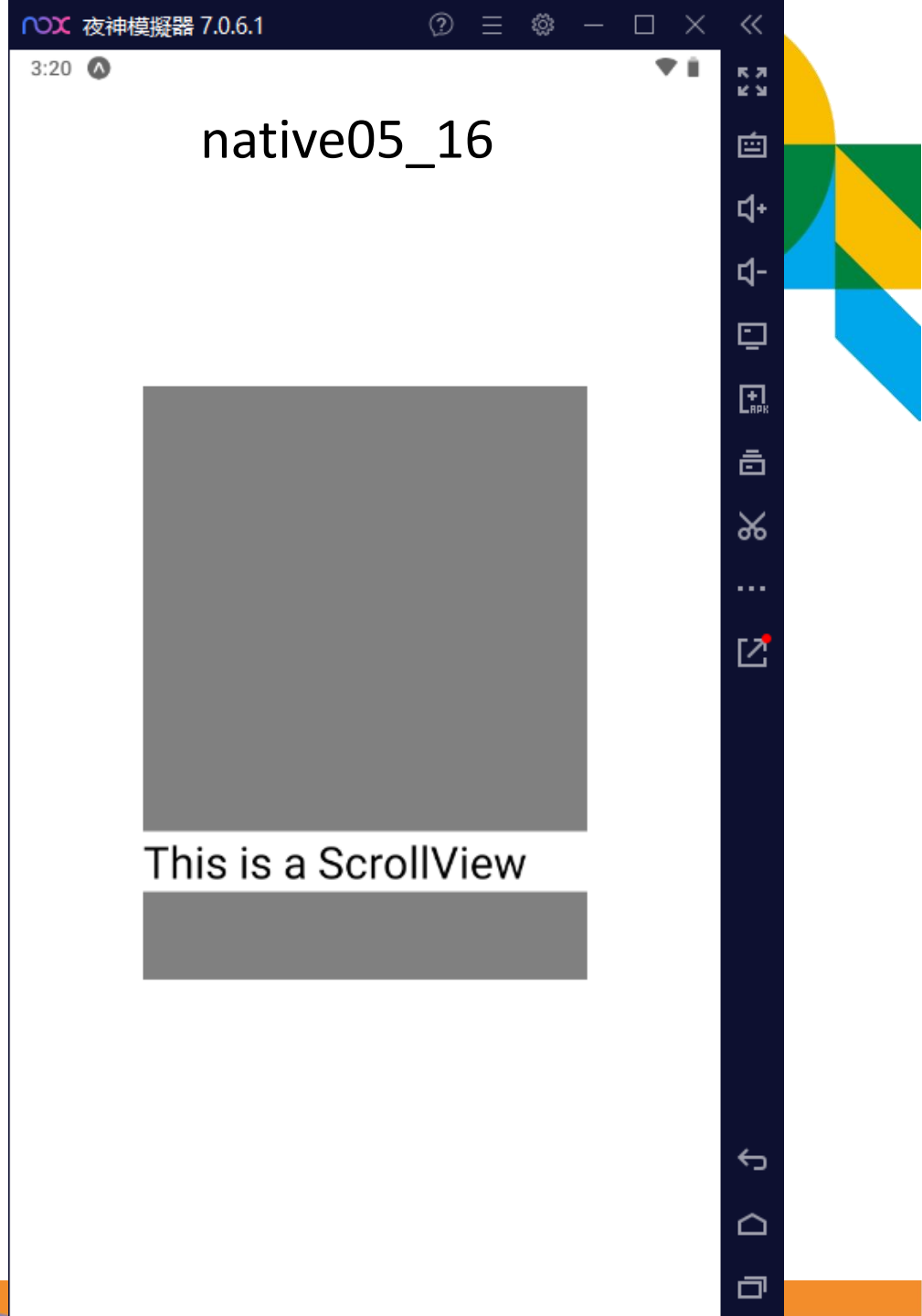
File: App.js

```
04: export default function App() {
05:
06:   var data01=[
07:       {key: '1', data:'React'},
08:       {key: '2', data:'React Native'},
09:       {key: '3', data:'Javascript'},];
10:  return (
11:     <View style={styles.container}>
12:        <View style={{ width: 300, height: 400, }}>
13:           <FlatList
14:             data={data01}
15:             renderItem={({item})=>(
16:                <View style={{marginTop:20, alignItems:'center'}}>
17:                   <Text>{item.key}</Text>
18:                   <Text>{item.data}</Text>
19:                </View>
20:             )}
21:             keyExtractor={item => item.key}
22:             style={{borderWidth: 1}}
23:             contentContainerStyle={{alignItems: 'center'}}/>
24:        </View>
25:     </View>
26:  );
27: }
```

練習一下: 試著加入更多的資料。



```
File: App.js
import { StyleSheet, Text, View, ScrollView} from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <View style={{ width: 300, height: 400, }}>
       <ScrollView>
          <View style={{ backgroundColor: 'gray', height: 300 }} />
          <Text style={{fontSize: 30}}>
            This is a ScrollView
          </Text>
          <View style={{ backgroundColor: 'gray', height: 300 }} />
          <Text style={{fontSize: 30}}>
            This is a ScrollView2
          </Text>
        </ScrollView>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



# FlatList

- keyExtractor={item => item.key}
  - 這行程式碼在你的 React Native 應用程式中，主要是用來指定 FlatList 中每個項目的唯一鍵（key）。
  - 解釋：FlatList：這是一個 React Native 中用來有效地渲染大型列表的組件。它具有高效能，因為它只會渲染目前在螢幕上可見的項目，而不會渲染螢幕外的項目。
  - keyExtractor：這是一個你提供給 FlatList 的屬性，用來告訴它如何為列表中的每個項目提取唯一的鍵。鍵的存在幫助 React 識別哪些項目已更改、被添加或被移除，這對於有效地更新 UI 十分重要。

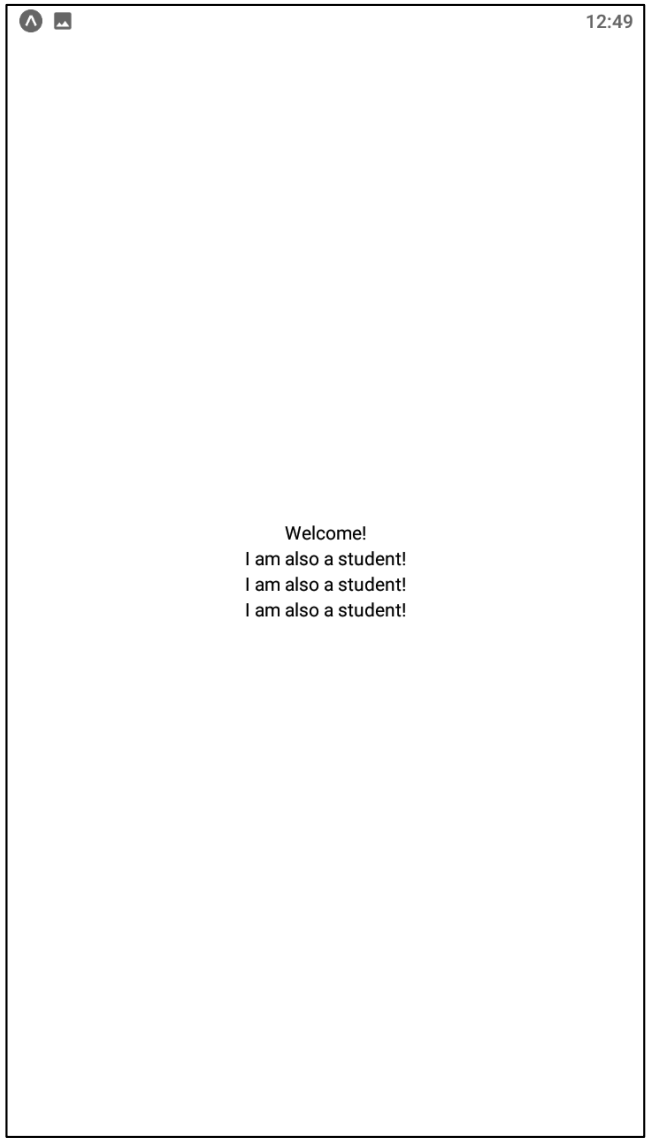
# FlatList

- keyExtractor={item => item.key}：這行是一個函數，它接收數據陣列（data01）中的每個 item，並返回其 key 屬性的值。在這種情況下，key 屬性被假定為每個項目的唯一識別符。
- keyExtractor 為什麼重要？
  - React Native 使用鍵來跟踪列表中的每個項目，這有助於優化渲染性能。如果兩個項目有相同的鍵，React Native 可能會混淆它們，導致不預期的行為。因此，確保鍵對於列表中的每個項目都是唯一的非常重要。
- keyExtractor={item => item.key}
  - 這個 keyExtractor 屬性告訴 FlatList 使用每個 item 中的 key 屬性作為唯一的識別符。

# 自定義組件

- 開發者可以自已定義 components，達到重複使用 components的目的，減輕開發負擔。
- 例如: 表單頁面會重複出現，這時就可以利用自定義組件來避免重複程式碼。

# 自定義組件




File: App.js

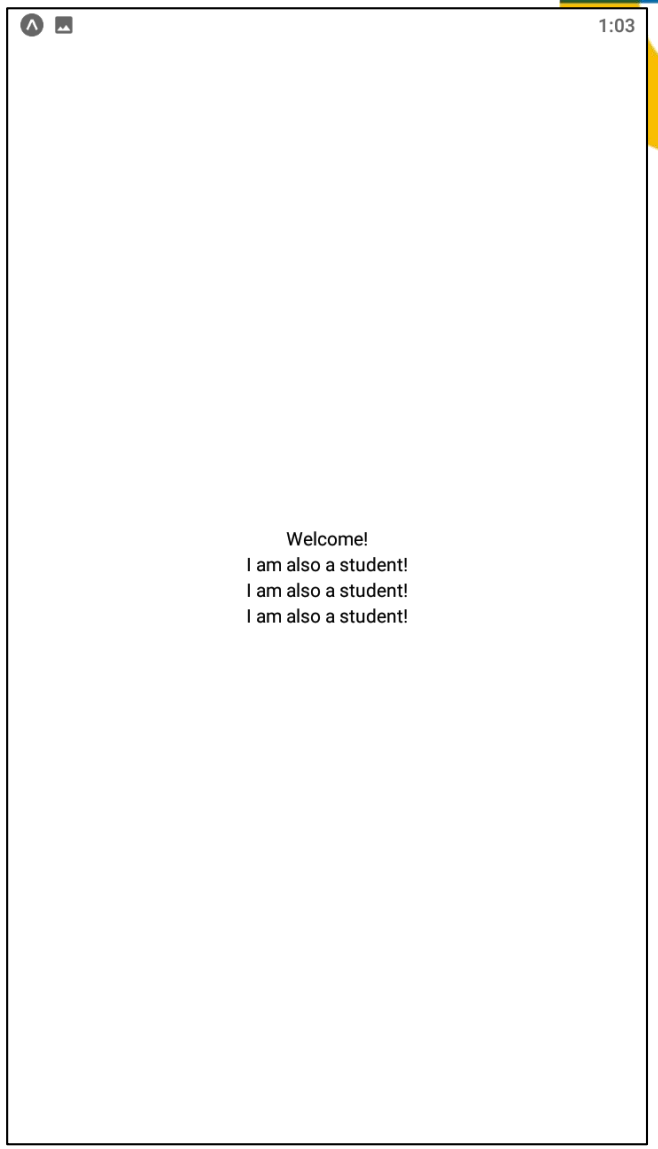
```
04: const Student = () => {
05:     return (
06:       <View>
07:         <Text>I am also a student!</Text>
08:       </View>
09:     );
10:   }
11:
12: export default function App() {
13:
14:   return (
15:       <View style={styles.container}>
16:         <Text>Welcome!</Text>
17:         <Student />
18:         <Student />
19:         <Student />
20:       </View>
21:   );
22: }
23:
```

File: App.js

```
03: import Student from './Student';
04:
05: export default function App() {
06:   return (
07:     <View style={styles.container}>
08:       <Text>Welcome!</Text>
09:       <Student />
10:       <Student />
11:       <Student />
12:     </View>
13:   );
14: }
```

File: Student.js

```
01: import React, { Components } from 'react';
02: import { Text, View, Button } from 'react-native';
03:
04: export default class Student extends React.Component {
05:    render() {
06:       return (
07:          <View>
08:             <Text>I am also a student!</Text>
09:          </View>
10:       );
11:    }
12: };
```



# 作業 1

- 請製作一個頁面如右

# Q&A