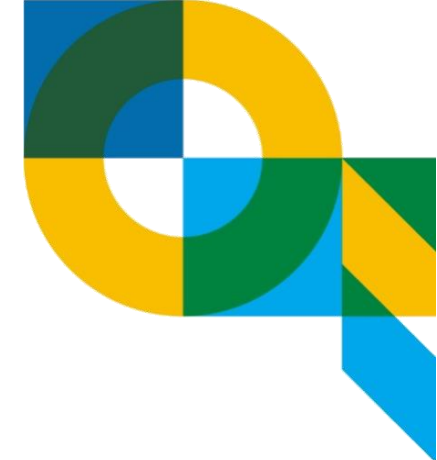# 8. 互動設計

廖奕雯

yiwen923@nkust.edu.tw

Outline
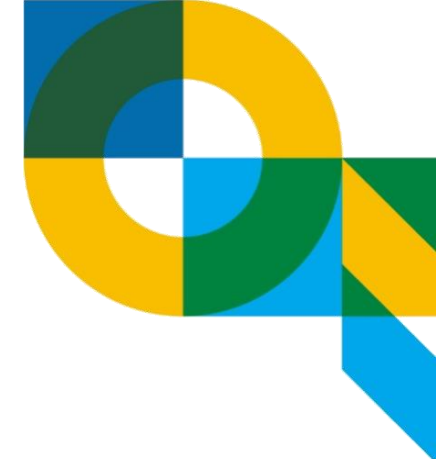
- Alert
- Platform
- Appearance
- Gesture Responder System

Alert

Alert

- 啟動一個提示對話方塊，包含對應的標題和資訊。
- 可以指定一系列的按鈕，點擊對應的按鈕會調用對應的 onPress callback function 並且關閉提示框。
- 預設情況下，對話方塊會僅有一個 "確定" 按鈕。
- 在 Android 上最多能指定三個按鈕，這三個按鈕分別具有 "neutral"、 "negative" 和 "positive" 的概念：
  - 如果你只指定一個按鈕，則它具有 "positive" 的屬性(比如"確定")
  - 兩個按鈕，則分別是 "negative" 和 "positive"(比如"取消"和"確定")
  - 三個按鈕則意味著 "neutral"、 "negative" 和 "positive"(比如"稍候再說"，"取消"，"確定")。
- 本介面可以在 iOS 和 Android 上顯示一個靜態的提示框。只有 iOS 系統支援在提示框中加入TextInput。

alert()

```
static alert(title, message?, buttons?, options?)
```

Parameters:

NAME	TYPE	DESCRIPTION
title **Required**	string	The dialog's title. Passing `null` or empty string will hide the title.
message	string	An optional message that appears below the dialog's title.
buttons	Buttons	An optional array containing buttons configuration.
options • Android	Options	An optional Alert configuration for the Android.

Buttons

Array of objects containing Alert buttons configuration.

TYPE
array of objects

Objects properties:

NAME	TYPE	DESCRIPTION
text	string	Button label.
onPress	function	Callback function when button is pressed.
style • iOS	AlertButtonStyle	Button style, on Android this property will be ignored.

Options   • Android

TYPE
object

Properties:

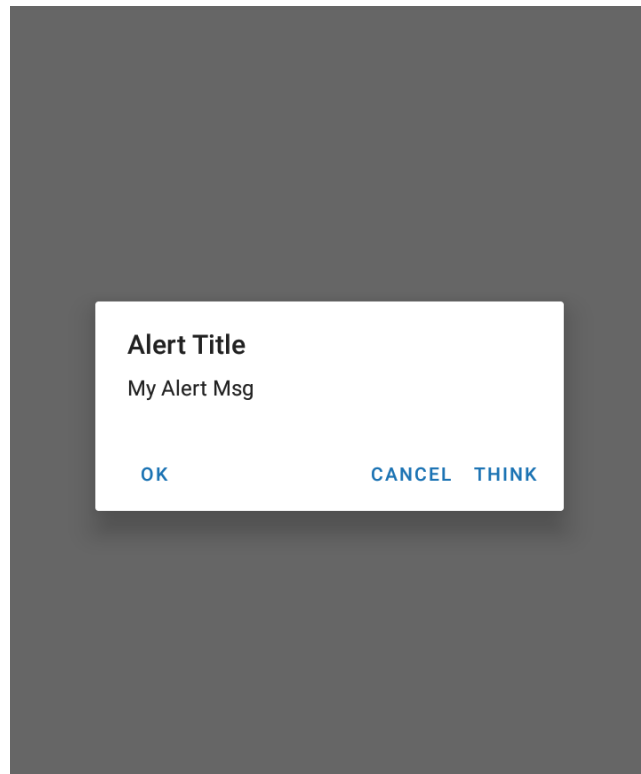| NAME	TYPE	DESCRIPTION
cancelable	boolean	Defines if alert can be dismissed by tapping outside of the alert box.
onDismiss	function	Callback function fired when alert has been dismissed.

```
File: App-Alert.js
04: const showAlert = () =>
05:   Alert.alert(
06:     "Alert Title",
07:     "My Alert Msg",
08:     [
09:       {
10:         text: "Ok",
11:         onPress: () => Alert.alert("Ok Pressed"),
12:         style: "cancel",
13:       },
14:       {
15:         text: "Cancel",
16:         onPress: () => Alert.alert("Cancel Pressed"),
17:         style: "cancel",
18:       },
19:     ],
20:     {
21:       cancelable: true,
22:       onDismiss: () =>
23:         Alert.alert(
24:           "This alert was dismissed by tapping outside of the alert dialog."
25:         ),
26:     }
27:   );
28:
29: export default function App() {
30:   return (
31:     <View style={styles.container}>
32:       <Button title="Show alert" onPress = {showAlert} color='red'/>
33:     </View>
34:   );
```
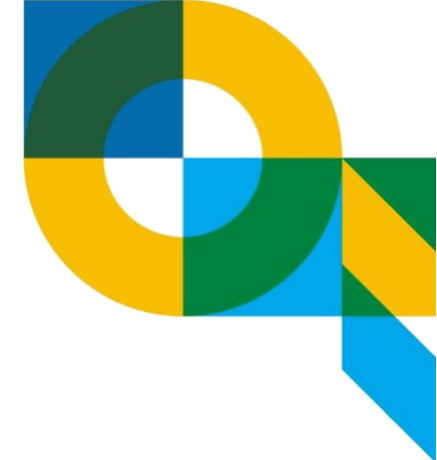
SHOW ALERT

練習一下

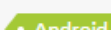- 建立一個 alert 提示框，包含三個按鈕

Platform

獲取環境變數

Platform

- 獲取系統環境變數
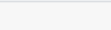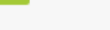
Properties

constants

```
Platform.constants;
```

Returns an object which contains all available common and specific constants related to the platform.

Properties:

| NAME	TYPE	OPTIONAL	DESCRIPTION
isTesting	boolean	No	
reactNativeVersion	object	No	Information about React Native version. Keys are `major`, `minor`, `patch` with optional `prerelease` and values are `number`s.
Version `• Android`	number	No	OS version constant specific to Android.
Release `• Android`	string	No	
Serial `• Android`	string	No	Hardware serial number of an Android device.
Fingerprint `• Android`	string	No	A string that uniquely identifies the build.
Model `• Android`	string	No	The end-user-visible name for the Android device.
Brand `• Android`	string	No	The consumer-visible brand with which the product/hardware will be associated.
Manufacturer `• Android`	string	No	The manufacturer of the Android device.
ServerHost `• Android`	string	Yes	
uiMode `• Android`	string	No	Possible values are: `'car'`, `'desk'`, `'normal'`, `'tv'`, `'watch'` and `'unknown'`. Read more about Android ModeType.
forceTouchAvailable `• iOS`	boolean	No	Indicate the availability of 3D Touch on a device.
interfaceIdiom `• iOS`	string	No	The interface type for the device. Read more about UIUserInterfaceIdiom.
osVersion `• iOS`	string	No	OS version constant specific to iOS.
systemName `• iOS`	string	No	OS name constant specific to iOS.

```
File: App-platform.js
05: export default function App() {
06:     return (
07:         <ScrollView contentContainerStyle={styles.container}>
08:             <Text>OS:
09:                 <Text style={styles.value}>{Platform.OS}</Text>
10:             </Text>
11:             <Text>OS Version:
12:                 <Text style={styles.value}>{Platform.Version}</Text>
13:             </Text>
14:             <Text>isTV:
15:                 <Text style={styles.value}>{Platform.isTV.toString()}</Text>
16:             </Text>
17:             {Platform.OS === 'ios' && <>
18:                 <Text>isPad</Text>
19:                 <Text style={styles.value}>{Platform.isPad.toString()}</Text>
20:             </>}
21:             <Text>Constants</Text>
22:             <Text style={styles.value}>
23:                 {JSON.stringify(Platform.constants, null, 2)}
24:             </Text>
25:         </ScrollView>
26:     );
27: }
```
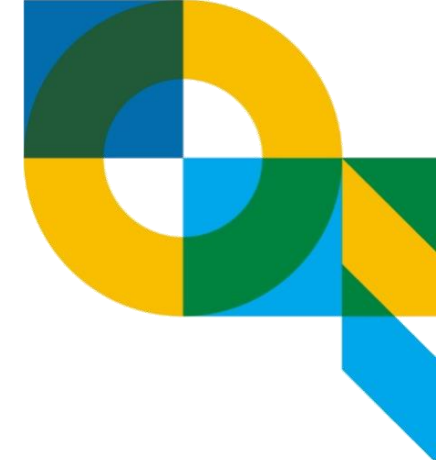
OS:android
OS Version:25
isTV:false
Constants

{
  "Fingerprint": "OnePlus/OnePlus3/OnePlus3T:7.1.1/NMF26F/
10122113:user/release-keys",
  "Version": 25,
  "Release": "7.1.1",
  "Serial": "5cb3c6239811",
  "isTesting": false,
  "uiMode": "normal",
  "Manufacturer": "OnePlus",
  "Model": "ONEPLUS A3010",
  "reactNativeVersion": {
    "prerelease": null,
    "major": 0,
    "patch": 0,
    "minor": 69
  },
  "Brand": "OnePlus"
}

練習一下:
嘗試讀取系統
資訊

useWindowDimensions

獲取螢幕解析度

useWindowDimensions

```
import { useWindowDimensions } from 'react-native';
```

useWindowDimensions 会在屏幕尺寸变化时自动更新获取到的设备 width 和 height 值。使用方法如下：

```
const windowWidth = useWindowDimensions().width;
const windowHeight = useWindowDimensions().height;
```

useWindowDimensions

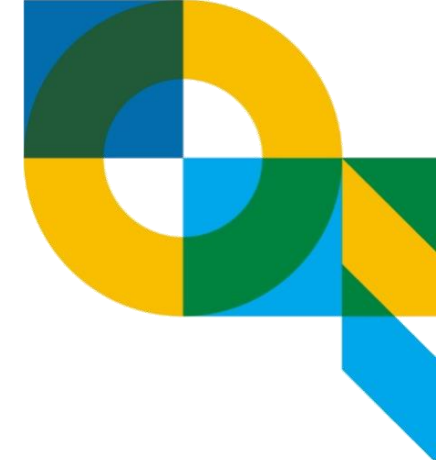| Properties	Description
fontScale	The scale of the font currently used. Some operating systems allow users to scale their font sizes larger or smaller for reading comfort. This property will let you know what is in effect.
height	The height in pixels of the window or screen your app occupies.
scale	The pixel ratio of the device your app is running on. • A value of 1 indicates PPI/DPI of 96 (76 on some platforms). • 2 indicates a Retina or high DPI display.
width	The width in pixels of the window or screen your app occupies.

```
File: App-useWindowDimensions.js
04: const App = () => {
05:   const window = useWindowDimensions();
06:   return (
07:     <View style={styles.container}>
08:       <Text>{`Window Dimensions: height ${window.height}, width ${window.width}`}</Text>
09:       <Text>{`Window fontScale: ${window.fontScale}`}</Text>
10:       <Text>{`Window scale: ${window.scale}`}</Text>
11:     </View>
12:   );
13: }
```

Window Dimensions: height 853.3333333333334, width 480
Window fontScale: 1
Window scale: 1.5

練習一下:
嘗試讀取螢幕
資訊

Appearance

Appearance

```
import { Appearance } from 'react-native';
```

The `Appearance` module exposes information about the user's appearance preferences, such as their preferred color scheme (light or dark).

`Appearance` 模块用来检测用户是否开启了暗色模式（夜间模式）：

```
const colorScheme = Appearance.getColorScheme();
if (colorScheme === 'dark') {
  // 用户开启了暗色模式
}
```
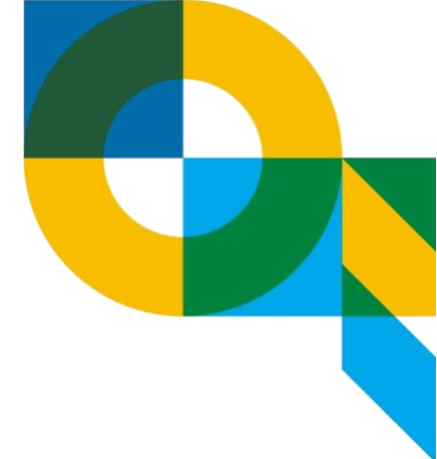
Appearance

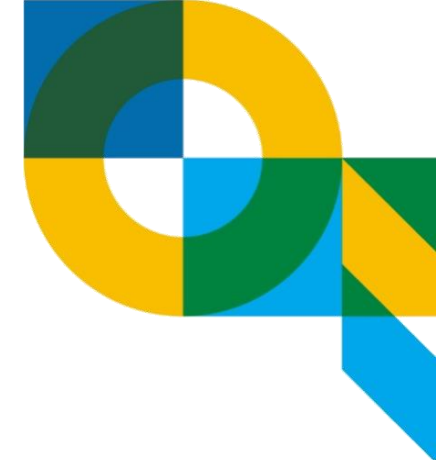| Methods	Description
getColorScheme()	Indicates the current user preferred color scheme. The value may be updated later, either through direct user action (e.g. theme selection in device settings) or on a schedule (e.g. light and dark themes that follow the day/night cycle). Supported color schemes: | VALUE	DESCRIPTION
-------	-------------
"light"	使用者選擇了淺色模式。
"dark"	使用者選擇了深色模式。
null	使用者沒有明確選擇某種模式。	
addChangeListener()	• Add an event handler that is fired when appearance preferences change. • Use the remove() method on the event subscription returned by addChangeListener().

```
File: App-Appearance.js
04: const App = () => {
05:    const colorScheme = Appearance.getColorScheme();
06:    return (
07:      <View style={styles.container}>
08:        <Text style={{fontSize:50}}>用戶開啟 {colorScheme} 模式 </Text>
09:      </View>
10:    );
11: }
```

用戶開啟 light 模式

練習一下

- 建立一個頁面，顯示前面提到取得的環境變數
  - Platform
  - useWindowDimensions
  - Appearance

Q&A