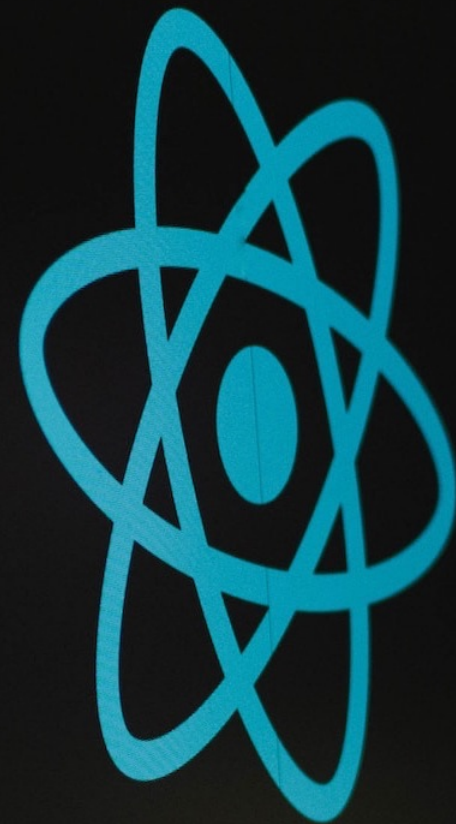


# Props 屬性

Eddy Chang

✉ [hello@eddychang.me](mailto:hello@eddychang.me)



Edit src/App.js and save to  
Learn React

# Props 屬性

固定的(靜態的)屬性值，來自父母元件或是預設值

## State(狀態)

行進速度  
輪胎磨損情況  
行進距離  
...



## Props(屬性)

車身尺寸  
品牌  
顏色  
重量  
...

## State(狀態)

目前輸入的文字  
是否正在輸入文字中  
...



## Props(屬性)

寬度  
高度  
預設文字  
...

# Props 屬性

父母元件對子女元件的溝通方式(資料傳遞方式)

React 中只有單方向的資料流動，只有父母元件能傳遞資料給子女元件(P->C)

## 父母元件

```
function Parent() {  
  return <Child text="今天開始學 React">  
}
```

## 子女元件

```
function Child(props) {  
  return <h1>{props.text}</h1>  
}
```

# Props 屬性 - 傳入參數預設值



重要必記

當父母元件沒給定值時會應用預設屬性值

```
function Child({ name = 'Eddy', text = 'Hello' }) {  
  return (  
    <div>  
      {name}  
      {text}  
    </div>  
  )  
}
```

# Props 屬性 - 屬性類型檢查

用於檢查(限制)傳入屬性的類型(只有警告訊息)

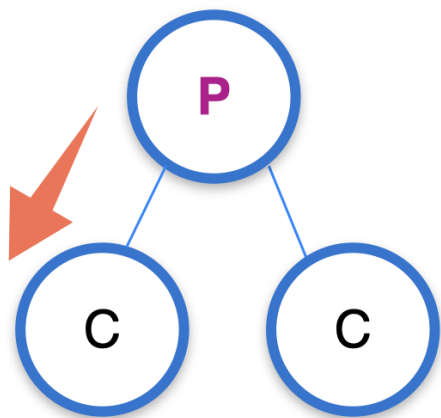
**PropTypes** 套件需要額外再安裝，安裝指令：`npm install prop-types`

```
import PropTypes from 'prop-types'

function Child(props) {
  return <h1>{props.text}</h1>
}

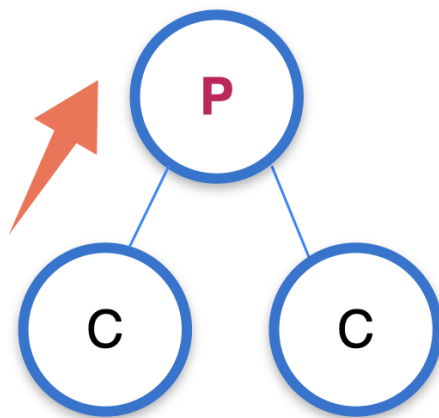
// 設定屬性的限制類型或是必填
Child.propTypes = {
  text: PropTypes.string.isRequired,
}
```

# Props 屬性 - 三種基本元件溝通方式



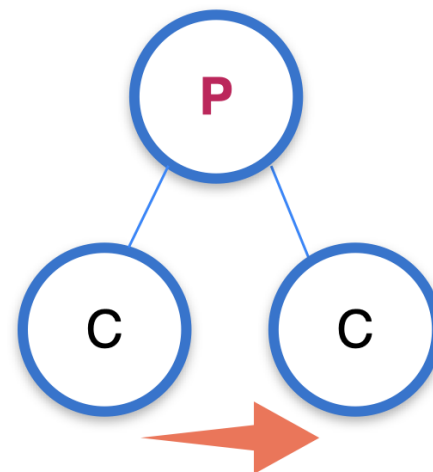
父母 -> 子女

原本就有的單向資料流



子女 -> 父母

利用callback函式



子女 -> 子女

利用父母元件與前述兩種方式

## State(狀態) vs Props(屬性)

-	State(狀態)	Props(屬性)
用途	元件內部的私有變數	在元件層級間傳遞資料
資料類型	物件(類別型元件) / 自訂(函式型元件)	物件
在此元件中可否改變	可(讀+寫)	不可(唯讀)
可否被其它元件存取	不可	可
可否被父母元件改變	不可(唯讀)	可(讀+寫)

## 屬性是唯讀的(Props is read-only)






React 會假定你開發的所有元件都是一個 **純函式(pure function)**

這代表不應該改變任何你的元件中用於渲染的輸入值，這包含了 **props**, **state** 與 **context**。要更動畫面的話，要用“設定(set)” state 而不是直接改變已存在的物件值

參考來源: [react.dev](https://react.dev), [reactjs.org](https://reactjs.org)

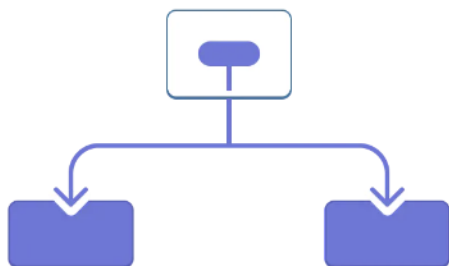


## Context (上下文/環境)

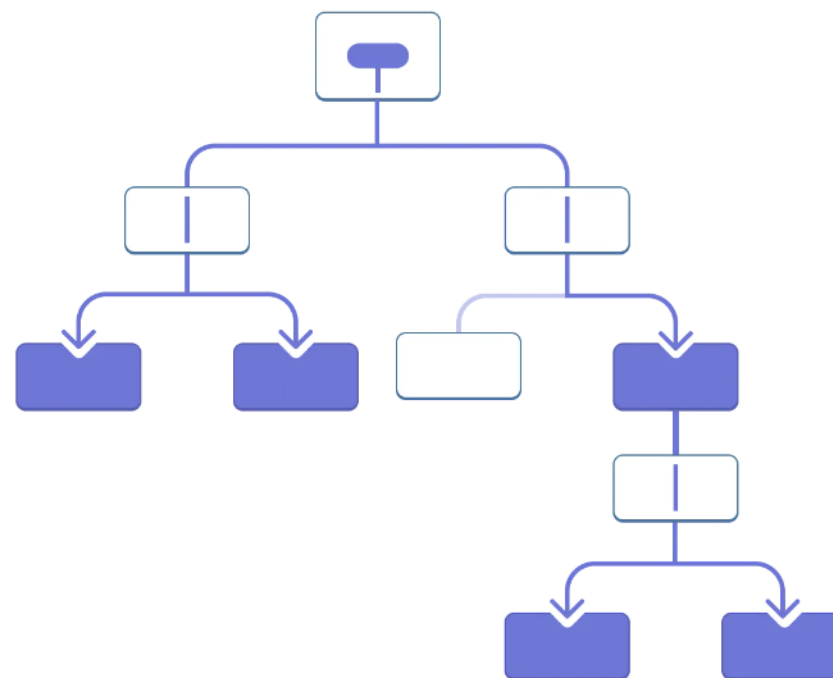
-  Context 可以讓一個元件提供資料，給在其下樹狀結構的所有子元件
-  Context 可以穿透其中階層的任何的元件(蟲洞/傳送門 teleport)
-  Context 的使用目的，是為了"共享狀態"，讓子元件能"適應符合周邊環境改變"
-  常用的 Context 情況為: 1.)登入狀態 2.)語言 3.)佈景樣式 4.)路由套件搭配
-  Context 在使用前建議先使用 props 機制傳值，如產生 props drilling(鑽孔取探)時，再使用 Context 解決，過渡使用也會產生其它問題(效能...)

# Props Drilling(鑽孔採取)

Lifting state up



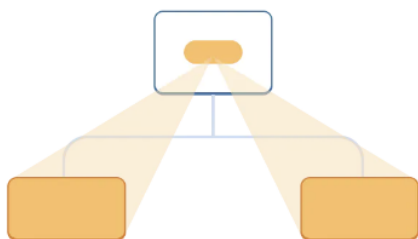
Prop drilling



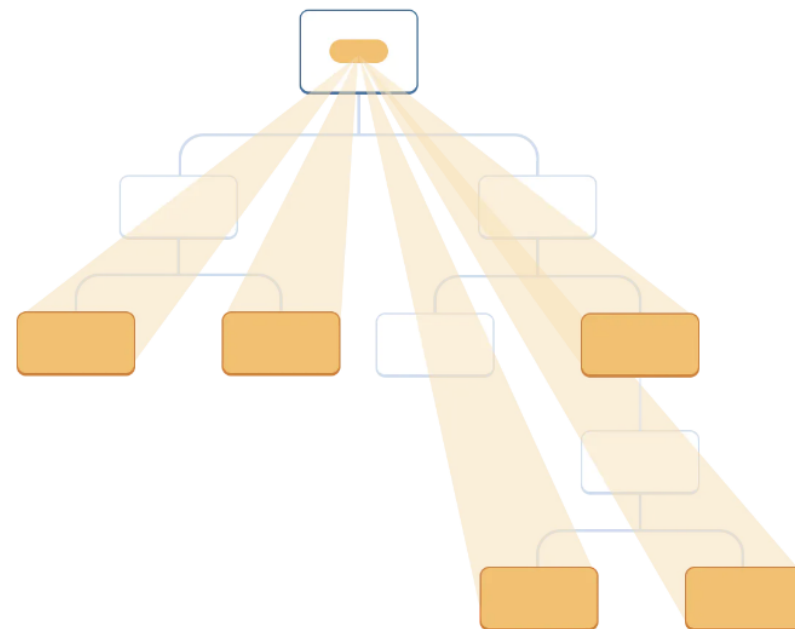
來源: [react.dev](https://react.dev)

# Context 應用示意圖

Using context in close children



Using context in distant children



來源: [react.dev](https://react.dev)

## Context 使用三步驟

1. 建立與導出它:

```
export const MyContext = createContext(defaultValue)
```

2. 在任何子元件層級深度，使用 `useContext` 勾子讀取它:

```
useContext(MyContext)
```

3. 最外(上)元件階層包裹提供者元件，讓父母元件可以提供它:

```
<MyContext.Provider value={...}>
```