全域狀態方案

	useState	useReducer + useContext	Redux
適合專案	小型專案/一般應用	中型專案	中大型專案/特殊應用
建議情況	元件數量 < 20 DOM層級 < 5	元件數量 20~50 DOM層級 5~10	元件數量 50+ DOM層級 10+
實作/學習難度	容易	中階	困難
適合狀態情況	一般值(數字、字串、布林)居多	有許多物件/陣列值 更動狀態改為發送 action 到 reducer	物件/陣列值且複雜 更動狀態要改為發送 action 到 reducer
實際內部設計	預先設定好的 useReducer 版本	Redux的陽春版	針對React全域狀態流行的解決方案
向下傳遞狀態	利用props	利用Context	利用Context(綁定器)
優點	大部份情況適用，學習與開發容易快速	可精確操控狀態轉變邏輯，解決大部份複雜狀態更新與最佳化問題	可以擴充中介軟體、獨立處理副作用，專案程式碼邏輯區分清楚，容易進行除錯與作最佳化

useReducer

為了管理更複雜的狀態(State)值與轉變運作

- useState 是它的已設定好的版本，所以它可以取代原本的useState作法，但整個程式碼的邏輯運作方式需要改寫
- 當需要複雜的 state 邏輯，要更精確的操控狀態轉變，其中包括多個子數值，或下一個 state 依賴前一個 state (通常是一個物件或陣列)
- useReducer傳入參數：(state, action) => newState
(reducer，歸納/累加函式)
- useReducer回傳為 [state, dispatch]
(state為狀態值，dispatch是用來發送動作物件的函式)

useReducer的設計概念都是從Redux來，可以說它是陽春版的Redux功能，或是擷取了一部份的設計

useReducer

從 useState 到 useReducer 改變

- 撰寫 reducer ，它是一個描述 state 將會因 action(動作) 如何改變的函式(純粹函式)：(state, action) => newState
- 當要改變狀態時，從原本的用setState的方法，變為發送動作物件 setXXX -> dispatch(action)
- action(動作) 是一個純物件值，內容有類型(type)和資料值(通常稱為payload，有效資料)