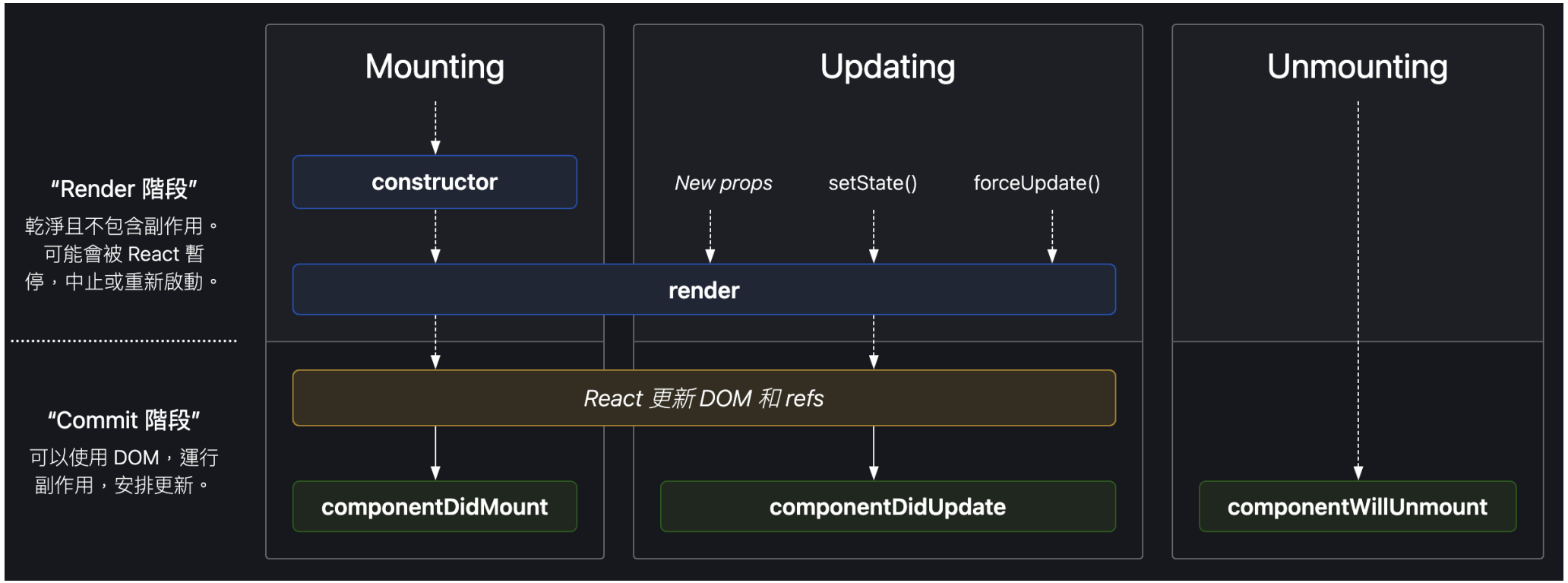
生命周期 & useEffect

Eddy Chang

✉️ hello@eddychang.me

元件生命周期

> React 元件在網頁上的各個執行階段。(註: 圖中是"類別型元件"的生命周期方法)



> 來源: react-lifecycle-methods-diagram

CSR SSG SSR

元件生命周期

Mount(掛載) - 元件首次在網頁上被渲染呈現

Update(更新) - 在網頁上的元件內容被更新

[⭐ 非常重要!] 當元件從父母元件接收到新的 props 或自己本身的 state 改變

Unmount(卸載) - 元件從網頁上被移除

> 註: 在網頁沒被看見，並不代表沒存在於 DOM 中，可能是隱藏起來。卸載是完全從 DOM 中移除。

`Effects` (作用) 是什麼

是用來定義會因為渲染本身所產生的副作用，而不是用事件函式定義

> 例如: 在聊天室中送出訊息，是由於使用者點按某個按鈕後，觸發執行某個事件處理函式。但聊天室一開始啟動時，需要連接某個伺服器，這會是一個 `Effect` (作用)，因為它是需要在元件呈現後，與伺服器進行互動的。 `Effect` (作用)會在畫面更動完成後，才會執行，這是 React 用來與某些外部系統同步化的時間點(例如網路或第三方函式庫)

> 註: React 中的 `Effect` 專用詞，指的是由執行 `render` (渲染) 時造成的(或互為關聯的) `Side Effect` (副作用)。與一般電腦科學中定義的副作用會有所不同。
> 來源: react.dev

CSR SSG SSR

`Effects` (作用) 是什麼

是可以進行反應(reactive)的程式碼區塊

- 當在 Effects(作用)裡面的變數的值有更動時，Effects(作用)可以重新進行同步化。與事件處理函式不同的是，事件處理函式是當有互動觸發時會執行一次，Effects(作用)則是在有需要時，都可以執行同步化。
- 每個 Effects(作用)應該是完整的、獨立的同步化用執行程序
- Effects(作用) 可以與反應的變數值互動反應。這些可反應的值稱為"相依變數(dependencies)"。Effects(作用)會以它們來決定是否要執行其中的程式碼區塊(進行同步化)

相依變數(dependencies)的選擇

> [⭐ 非常重要!] 主要是 props 或 state ，因為只有這兩種會觸發重新渲染(re-render)

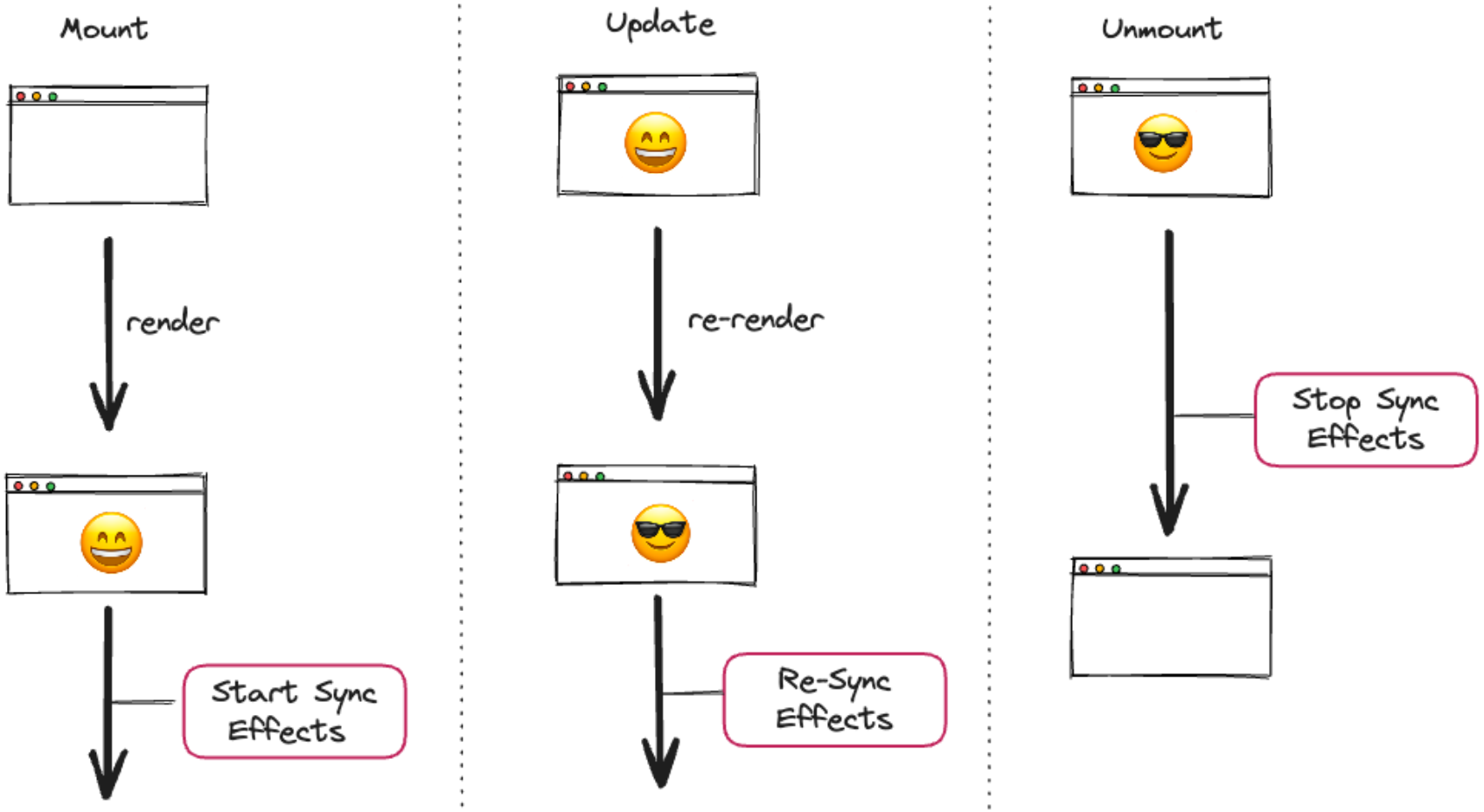
- 主要是 props 或 state 或和這兩種相關的計算得來的變數(這和元件的生命周期 update(更新)期有關)
- 不和任何變數相依(空陣列 `[]`)，即為元件 mount 期間(mount 後執行一次)
- 避免使用物件、陣列或函式，因為只要有任何改變會重新執行同步(物件或陣列相等比較必定是不同值)
- 全域變數例如 `location.pathname` 不能當相依變數(它不會觸發 re-render)
- Refs 例如 `ref.current` 不能當相依變數(它不會觸發 re-render)

Effect(作用) 的同步化觀念

> [⭐ 非常重要!] 前後的時間點很重要

- 當元件掛載(mount) 後 -> 開始進行同步(start synchronizing)
- 當元件卸載(unmount) 前 -> 停止同步(stop synchronizing)
- 當元件更新(update) 後 -> 重新同步(re-synchronizing)

Effects (作用)與元件生命周期



CSR SSG SSR

useEffect 使用情況

- 原本的生命周期方法是專門設計給"類別型元件"使用的，**不是**函式型元件，函式型元件可以用 useEffect 來模擬，但注意行為不是 100%相同
- `mount` 之後進行同步化。代表元件"已經"出現在網頁上，這個方法中可以使用直接 DOM 處理，或向伺服器要初始化資料的 JS 程式碼。
  > 註: 類似於 DOM 已載入完成的事件
- `update` 之後進行同步化。代表元件"已經"更新完成(真實 DOM)，這個方法中可以得到最後更新的狀態值 (只有 state 更新，或接收到新的 props)
  > 註: 類似於 state 或 props 的 change 事件
- `unmount` 之前進行同步化。代表元件"即將"被移除到 DOM 外，通常稱是作 cleanup(清理)的時間點。

useEffect 基本樣式

```
useEffect(() => {
  // 這裡每次render(渲染)都會被執行
})

// mount
useEffect(() => {
  // 這裡只有當元件呈現(mount)(初次render(渲染))會執行一次
}, [])

// mount+update
useEffect(() => {
  // 這裡當元件呈現(mount)(初次render(渲染))會執行一次，以及在a或b有改變時，在改變後會執行
}, [a, b])
```

useEffect 基本樣式 - cleanup

> `unmount`(卸載)前的`cleanup`(清理)工作，都需要搭配`mount`的程式碼來定義，所以用途就是，在這元件將被移除前的清理工作，例如與伺服器斷開連線、清除記憶體、清除計時器...etc，使用的時機和方式很固定

```
useEffect(() => {
  const connection = createConnection(serverUrl, roomId)
  connection.connect()

  return () => {
    // cleanup(清理)
    connection.disconnect()
  }
}, [])
```

useEffect 使用注意

- React 18 加入了 `StrictMode` 檢查工具，這會在有渲染情況時，多次觸發執行 useEffect 中的程式碼，如果不是你要的行為，或造成運作不正常可以將之關閉。(註: 它只會在開發期間有用，營運後沒這功能。經常會出現在與外部服務相連時，或第三方登出入功能經常會導致錯誤)
- useEffect 有可能會造成無窮迴圈，導致程式(或瀏覽器)整個當掉，在 useEffect 中使用 setState 務必小心
- 另有一個稱為 `useLayoutEffect` 的勾子，它的執行時間點會是在瀏覽器重畫(painting)前。如果畫面上在變動時發生閃爍時，可以試著改用它，但要注意它有可能會擋住瀏覽器進行繪製畫面，造成畫面空白。
- 雖然經常見到使用 `useEffect` 來作 fetch 資料，但這種方式問題非常之多，實務上應改用其它這類更理想的擴充類函式庫(如swr, React Query), RTK Query)