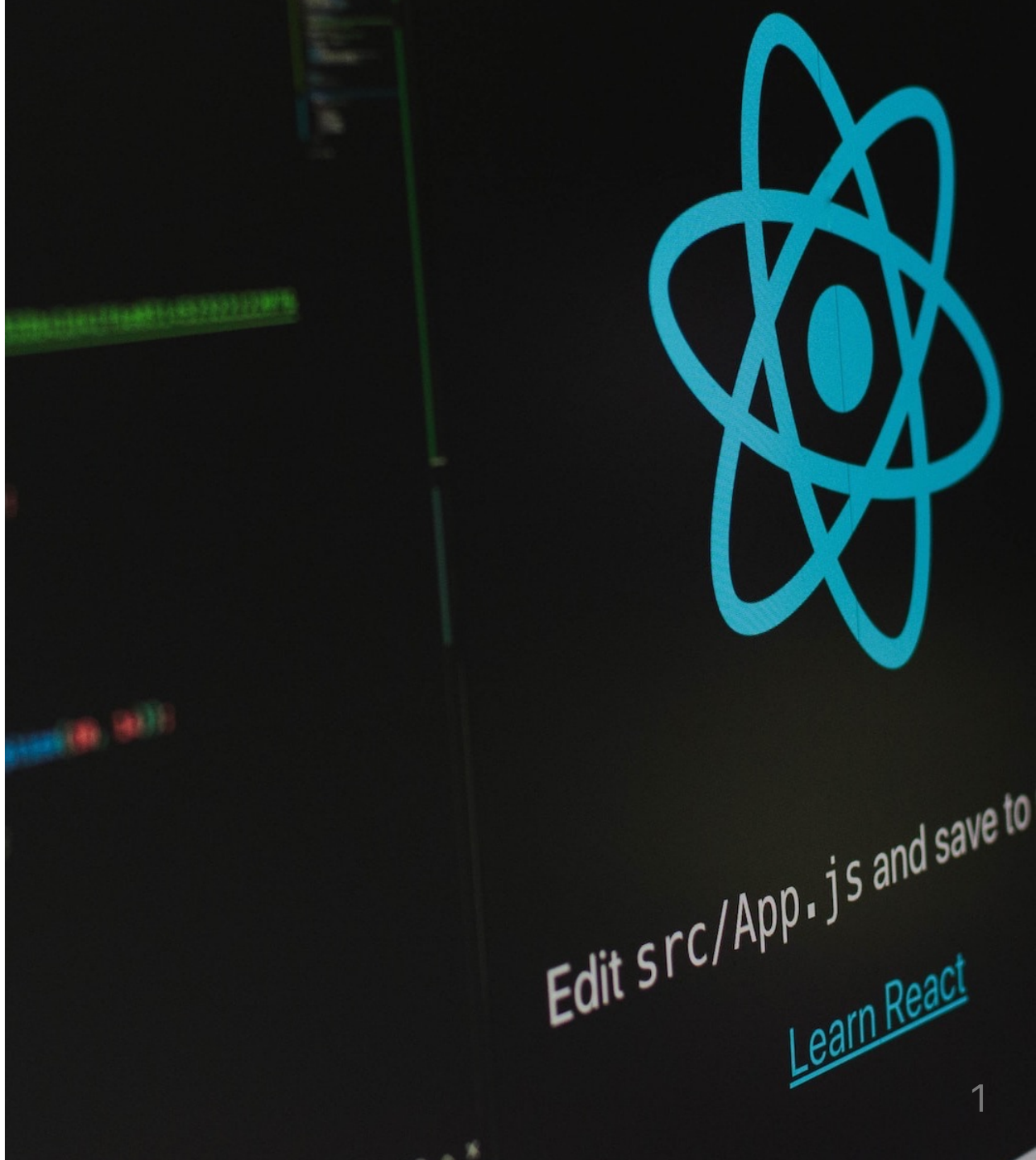
表單元件 & Refs

Eddy Chang

✉️ hello@eddychang.me



可控(controlled) vs 不可控(uncontrolled)

-	可控(controlled)	不可控(uncontrolled)
說明	由 React 控管資料狀態的表單元素	脫離 React 掌控狀態，原本的 DOM 的表單元件，需由開發者自訂控管資料的表單元件
應用搭配	State(狀態)	Refs(參照)
資料動態檢查	容易達成	送出時才檢查/開發者自訂處理
資料管理	父母元件中管理	DOM 中由開發者自行管理
建議使用	大部份情況	少部份情況 (ex.有需要整合非 react 應用時)

常用表單元素

種類	標記	值屬性	事件callback	更新事件值 (針對 callback)	備註
文字輸入框	**<input type="text" />**	value="string"	onChange	e.target.value	
複選框 (核取方塊)	<input type="checkbox" />	checked={boolean}	onChange	e.target.checked	
選項按鈕	<input type="radio" />	checked={boolean}	onChange	e.target.checked	
文字輸入區域	<textarea />	value="string"	onChange	e.target.value	與HTML不同，用value代表其中輸入文字
下拉式選單	**<select>**	**value="option value"**	**onChange**	**e.target.value**	與HTML不同，用value代表被選中的選項值(並非使用selected屬性)

CSR SSG SSR

可控表單元件

1. `value` 屬性必需對應到某個 state(狀態)
2. 元件的事件處理函式必需可以更動到 `value` 屬性的對應 state(狀態)值

> 註: 類型 `checkbox` / `radio` 指的是 `checked` 屬性)

```
function Form() {
  const [firstName, setFirstName] = useState('') // 宣告狀態變數
  // ...
  return (
    <input
      value={firstName} // 強制input輸入框的值要符合狀態值
      onChange={(e) => setFirstName(e.target.value)} // 任何更新都會改變到狀態值
    />
  )
}
```

CSR SSG SSR

不可控表單元件

> 註: `defaultValue` 和 `defaultChecked` 是"不可控表單元件"在設定初始值用的

```
function MyForm() {
  return (
    <>
      <label>
        Text input: <input name="myInput" defaultValue="Some initial value" />
      </label>
      <hr />
      <label>
        Checkbox:
        <input type="checkbox" name="myCheckbox" defaultChecked={true} />
      </label>
    </>
  )
}
```

CSR SSG SSR

不可控表單元件 - 使用 Refs

```
import { useRef } from 'react'

export default function Form() {
  // 1. 初始值為`null`，要對應表單元素的API，例如`getElementById`獲取元素參照時，沒得到時會回傳`null`
  const inputRef = useRef(null)

  function handleClick() {
    // 3. 這裡已得到元件參照，可以直接呼叫API
    inputRef.current.focus()
  }

  return (
    <>
      {/* 2. 這裡要定義ref值對應 */}
      <input ref={inputRef} />
      <button onClick={handleClick}>Focus the input</button>
    </>
  )
}
```

Refs

> Refs 可以讓元件保持住，某些不需要用於渲染的資訊，例如 DOM 節點元件或是計時用 ID。在 React 正規模式中，Refs 應該被當成是一個"逃生出口"的作法。它會在需要整合非 React 的系統，例如瀏覽器的內建 API 時很有用。

1. `useRef(initialValue)` 只會回傳一個物件值: `{current: initialValue}`
2. 每個 ref 中包含的資料是獨立的，並不會在元件中共享
3. Refs 定位也是 React 中的變數。但它和狀態不同，`ref.current` 值是可以改變的(讀寫的)，而且更動 ref 並不會觸發重新渲染(re-render)
4. 不要直接在渲染時讀寫 ref，應在事件處理函式或 `useEffect` 中讀寫它(參考文件)
5. ref 並不是 props(屬性)，也不會自動在元件間階層傳遞。要傳遞 ref 要用 forwardRef

表單驗證方式

1. 開發者自行處理，針對簡單的表單應用情況
2. [⛔ 相容問題所以不推薦] 以 HTML5 表單 `<form>` 驗証的方式，針對簡單表單的應用情況 (不同瀏覽器的錯誤訊息呈現與文字會有差異)
3. 專門的 React 表單套件，針對複雜的表單應用情況(ex. `formik`, `React Hook Form`)