Mobile Web Creator Prototype

周昱安 / 賴彥凱
2016-10-14

前端技術堆疊

- ReactJS
  - View Client-Side Rendering
- Redux
  - Client-Side State Container
- React Router
  - Client-Side Routing with ReactJS

React：僅僅是 UI

- React 本身並不是一個完整的前端框架，只是一個函式庫
- React 本身只處理 View 的部份，也就是 HTML 的 DOM
- React 是一個中間媒介，連結了 UI 純粹的定義層面與 DOM 的實際層面

- 基本上，React 本身只做以下兩件工作；
  - 讓你定義 UI 要長怎樣
  - 幫你把這個 UI 印出來到使用者的瀏覽器畫面上

React：聲明式的定義前端 UI

- 前端 UI 程式碼本身，應該要足以完整的自我表達其擁有的行為與可能的顯示變化
- UI 渲染無法避免邏輯，將 UI 的定義直接在 JavaScript 中進行，有助於提高 UI 的自我表達能力

```
class AlertButton extends React.Component {
  constructor(props) {
    super(props);
    this.alertText = this.alertText.bind(this);
  }
  alertText() {
    alert(this.props.text);
  }
  render() {
    return (
      <button onClick={this.alertText}>我是一個 {this.props.text} 按鈕</button>
    );
  }
}
```

React : JSX

- JSX 是 React 在使用的一種特殊 JavaScript 語法，看起來像是在 JavaScript 程式碼中直接寫 HTML
- 能幫助開發 React 應用，因可讀性高、很類似 HTML
- 瀏覽器看不懂，需要編譯成原生的 JS 程式碼才能正常的在瀏覽器上執行

```
<button id="btn" onClick={this.handleClick}>我是一個按鈕</button>
```

```
React.createElement("button", {id: "btn", onClick: this.handleClick}, "我是一個按鈕");
```

React : Component

- 自定義元素 (Component)
- 以 Component 的形式來表達與拼裝 UI，能夠讓前端 UI 程式碼有更好的可組合性與可重用性，這點對於我們想做的網頁 UI 產生器功能來說非常適合

```
class AlertPage extends React.Component {
  render() {
    return (
      <div>
        <AlertButton text='hello'/>
        <AlertButton text='zet'/>
        <AlertButton text='moomoo'/>
      </div>
    );
  }
}
```

```
class AlertButton extends React.Component {
  constructor(props) {
    super(props);
    this.alertText = this.alertText.bind(this);
  }
  alertText() {
    alert(this.props.text);
  }
  render() {
    return (
      <button onClick={this.alertText}>我是一個 {this.props.text} 按鈕</button>
    );
  }
}
```

React : Props

- Component 外部（父 Component）傳遞給 Component 內部的靜態參數
- 抽象化出跟問題有關的參數，方便 Component 進行重用
- Props 傳遞到 Component 內部後，應是不可變更的固定值
- 當 Component 外部改變傳遞進來的 Props 時，Component 內部會自動發起重繪

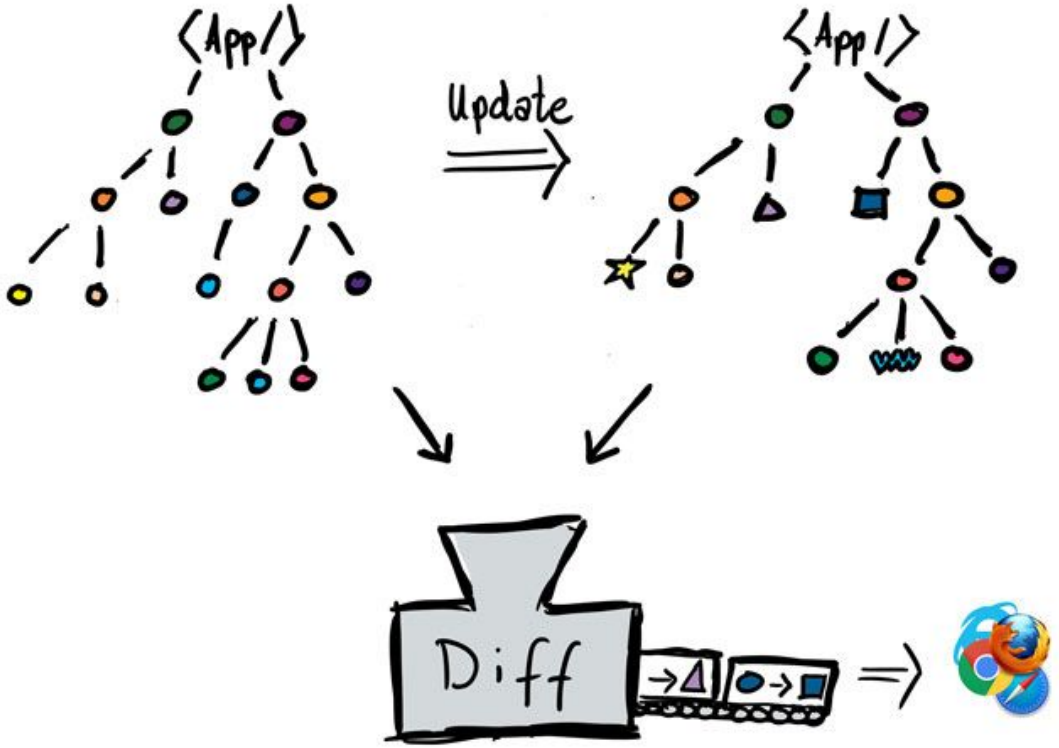
React : Virtual DOM

- Virtual DOM 是一份純資料的 Tree 物件，對應到實際的 DOM
- Virtual DOM 為自定義 Component 提供了中介的虛擬層，讓開發者能描述 UI 樣貌與邏輯
- 我們透過定義 Component 來表達「UI 什麼情況該如何呈現」。而「要用什麼手段來達到這個畫面改變（如何操作 DOM）」，React 則會自動幫你做

Always Redraw

- Single Source of Truth
  - UI 要長得如何，應當是取決於當時的 Model 資料以及 UI 狀態而決定
- 把畫面全部洗掉重新再畫，顯示結果通常一定是正確的
- 每次都重繪全部的實體 DOM 顯然是不可行，但是重繪 VDOM 則相對高效許多

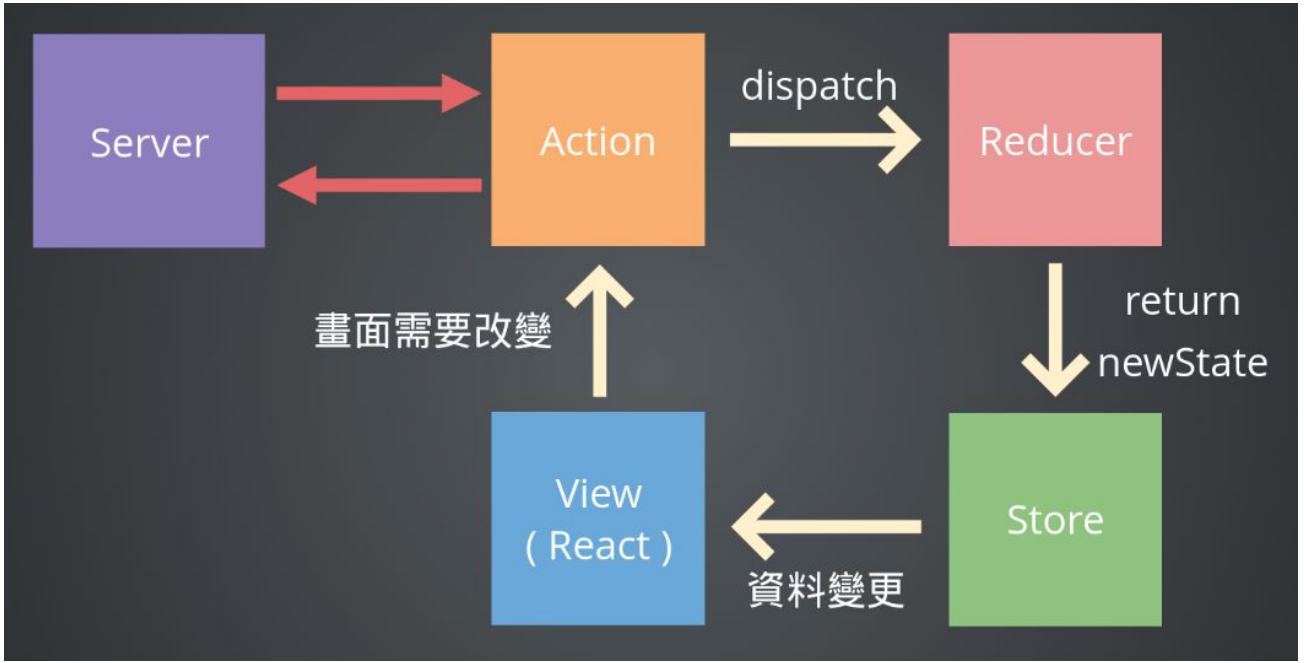
Always Redraw



Redux

- Redux 是 JavaScript 的狀態容器，提供可預測化的狀態管理
- 採用單向資料流（One-way Dataflow）的概念
- 單一資料源，整個前端應用程式的 State 都儲存在同一個 Store 當中
- 搭配 React 使用時，將 Store 中的資料以 Props 的形式傳進 Component

Redux 的 One-way Dataflow



Demo