第 6 章 - 樣式 style

全域樣式 global css

只有`pages/_app.js`這檔案中可以直接導入，其它元件(或頁面)都不行，這是 Next 中有自動靜態最佳化之類機制的限制。

```
import '@/styles/globals.css'

function MyApp({ Component, pageProps }) {
  return <Component {...pageProps} />
}

export default MyApp
```

其它元件(或頁面)直接導入 css 檔案會出現下面的警告:

```
Global CSS cannot be imported from files other than your Custom <App>. Due
to the Global nature of stylesheets, and to avoid conflicts, Please move
all first-party global CSS imports to pages/_app.js. Or convert the import
to Component-Level CSS (CSS Modules).
```

最多這裡能應用，的應用策略就針對不同頁面或區域分各名稱的 CSS 檔案，然後在`pages/_app.js`中導入。CSS 類別名稱和覆蓋需要自行解決:

```
import '@/styles/globals.scss'
import '@/styles/product.scss'
import '@/styles/member.scss'
```

結論

優點 & 適用情況:

- Next 內建
- 最簡單
- 可再搭配 SASS 技術

缺點 & 不適用情況:

- 無法搭配各別元件單獨使用，不易拆分和管理
- 不適合中大型專案，延伸性不足

CSS-in-JS - 內聯樣式 (inline style)

意指直接把 CSS 寫在 JSX 中，這是原本的 style 屬性語法也稱如下:

```
export default function HiThere() {
  return <p style={{ color: 'red' }}>hi there</p>
}
```

CSS-in-JS - styled-jsx

用法介紹

> 註: `<style jsx>`與原本 HTML 中就有的、很少用到的`<style>`標記，執行的行為並不相同。

Next 內建的styled-jsx，透過`<style jsx>...</style>`加入純 CSS 的樣式在裡面，如下範例:

```
export default function HelloWorld() {
  return (
    <div>
      Hello world
      <p>scoped!</p>
      <style jsx>{`
        p {
          color: blue;
        }
        div {
          background: red;
        }
        @media (max-width: 600px) {
          div {
            background: blue;
          }
        }
      `}</style>
    </div>
  )
}
```

`<style global jsx>`是將其中的 CSS 全域化(global)，一般用途有兩種。

一是注入樣式在這個元件(頁面)，但套用的有可能是外部的父母元件或元素，由於它會比全域 css 更接近本元件(頁面)，所以會覆蓋掉原本全域 css 中的同樣樣式定義，但只會作用在本元件(頁面)上，其它元件(頁面)並不受影響:

```
export default function PageOne() {
  return (
    <div>
      PageOne
      <style jsx global>{`
        body {
```

```
            background: red;
          }
        `}</style>
      </div>
    )
  }
```

另一種用途與上述相似，通常是用來套用在這個元件中的其它元件的樣式，如下範例:

```
import Image from 'next/image'

export default function PageThree() {
  return (
    <div>
      <Image src="/next.svg" className="icon" width={100} height={40}
alt="" />
      PageThree
      <style jsx>{`
        div :global(.icon:hover) {
          cursor: pointer;
        }
      `}</style>
    </div>
  )
}
```

因為 styled-jsx 也是寫在 JSX 語法中，樣式是寫在樣版字串中，所以它也是能用表達式運算，來達成動態樣式的，如下範例:

```
export default function MyButton(props) {
  return (
    <button>
      {props.children}
      <style jsx>{`
        button {
          padding: ${'large' in props ? '50' : '20'}px;
          background: ${props.theme.background};
          color: #999;
          display: inline-block;
          font-size: 1em;
        }
      `}</style>
    </button>
  )
}
```

結論

優點 & 適用情況:

- Next 內建
- 容易達成動態樣式，而且使用的是原本 CSS 語法
- 適用簡單地快速、單獨套用樣式的元件(頁面)情況

缺點 & 不適用情況:

- 樣式與程式碼都混在 JSX 一起，造成閱讀困難(但可以另外定義檔案再用導入或定義在元件外)
- 支援性比不上其它知名套件(如 styled-components)

其它搭配(visual studio code 擴充外掛):

- vscode: styled-jsx Syntax Highlighting
- vscode: styled-jsx Language Server
- vscode: styled-jsx

CSS Modules

用法介紹

CSS Modules 是現今很流行的另一種在 React 元件中套用樣式的解決方案，Next 內建支援，也建議使用它在各別的元件或頁面套用自訂樣式時使用。

用法是獨立出副檔名為 `.module.css` 的樣式檔案，導入後會轉換為物件值，在真正進行編譯或渲染到網頁上時，會產生帶有 hash 碼(一種隨機的 ID 字串)的類別名稱，以避免 CSS 類別名稱套用上的衝突。(和其它解決方案如 `styled-jsx` 很相似，都是自動產生的)

```
/* styles/dashboard.module.css */
.dashboard {
  padding: 24px;
}
```

```
// app/dashboard/layout.js
import styles from './dashboard.module.css'

export default function DashboardLayout({ children }) {
  // 這裡最後輸出時會產生一個有hash碼的類別名稱，例如`style_dashboard__s0dsl`
  return <section className={styles.dashboard}>{children}</section>
}
```

CSS Modules 預設就是`:local`的本地作用域(指的是目前使用的本元件而已)，這不需特別加上。也允許使用 `:global` 全域樣式，加上關鍵字後不會產生有 hash 碼(隨機的 ID 字串)的類別名稱，例如下面的例子:

> 註: Next 要求在使用`:global`時至少要在有一層 local 的類別或 ID 裡才能使用，相對原本的 CSS Module 官網上說明的，或其它 React 專案用法上會更嚴格。

```
/* 如果用styles.myclass套用，將不會產生 hash 碼 */
.myclass :global .tweet {
  background-color: red;
}

.myclass :global .user {
  background-color: green;
}
```

套用範例:

```
return (
  <div className={styles.myclass}>
    <div className="user">Hello</div>
  </div>
)
```

會使用`:global`全域樣式，通常也是為了要覆蓋從全域來的 CSS 類別名稱，但這在意義上有些違背了設計這種語法的原意，所以並不是它專長使用的情況，當然是不適合在這這時機發揮。

結論

優點 & 適用情況:

- Next 內建，技術成熟、使用很廣泛、支援性很好
- 用一般的 CSS 撰寫語法，無學習門檻
- 可搭配 SASS
- 容易分離各元件樣式與組織管理
- 主要用來解決 CSS 類別名稱衝突情況

缺點 & 不適用情況:

- 不適合要使用動態樣式(帶變數決定樣式)的情況
- 不適合用於覆蓋全域語法(在複雜時不容易撰寫)
- 不適合太多層的巢狀定義，雖可配合 SASS，但套用時也要一層層手動套用

其它流行的解決方案

- styled-components
- Tailwind CSS
- Emotion