第 2 章 - 命名規則

程式設計中，對於檔案或變數是有其一定的命名規則，除了與各程式語言中關於合法命名的條件有關外，另外也有從很早期發展出來的命名通則，開發程式時需考量這些通則，而非混合或自創命名規則。

命名規則通常需要與開發小組協商後，統一一種開發風格。不論使用何種命名規則(或風格)，重點在於團隊開發時要**"保持一致性"**。Next 專案只需遵守幾個基本的原則，並沒有嚴格的命名規則。

本教學採用的是 Next 官方教學與範例專案中的主要命名規則。

命名規則摘要

資料夾與檔案命名規則:

1. 所有"資料夾"與"檔案"均使用`kebab-case`(烤肉串命名法，或稱`dash-case`)，意即各英文字詞間以"全小寫"加上以連接符號(-)分隔。例如`product-layout.js`代表為`ProductLaout`元件檔案
2. 檔案命名時只使用連接符號(`-`)，其它可用符號如`$`或_均不使用
3. 元件程式碼副檔名只使用`js`。目前不使用`jsx`, `ts`, `tsx`
4. 數字作為命名應為一個單詞。例如元件名稱為`Counter1`的檔案名稱應為`counter-1`，而非`counter1`。建除非有必要才使用數字命名，建議例如使用為`counter-one`或`counter-a`會較清楚識別

函式、類別、變數命名規則:

1. React 元件使用`CamelCase`(大駝峰，又稱巴斯卡`PascalCase`)命名，例如`ProductLayout`, `CartButton`
2. class(類別)使用`CamelCase`(大駝峰)
3. 函式、變數、常數等識別名稱，使用`camelCase`(小駝峰)命名，例如`useDocsRoute`, `currentRoute`
4. 可用符號例如`$`或_均不使用

常用動作詞

- get/set: 得到/設定
- use: 使用(註: 不論是函式與檔案，請保留給 React hooks(勾子)使用，它都是開頭為`use`的函式)
- get/post/put/delete: 對應 RESTful 的動作
- create/read/update/delete: 對應 CRUD(建立、讀取、更新、刪除) 四種操作動作使用
- login/logout/register: 會員登入/登出/註冊
- signin/signout/signup: 會員登入/登出/註冊
- add/new: 新增
- delete/remove: 刪除
- edit: 編輯
- reset: 重置
- update: 更新
- load: 載入
- search: 搜尋
- send: 送出
- handle/do: 處理/作(某件事)

- make/create/generate/produce: 製造/建立/產生

範例:

- 從資料夾獲得使用者資料: `getProfile`或`getUserProfile`
- 新增商品到購物車中: `addToCart`

資料夾/檔案名稱命名

對`pages`目錄而言，只能使用`kebab-case`命名規則，它與網頁的路由路徑有相關，因此它的別名又稱為`slug-case`或`friendly-url-case`。

`Slug /slʌg/`是一些用人類可閱讀的字詞, 來描述或識別一個網頁的網址，它是網址的一部份，通常位於網址最末端。Slug 經是網頁標題的網址友善化版本，直接由頁面標題自動產生(英文)，可以達到更佳的 SEO。

例如我們要描述一個球鞋商品分類頁面的網址，單純用 id 的 SEO 不如使用 slug 好，`/category/:id/products`相比於`/category/:slug/products`後者較好，實例像是`/category/10995/products`相比於`/category/nike-mens-shoes/products`後者較好。

> 更多資訊參考: Slug(wikipedia)或What's a slug

結論是在此專案中目錄的資料夾與檔案，建議都統一使用`kebab-case`命名法，例如 `my-header.js`檔案中的元件名稱應為`MyHeader`，要注意兩種命名法間的轉換。

在此`pages`資料夾中的範例:

`about.js`檔名 (注意元件中的函式命名要用英文開頭大寫):

```
// Location: /pages/about.js
export default function About() {
  return <div>About</div>
}
```

`what-is-next.js`檔名 (注意元件中的函式命名要用英文開頭大寫):

```
// Location: /pages/what-is-next.js
export default function WhatIsNext() {
  return <div>What is Next</div>
}
```

> 註: 目前不建議使用以英文全小寫的`snake_case`(蛇形命名法)或其它檔案、變數命名法。

React 元件命名規則

在程式碼檔案內容中，React 自訂元件因為有強制規定，元件的函式(或類別)的名稱，要求一定為**"英文大寫開頭"**，所以會使用 `CamelCase`(大駝峰)命名法。

輔助 page 元件的資料夾一律使用對應的資料夾名稱，例如輔助`pages/index.js`頁面的`Home`元件，資料夾使用`components/home`命名；輔助`pages/product/cart.js`的`Cart`元件頁面，應有對應的資料夾使用

`components/product/cart.js`命名。

> 註: `Home`元件通常是 `pages` 目錄中的 `index.js` 檔案，它會是個例外的特別命名。一般常見也可以用 `root` 或 `landing` 來命名

> 註: 資料夾中的 `index.js` 相當於此資料夾的根檔案，例如`pages/product/index.js`的路由相當於`/product`

CSS Moudle 命名規則

輔助 page 元件的檔案一律使用對應的檔案名稱，例如輔助`home.js`的 css module 檔案使用 `home.moudle.css`。

資料夾與檔案命名與上述元件命名說明一致。

其它問答

常看到將除了頁面外的 React 元件檔案，均以`CamelCase`命名，例如`ProductList.js`，為什麼教學上不使用這種命名規則？

1. Next 官方的範例專案均使用`kebab-case`的檔案命名，只是參考這種命名方式
2. 只是為了一致性，也為了避免造成命名混亂，改為`kebab-case`的檔案命名，不會花費太多時間和造成影響
3. 如果有需要改為`CamelCase`命名，請協調所有團隊小組成員一致修改

資料夾的分類均是以依照類型分類(by type)，有網路上的文章，認為應該要依照用途分類(by feature)才能較適合大規模專案？

1. 依照用途分類(by feature)的確更適合大規模專案，但它需要進一步的再模組化分類，需要利用 module 導出入、重新導出入特性，先預作額外設計，相對來說對初學者較為困難且複雜
2. 大型專案必定還會加入其它的特定或專用功能，這些都需要額外制訂規則和調整結構，並非單純地依照用途分類(by feature)，就能通用各專案中，都需要視實際應用情況調整
3. 目前教學上的專題實作，並非大型專案，使用依照用途分類(by feature)或模組化後會顯得更難理解和分類，為了效率使用依照類型分類(by type)會更易於進行
4. 目前教學上專案結構未來也可以依照依規劃，再重構後或重新制定結構再修正