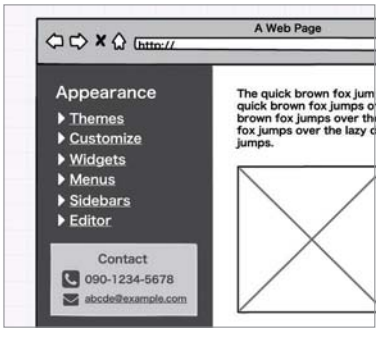
第 2 章

缺少 CSS 設計會遇到的困擾

那麼，在正式討論 CSS 設計之前，再稍微談點其他事情。
有些人會覺得：「我已經有在編寫 CSS 了，沒有遇到什麼大問題啊！」
不，若真是如此的話，根本不會閱讀到這邊……吧。
針對抱有這類想法的人，本章將會說明不經思考的隨意編寫 CSS 會遇到哪些困擾。

總之先寫再說

學會幾個屬性而感到興奮的 A 職員，喃喃自語：「CSS 簡單啦～～」當天就寫了網站側邊欄的程式碼。



開頭先寫個 id、類別，再套用樣式。編寫的程式碼如下：

```
/* 導覽標題 */
.nav dt {
  color: white;
  background color: black;
}

/* 導覽項目 */
.nav dd {
  ...
}
```

第 3 章

先來瞭解 BEM

嗯，CSS 設計很重要。那麼，趕緊來學習 CSS 設計吧。首先，第一步要先瞭解 BEM。

本章將會大致講解 BEM 的內容。

何謂 BEM ？

BEM 是下述三個單字的字首，可直接拼唸成「bem」。

- Block（區塊）
- Element（元素）
- Modifier（修飾符）

BEM 官方網站有詳細的解說內容。
若讀完本書想要深入瞭解的話，可參考下述網址：

BEM
https://en.bem.info/

第 8 章

SMACSS：基礎規範

前面講解了 BEM 的內容。「好喔！將頁面劃分不同的區塊，似乎可寫出不錯的程式碼」……有些人可能會這麼認為。然而，光靠 BEM 就能順暢無阻地編寫 CSS，卻又不是這麼一回事，肯定會碰到各式各樣的問題。下面將會解說有幫助的四個議題：

- **基礎規範**
- **布局規範**
- **主題規範**
- **功能類別**

SMACSS

這四個議題中，前面三個是根據 **SMACSS** 的思維。SMACSS 唸成「smacks」，是 Scalable and Modular Architecture for CSS 的簡稱。SMACSS 的作者是 Jonathan Snook，這本書他根據自己在 Yahoo! 從事網頁設計開發的心得所寫的。

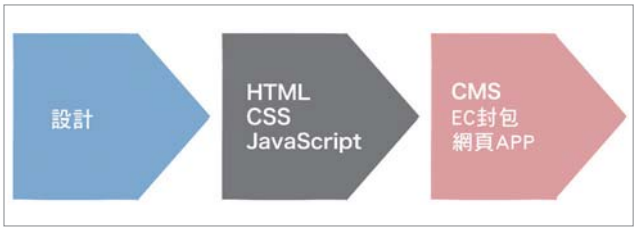
這本書籍出版於 2012 年，如今已成為經典著作，內容統整得簡單易懂，筆者認為即便是現在也有助於理解網頁設計。該書籍的頁數並不多，讀完本書後產生興趣的讀者，建議可翻閱原著看看。

SMACSS 的官網有公開完整內容，也可找到翻譯成日語的版本。

SMACSS
http://smacss.com

SMACSS: 日文版
http://smacss.com/ja

後續處理



前置處理是設計人員的工作，那後續的處理呢？這次來討論後處理，此時嵌入 CMS 管理系統、網頁 APP 的情況居多。若設想的流程是先製作模板，再據此製作大量的頁面，則量產頁面也可視為後處理。

想要自己編寫的 HTML 和 CSS 在後處理正常發揮作用，不假思索地編寫程式碼是行不通的。

- CMS 管理系統會怎麼重複這邊的內容？
- 這種 HTML 程式碼是否會對 CMS 造成過大的負擔？
- 這邊不是應當使用 CMS 產生的固定 HTML 嗎？
- 此區塊的劃分方式不會難以理解設計意圖嗎？
- 不能再製作更單純的頁面嗎？

未思考這些問題就編寫 HTML 和 CSS，反而會增加後處理的負擔，有時還會因為 HTML 不能嵌入而被退件。

例如，請回想第 10 章介紹的「SMACSS：主題規範」，將標頭部分的 HTML 當作共通程式碼。雖然未必需要此實作，但是否留心這層顧慮，將大幅改變編寫 HTML 和 CSS 的工作價值。儘管難以獲得外部的讚賞，但至少可得到團隊成員感謝的眼神。

如前所述，編寫 HTML 和 CSS 所需的能力是，實作時能夠　併考慮前後處理，並理解專案需要哪些東西。徒有優秀的技術能力，是無法達成這個要求。

第 19 章

使用建置製作 CSS：Sass

本章將會介紹 Sass。

在深入編寫 CSS 的時候，肯定會選用 Sass、PostCSS 等更加靈活編寫 CSS 的工具。為了方便學習理解，建議先瞭解 Sass 能夠做到哪些事情。

何謂 Sass ？

首先，**Sass** 是什麼？Sass 是 CSS 的擴充套件語言，全稱為 Syntactically awesome style sheets，可直譯為「語法令人驚艷的樣式表」。

前章解說建置時有提到：

「想辦法最佳化 CSS，或者轉換成其他格式的文字檔案，加載經過處理的 CSS 檔案至瀏覽器。」

在非 CSS 的其他格式中，當屬 Sass 最為有名。Sass 具備諸多強力輔助 CSS 設計的功能，能夠解決編寫 CSS 時遭遇的各種問題。

Sass 擁有許多 CSS 沒有的語法，編寫後程式碼可由其他程式轉成 CSS。雖然最終想要的是 CSS 檔案，但通常會使用更有效率的 Sass 編寫，再將程式碼轉成 CSS 格式。這類進行前置處理的程式稱為「**預處理器（preprocessor）**」，而用來轉換 Sass 的程式稱為「**CSS 預處理器**」。

為了幫助讀者瞭解 Sass，下面舉例幾個 Sass 語法轉換前後的程式碼來解說。

即便不用特別安裝軟體，透過 SassMeister 的網站，就可於瀏覽器上確認 Sass 的轉換結果。

相較於編寫 20px、30px，後者的程式碼更能清楚表達意圖。跟調色盤的做法　樣，如下定義整個網站使用的留白類型，再從中選擇來使用。

```
$spacing xs: 10px;
$spacing s: 20px;
$spacing m: 30px;
$spacing l: 40px;
$spacing xl: 50px;
```

使用這樣的變數編寫 CSS，自然能夠完成留白整齊的使用者介面。跟色彩設計的情況　樣，當留白的類型過多時，建議與設計人員多加討論。「這邊應該統整相同的樣式比較好吧？」等等，最好能夠像這樣交換意見。

與 Sass 的交流方式

那麼，專案應該導入 Sass 嗎？「不用直接編寫 CSS 真神奇，但大家都在用嗎？應該使用嗎？使用後沒有問題嗎？」有些人會感到不安吧。

雖然看法見仁見智，但對於學習、導入 Sass 的問題，筆者認為根本不必感到猶豫，Sass 已經廣泛應用於各處。

CoffeeScript

其實，還有諸多程式語言如同 Sass，為了彌補既存技術的缺點而創造出來。

例如 CoffeeScript 跟 Sass 相似，是以轉成 JavaScript 為前提的程式語言。

CoffeeScript
https://coffeescript.org/

第 22 章

進階元件：通用型區塊、限定型區塊

接著，回歸討論設計的議題。

本章將會講解區塊不好命名的問題。

即便瞭解 BEM，掌握基礎規範、布局規範、Sass 等，仍舊會對命名感到困擾吧。

該取什麼名稱呢？

那麼，假設有如右的區塊。

你接著要編寫 HTML 和 CSS。拿到 FAQ 頁面的設計圖，發現裡頭有這樣的使用者介面。「好喔！把它作成區塊吧。」內心如此盤算，正準備開始編寫程式碼，雙手卻停了下來。

```
<div class "
```

程式碼寫到這邊，開始煩惱：「嗯……區塊該取什麼名稱呢？」例如，你可能想到這些名稱：

- box-text-set
- contact-block
- faq-contact-block

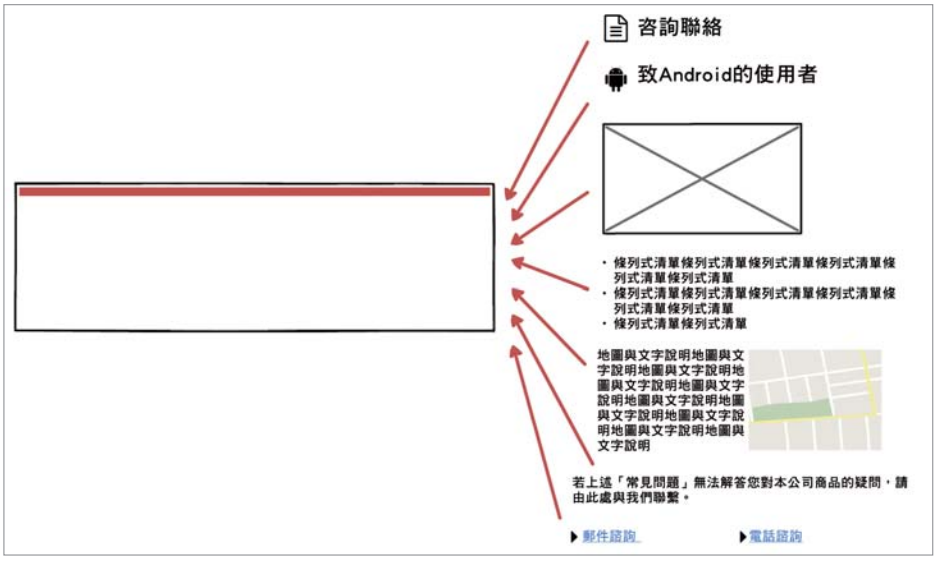
假設有分別選擇這三種名稱的開發人員。

嘗試詢問這三位開發人員的意見。

開發人員 A：box-text-set

使用者介面的周圍有框線、裡頭裝有內文，故取名為 `box-text-set`。

雖然光由設計圖無法斷定，但區塊似乎也可裝進其他的元素。設計圖中有諮詢導覽，不過這僅是 box-text-set 的部分功用吧？比如，裡頭也可能裝進圖片、清單列表。



日後可能追加額外的使用者介面，每次都得準備新的元素並搭配相關要素。製作區塊時，應該設想挪用至其他頁面的情況。

製作便利的通用型區塊架設整個網站，正是 CSS 設計的有趣之處。CSS 的檔案容量也降到最小，不是非常理想嗎？

開發人員 B：contact-block

此使用者介面貌似用於問答區塊，故取名為 `contact-block`。

由設計圖可知，區塊置於 FAQ 頁面的最底部。

這是設計圖的 FAQ 頁面，不過該區塊也有可能用於其他頁面，此時應當統整成同樣的外觀吧。

難道不會用於諮詢聯絡以外的用途嗎？

我不曉得像 A　樣考慮任何用途是否比較好。總而言之，這是用於諮詢的區塊，取為 `contact-block` 不會有太大的問題。

畢竟設計圖還只有 FAQ 的頁面，若取為 `box-test-set` 的話，根本不曉得該區塊的用途、用於什麼地方吧。

開發人員 C：faq-contact-block

我同意這是用於諮詢頁面的區塊，但會不會用於其他頁面就難說了。畢竟拿到的設計圖還只有 FAQ 的頁面，目前根本不曉得其他頁面的情況，而且我僅負責 FAQ 頁面的程式碼而已。因為是用於 FAQ 頁面的諮詢區塊，故 `faq-contact-block` 是最佳的命名。

其他頁面出現相同的使用者介面時怎麼辦？到時再取名 `product-contact-block`、`top-contact-block` 等，作成其他的區塊就好了啊。

況且，也不曉得那些頁面出現的諮詢區塊，外觀是否與 FAQ 中的區塊外觀相同。沒有更多的資訊，也就只能取名 `faq-contact-block`。

通用型還是限定型？

這個案例應該選擇哪　種命名中呢？

- A : box-text-set
- B : contact-block
- C : faq-contact-block

「這三種命名方式，正確答案是 A ！」沒有辦法如此下結論。

三者的差異在於，是設想**通用型**區塊還是**限定型**區塊。

其中，A 最為通用、C 最為限定、B 介於兩者之間。不如說，筆者正是如此設想來舉例的。

「原來如此？就算這麼說，還是不知道命名的基準吧？」讀者或許這麼認為。因此，下方就來討論各種命名的優缺點。

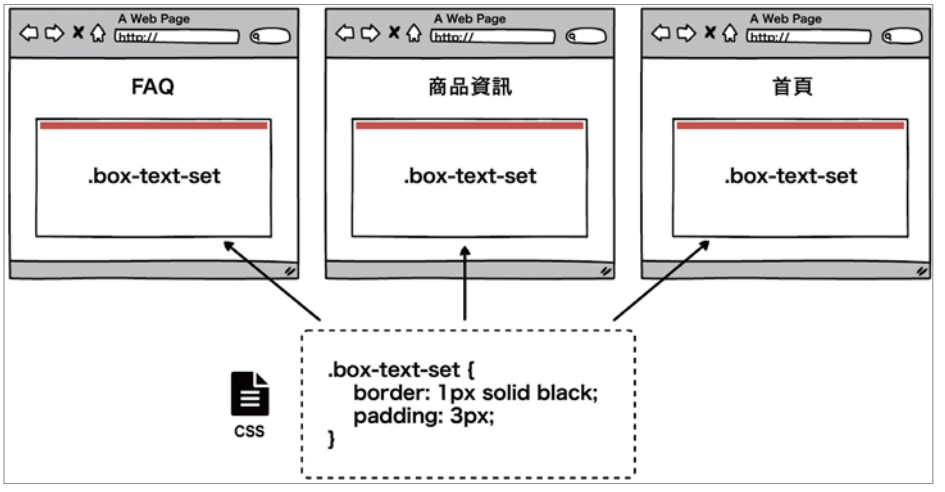
通用型名稱的優點

區塊決定取通用型名稱，並用於整個網站有什麼好處？這是選擇開發人員 A 的 `box-text-set`。在開發人員的 A 的意見中，就有提到　些優點了。

容易將變數反映至複數頁面

首先，第　個優點是只需要修改 CSS，含有該使用者介面的頁面，全部能夠套用同樣的變更。「……CSS 平時不是就這樣編寫嗎？」內心可能感到疑惑。

改變 `box-contact-block` 的背景顏色後，無論是 100 個頁面還是 1000 個頁面，完全不需要動到 HTML，所有頁面都會反映布局的變更。



若是取名為 `faq-contact-block`，僅設想用於 FAQ 頁面的話，情況會如何呢？如前所述，同樣的區塊在商品資訊得取為 `product-contact-block`；在首頁得取為 `top-contact-block` 等。

這樣即便僅想要稍微調整布局，好幾處都得做同樣的變更。複數頁面使用同　的區塊，日後想要調整區塊外觀時會相當輕鬆。連續接到這樣的委託時，肯定會讚嘆：「CSS 棒呆了！」

將 CSS 的檔案容量降到最小

相較於 FAQ、商品資訊、首頁分別製作區塊的 CSS 容量，以 `box-text-set` 統　製作僅需要少量的程式碼。比起三個區塊的 CSS，　個區塊的 CSS 當然比較少。

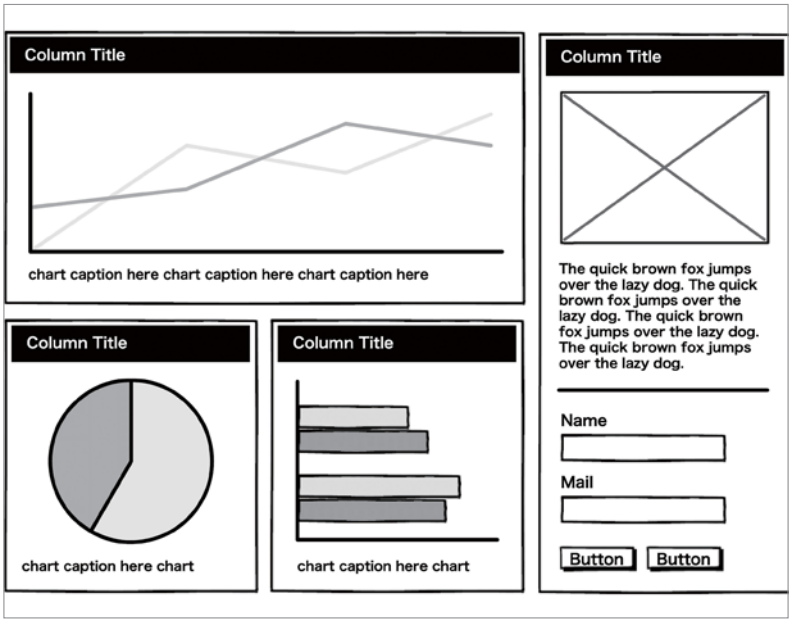
若僅是些微的布局差異，且採用 BEM 形式設計的話，可使用修飾符來修改。截然不同的外觀會直接製作其他的模組，但僅是稍微改變框線的顏色、調整裡頭的文字大小，使用修飾符修改　個區塊，CSS 的程式碼會最為精簡。

相似的區塊取不同的名稱來製作，到頭來許多地方還得複製貼上同樣的 CSS。這種情況總是令人莫名不快，筆者也能夠感同身受。`box-text-set` 正是「　源多用（one source multi-use）」的思維。

優點

將區塊作成嵌套的優點有，防止單　區塊的 CSS 量增加、變得複雜。

在前面的範例中，提到可將 box-column 的內容全部看作元素。這個例子沒有問題，不過若裡頭內容的變化眾多，可能會寫得相當辛苦。
例如，請想像下述頁面：



與其說是網站，更像是網頁應用程式，各種表單組件、圖表等，裡頭裝有多樣的使用者介面。

此時，將所有要素裝進　個 box-column 區塊並非不行，但這會產生塞進眾多元素的巨量區塊。
區塊應有盡有可能令人覺得方便，巨量區塊卻無法讓人欣然接受。　個區塊中出現眾多元素的時候，必須注意元素名稱不可重複、可能發生程式碼不精簡等問題。元素過多的區塊，其程式碼也會變得複雜。
於是，設計時僅切割出內容，各自作成獨立的區塊。然後，如同開頭的舉例，將區塊裝進外框部分的 box-column。