BEM 設計模式

使用 BEM 讓程式碼更易維護

什麼是BEM？

BEM

1. 由Yandex團隊提出的 CSS class 設計模式
2. 以區塊 Blocks、元素 Elements、修飾符 Modifiers 來命名
3. 使開發者透過 class 名稱就知道個元素間的關係
4. 缺點：命名方式長而難看，但相比 BEM 格式帶來的便利來說，我們應客觀看待

過去常見 Class 寫法

```html
<div class="navbar">
  <a class="logo-mask" href="##"></a>
  <ul class="btn-box">
    <li class="btn-wrap">
      <a class="btn active" href="##">連結文字</a>
    </li>
    <li class="btn-wrap">
      <a class="btn" href="##">連結文字</a>
    </li>
    <li class="btn-wrap">
      <a class="btn" href="##">連結文字</a>
    </li>
  </ul>
</div>
```

&

```scss
.navbar {
  display: block;
}
.navbar .btn-box {
  display: flex;
}
.navbar .btn-box .btn-wrap {
  width: 300px;
}
.navbar .btn-box .btn {
  color: #fff;
}
.navbar .btn-box .btn.active {
  color: #eee;
}
```

BEM 設計模式（HTML）

```html
<div class="navbar">
  <a class="navbar__logo-mask" href="##"></a>
  <ul class="navbar__btn-box">
    <li class="navbar__btn-wrap">
      <a class="navbar__btn navbar__btn--active" href="##">連結文字</a>
    </li>
    <li class="navbar__btn-wrap">
      <a class="navbar__btn" href="##">連結文字</a>
    </li>
    <li class="navbar__btn-wrap">
      <a class="navbar__btn" href="##">連結文字</a>
    </li>
  </ul>
</div>
```

BEM 設計模式（CSS -> SCSS）

```css
.navbar {
  display: block;
}
.navbar__btn-box {
  display: flex;
}
.navbar__btn-wrap {
  width: 300px;
}
.navbar__btn {
  color: #fff;
}
.navbar__btn--active {
  color: #eee;
}
```

->

```scss
.navbar {
  display: block;
  &__btn-box {
    display: flex;
  }
  &__btn-wrap {
    width: 300px;
  }
  &__btn {
    color: #fff;
    &--active {
      color: #eee;
    }
  }
}
```

BEM 基本架構（Block）

架構簡介

Block 區塊

每個 Block 在邏輯和功能上都是相互獨立具備自己特有的意義。在大多數情況下，任何獨立的元素都可以被視作一個區塊。

使用原則

1. Block 名稱需能清晰的表達出，其用途、功能或意義
2. 每個區塊在邏輯上和功能上相互獨立
3. 區塊是獨立的，可以在應用開發中進行複用，降低程式碼重複並提高開發效率
4. Block 可以放置在頁面上的任何位置，也可以互相嵌套

```
.header  .footer  .about  .btn  .text-sm
```

實際用例

BEM 基本架構（Element）

架構簡介

Element 元素

Element 為 Block 的一部分並且相依於 Block，使用雙底線 “__” 連接 Element，表示隸屬於 Block 底下的元素。

使用原則

1. Element 名稱需能簡單的描述出，其結構、佈局或意義
2. Block 的內部元素，都被認為是 Block 的子元素
3. 應避免使用串連方式命名 Element

```
.menu__info__list (X)  .menu__list_item (X)  .menu__list__item_link (X)

實際用例                                    .menu__info-list (V)  .menu__item (V)  .menu__link (V)
```

BEM 基本架構（Modifier）

架構簡介

Modifier 修飾符

Modifier 是定義 Block 和 Element 的外觀、狀態或類型，使用雙橫線 “--” 來連接 Modifier，表示在現有的元素樣式下做樣式微調。

使用原則

1. 能直觀易懂表達出其外觀、狀態或行為
2. 不能脫離 Block 或 Element 使用
3. 應該改變的是實體的外觀、行為或狀態，而不是替換它
4. 值可以是 Boolean 或 key-value 形式。

```
.navbar__btn--active   .navbar__btn--is-active   .navbar__btn--color-red
```

實際用例

BEM 設計模式

```
<div class="about">
  <form class="form about__form">
    <input class="form__field" type="text" />
    <div class="form__wrap form__wrap--align-right">
      <button class="btn btn--outline btn--light text-sm active">
      </button>
    </div>
  </form>
</div>
```

BEM 與命名空間的結合

命名空間提升代碼可讀性

Layout 佈局

用來定義「大架構」的 CSS 或區塊布局
例如： l-header l-about

Component 組件

組件可以包含其他組件和物件
例如： c-form c-card

State 狀態

狀態類表示物件/組件的當前狀態，
例如： is-active has-loaded

Object 物件

網頁中的最小構建塊，裡面不能包含其他物件或組件
例如： o-section-title o-btn

Utility 公共

將常用的樣式獨立出來
例如： u-text-sm u-color-red

JavaScript 插件

表示當前 html 物件使用了 JavaScript 操作，如插件、自定義行為等。
例如： js-parallax js-lazy

BEM 設計模式（命名空間的結合）

```
<div class="l-about">
  <form class="c-form l-about__form">
    <input class="c-form__field" type="text" />
    <div class="c-form__wrap c-form__wrap--align-right">
      <button class="o-btn o-btn--outline o-btn--light u-text-sm is-active">
      </button>
    </div>
  </form>
</div>
```

BEM 設計模式（修飾符變形）

說明

嚴謹 BEM 優缺點

可以清楚知道每個 class 之間的作用關係，不過書寫上太過攏長，需重複撰寫元素所依附的父層 class 名稱。

變形版解決的問題

變形版主要解決了原始 BEM 書寫過於攏長的問題，不過相對的，如果互相嵌套的 class 較多，必須謹慎思考如何擺放變形版無相依性 class 的修飾符。

變形版主要是將修飾符改以獨立 class 撰寫，並改以單橫線 “-“ 命名，但必須依附在 Block 或 Element 選擇器上，如：.btn.-outline

BEM 設計模式（修飾符變形）

```
<div class="l-about">
  <form class="c-form l-about__form">
    <input class="c-form__field" type="text" />
    <div class="c-form__wrap -align-right">
      <button class="o-btn -outline -light u-text-sm is-active">
      </button>
    </div>
  </form>
</div>
```

命名結構

網頁常用的命名層級結構

外層結構

&__outer
&__container
&__area

元素結構

&__box
&__wrap
&__list
&__item
&__group
&__content
&__img-mask
&__img

文字命名

&__slogan
&__title
&__subtitle
&__desc
&__info
&__text
&__link

尺寸命名

&-3xl
&-2xl
&-xl
&-lg
&-base
&-md
&-sm
&-xs
&-2xs

狀態命名

.is-active
.is-disabled
.is-opened
.is-valid
.has-loaded

顏色命名

.u-color-primary
.u-color-secondary
.u-color-success
.u-color-warning
.u-color-error
.u-bg-black
.u-bg-black-60
.u-bg-gray
.u-bg-gray-mid
.u-bg-gray-mid-40
.u-bg-gray-light

程式命名規則

了解程式通用命名規則

Camel Case 駝峰式

小駝峰：lowerCamelCase（JS 常用）
大駝峰：UpperCamelCase（前後端皆有使用）

Snake Case 蛇形

單字與單字間使用下劃線分隔
例如： snake_case（後端常用）

Kebab Case 肉串式

跟 Snake Case 類似，單字以橫線 - 分隔
例如： kebab-case（class 常用）

各種程式語言皆可依照個人習慣選擇不同命名規則

實際用例