CSS 排版教學

講師 劉國良
bettercalleason@gmail.com

Create your website more faster

01

沒有 CSS 的世界

如果官網沒有 CSS 長怎樣?

02

三種 CSS 套用方法

Link > Style > inline-Style

CSS link 用法

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>test</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
        ...
</body>
</html>
```

CSS style 用法

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>test</title>
    <style>
      .red {color: red;}
      .green {color: green;}
      .blue {color: blue;}
    </style>

</head>
```

CSS inline style 用法

```
<body>
<div class="box"
style="width:100px;height:100px;background:red;"></div>

<div class="box"
style="width:100px;height:100px;background:blue;"></div>

<div class="box"
style="width:100px;height:100px;background:green;"></div>


</body>
```

03

何謂 CSS 選擇器

學會之後，連爬蟲都能得心應手。

CSS 選擇器〔 1/2 〕

一次搞懂
CSS選擇器

ID 選擇器

網頁標籤唯一值，不可重複

```
#id{
  color:red;
}
```

tag 選擇器

使用 HTML 標籤來選擇

```
P {
  color:red;
}
```

Class 選擇器

多個網頁標籤共用同一種 Style

```
.class{
  color:red;
}
```

鄰近選擇器

隔壁答案借我抄

```
.class + p{ ...}
.class ~ p{ ...}
```

CSS 選擇器〔 2/2 〕

一次搞懂
CSS選擇器

我要你的第一胎

子層選擇器

```
ul > li {
    color:red;
}
```

連你孫子也不放過

孫層選擇器

```
ul li{
    color:red;
}
```

小孩才做選擇

Universal 選擇器

```
*{
    color:red;
}
```

你的 type 是什麼

屬性選擇器

```
a[href="#"] {
    color: red;
}
```

通通都是假的

偽類選擇器

```
ul li:nth-child() {
    color: purple;
}
```

3-1

何謂 CSS 權重

君不君，臣不臣
父不父，子不子

CSS 權重

Inline-style	ID	Class	Element
0,	0,	0,	0,

4-1

行內, 區塊元素

Inline, block, inline-block

什麼是行內元素 inline?

- 元素可在同一行內呈現，圖片或文字均不換行
- 不可設定寬高，元素的寬高由它的內容撐開
- padding, margin 上、下的設定不會有效果

什麼是區塊元素 block?

- 元素寬度預設會撐到最大，使其寬度占滿整個容器
- 可以設定長寬 / margin / padding，但仍會占滿一整行
- 設定 margin, padding 上下左右都會影響排版

什麼是 inline-block?

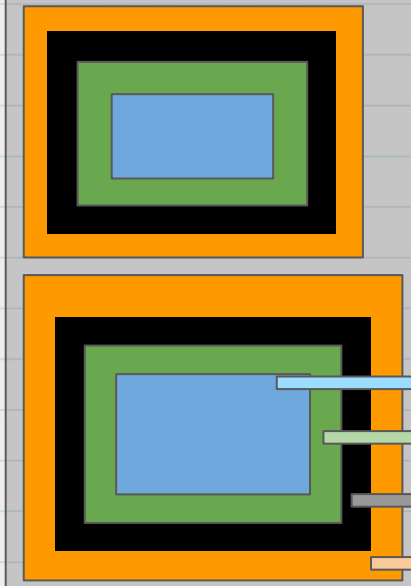
- 將預設改成 `Display:inline-block;`
- 同時具有行內元素的不換行特性
- 又保持區塊元素可以有寬高的方便排版

4-2

原始排版規則

BOX MODEL

什麼是 BOX model?



每個區塊元素可以想像成網頁裡的小盒子

依據你給他的內容物, 內邊距, 邊框, 外邊距

來決定如何呈現在網頁上,

以及網頁內的 box 要按什麼順序做排列。

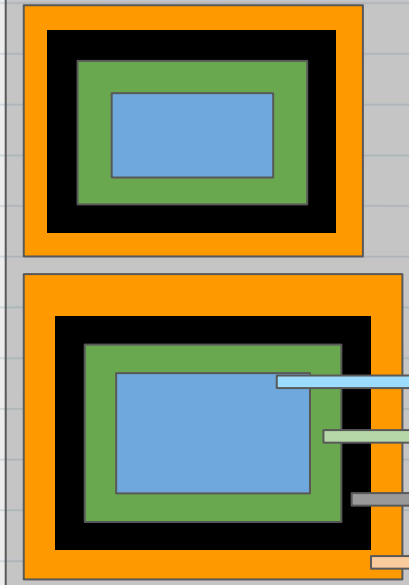
內容物（content)

內邊距（padding)

邊框（border)

外邊距（margin)

如何決定 BOX 寬度?



請問，這 box 在網頁該佔多少顯示寬度？

```
Width:200px;
Height:200px;
background:#afa;
```

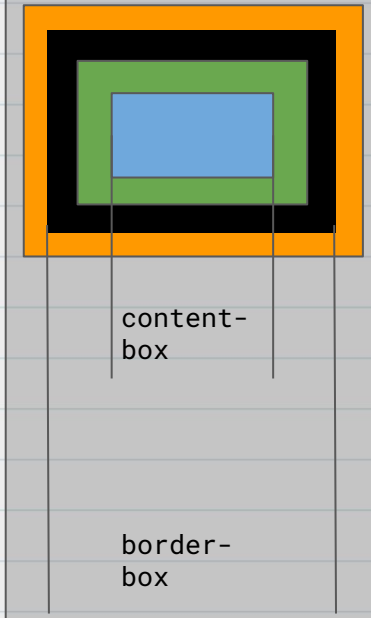
內容物（content)

內邊距（padding)

邊框（border)

外邊距（margin)

box-sizing



Box-sizing:content-box;（預設）

Box-sizing:border-box;

當我設定寬或高的時候，這數字應該算到 content 或者是 border。

5

讓區塊橫過來

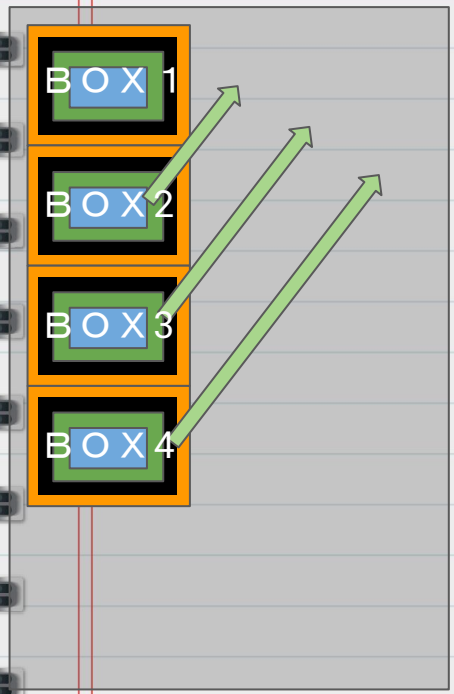
Inline-Block,Float,Grid,Flex

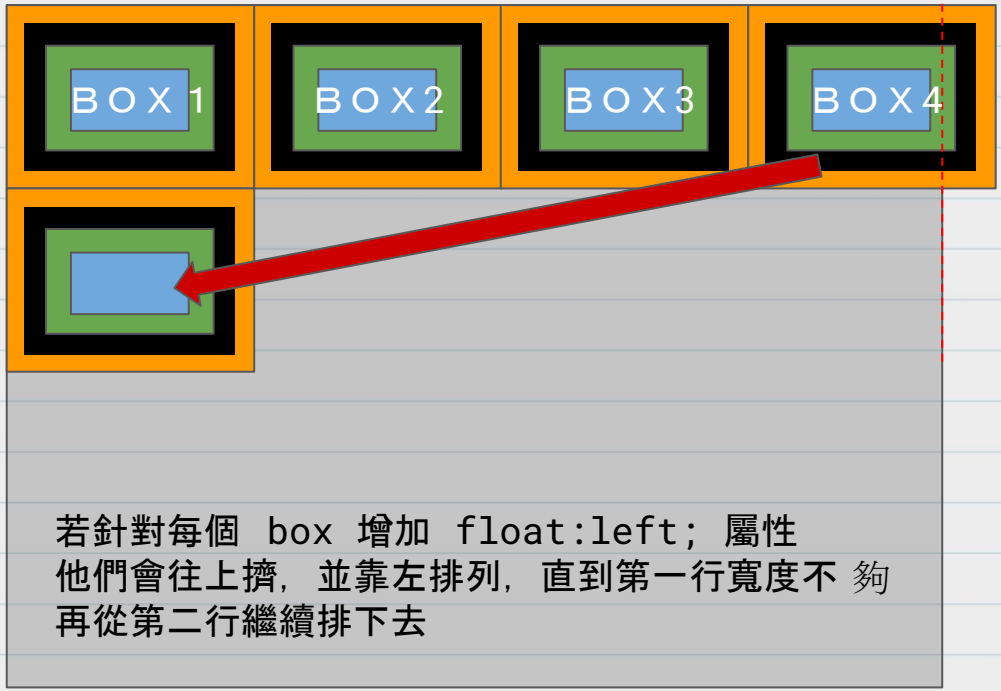
5-1

Float 排版

像氣泡般往上方漂浮

有空位就往上擠





若針對每個 box 增加 float:left; 屬性
他們會往上擠，並靠左排列，直到第一行寬度不夠
再從第二行繼續排下去

5-2

Float 排版

文繞圖

像雜誌一般的排版

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Sequi officia fugit dolores rem deleniti recusandae eaque aliquam culpa corporis provident, aperiam quasi nihil commodi obcaecati eligendi magni? Debitis suscipit cupiditate nesciunt fuga itaque corrupti tenetur illum est esse sunt. Obcaecati sequi amet nemo veniam molestias ex? Id debitis unde earum modi, itaque vitae soluta deleniti quam error repellendus, aut numquam inventore nisi recusandae impedit reprehenderit culpa ut optio nam repudiandae voluptatibus quaerat! Amet, inventore placeat? Temporibus, quam. Aut quasi, alias, ea est perferendis iure modi quaerat cupiditate non reprehenderit exercitationem ipsa a animi minus eligendi itaque voluptates fuga minima asperiores. Iure, quisquam. Eligendi non

5-3

Float 排版

Clear fix

解決高度消失的問題

在 box 使用 float 之後，全部不按照原規則排列
一直往上飄的緣故，使得父層在計算高度時會 產生問題
由於不知道要飄到什麼時候，所以在還沒結束漂浮之前
都無法判斷總高度到底是多少。

如果要解決這個問題，我們必須針對最後一個區塊設定
一個停止漂浮的屬性： clear:both;
我們習慣上稱之為 clear fix。

兩種 clear fix 用法

1.在下一個 DIV 做 clear
https://codepen.io/easonliu913/pen/BaKpvVM

2.在父層 DIV 做 clear
https://codepen.io/easonliu913/pen/OJNWror

6

Grid 排版

改善可怕的惡夢排版

Grid 排版方法

- Grid-template-columns: 定義容器有多少欄 (給寬度值)

  Grid-template-rows: 定義容器有多少列 (給高度值)

```
display:grid;
grid-template-columns: 240px auto;
Grid-template-rows: 100px 200px 1fr;
```

注意： 設定除了 px 之外，還有 1fr 與 auto 的關係。

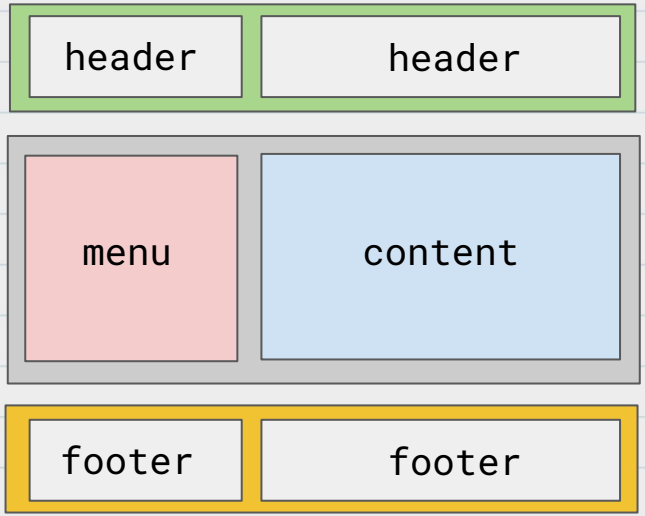
- Row-gap, Column-gap, Gap: 設定欄位之間的距離

```
row-gap: 5px; // column-gap: 5px; // gap: 5px 5px;
```

其實就是類似表格的排版感覺，但是長寬設定更加彈性了。

Grid 排版方法

```
Grid-template-areas:
"header header "
"menu content"
"footer footer ";
```



其實就是類似表格的排版感覺，但是區塊更加彈性了。

7

Flex 排版

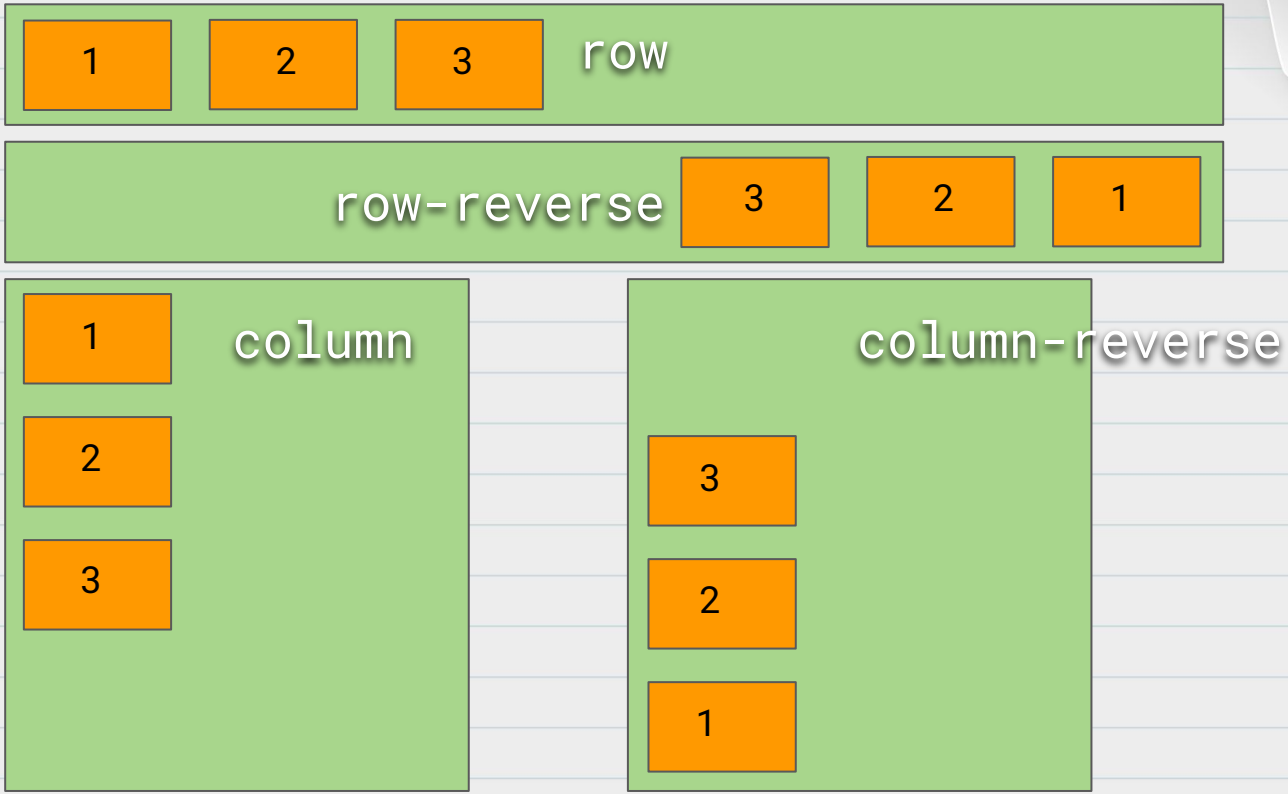
現今主流的彈性排版首選

7-1

flex-direction

方向

Flex: flex-direction



7-2

justify-content

主軸排列方式

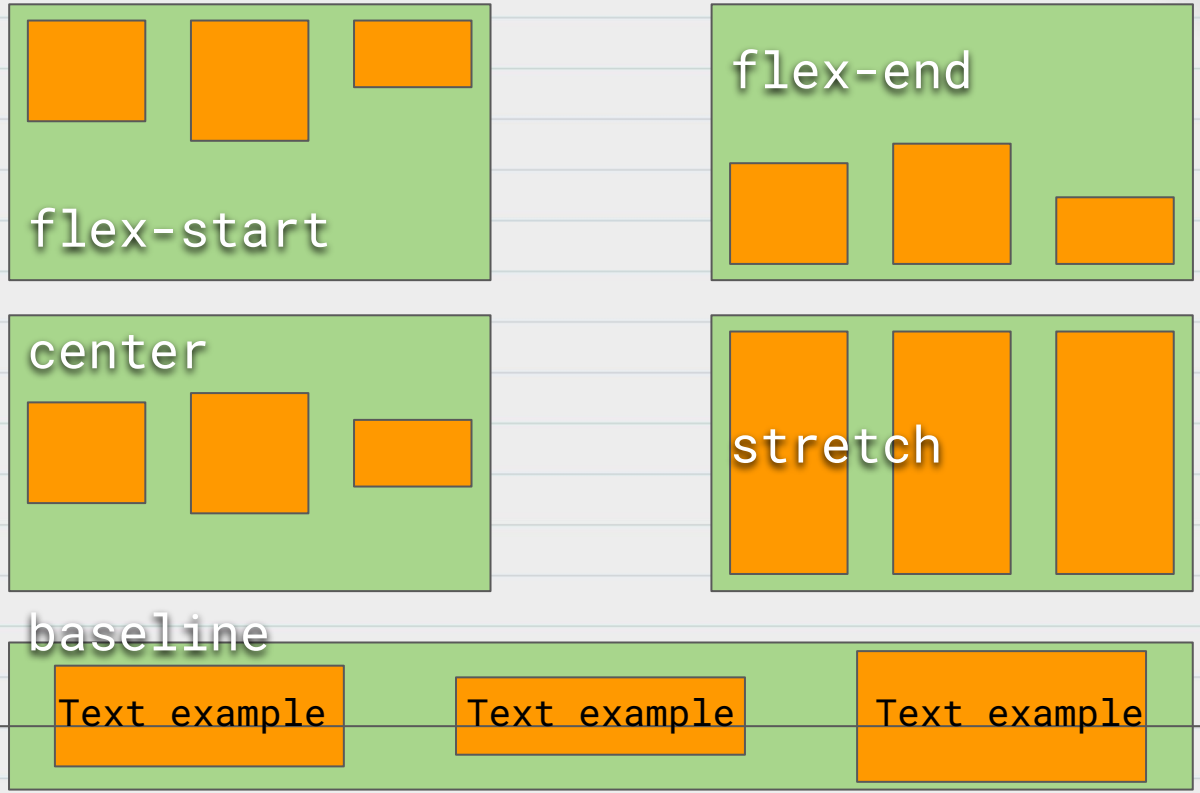
Flex: justify-content

7-3

align-items

次軸排列方式

Flex: align-items

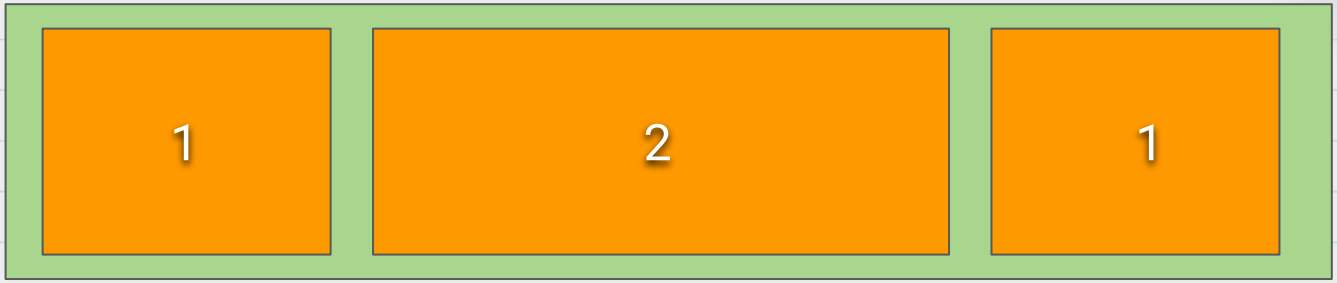


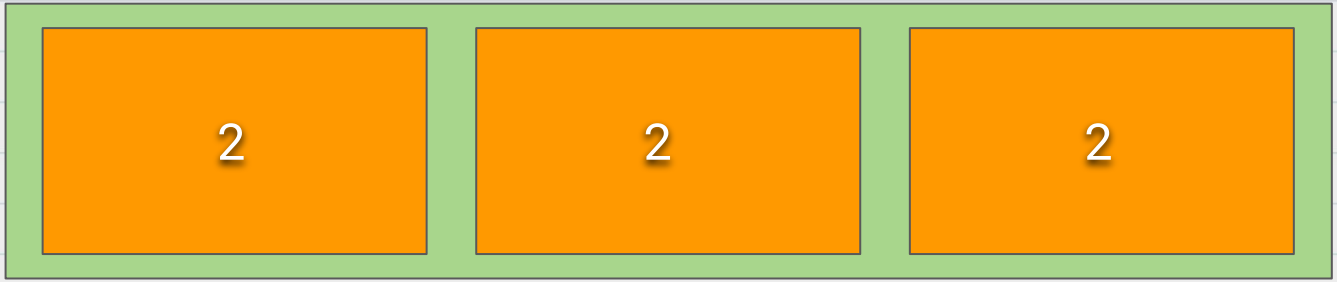
7-4

flex-grow

自動放寬

Flex: flex-grow





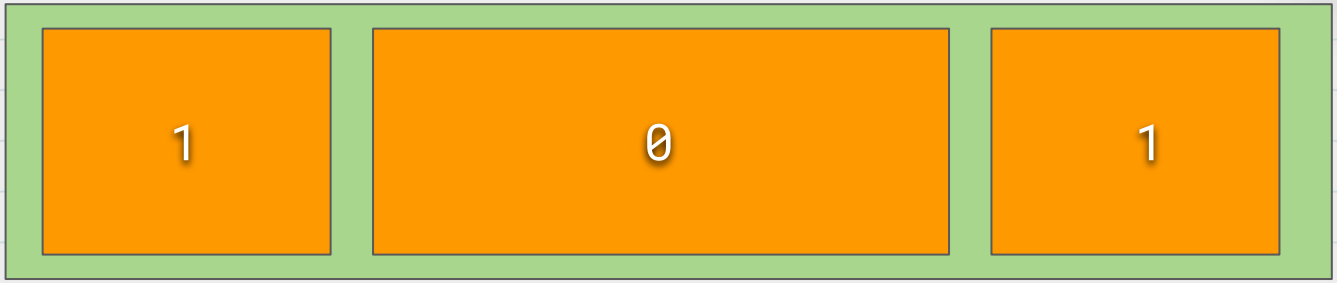
剩餘空間會按照權重分配寬度，預設值是 **0**，也就是不參與分配的意思。

7-5

flex-shrink

自動縮小

Flex: flex-shrink



空間會按照權重分配收縮程度，預設值是 **1**。

7-6

flex-basis

主軸計算基準

Flex: flex-basis

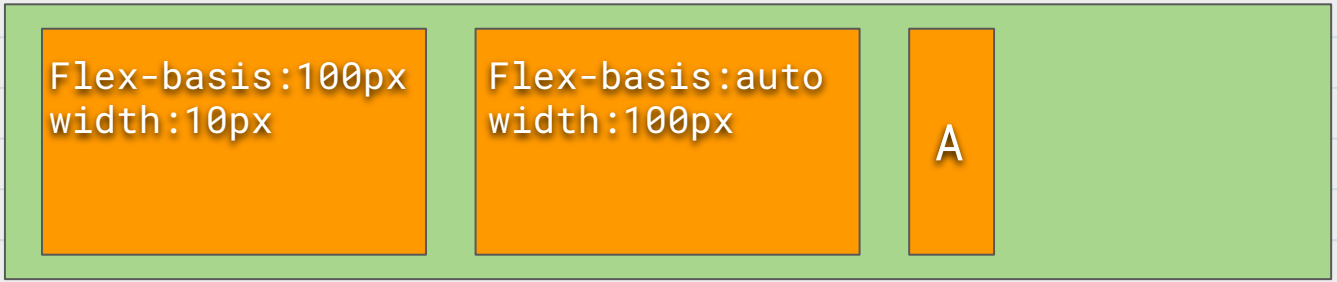
主軸為橫向的時候：

Flex-basis 其實就是決定 items 在放進 flex 容器時的寬度。

如果沒給 flex-basis，則預設值為 auto，寬度取決於 width 的設定。

如果沒給 width 設定，則寬度由內容決定。

注意：flex-basis 值會受到 max-width / min-width 影響。



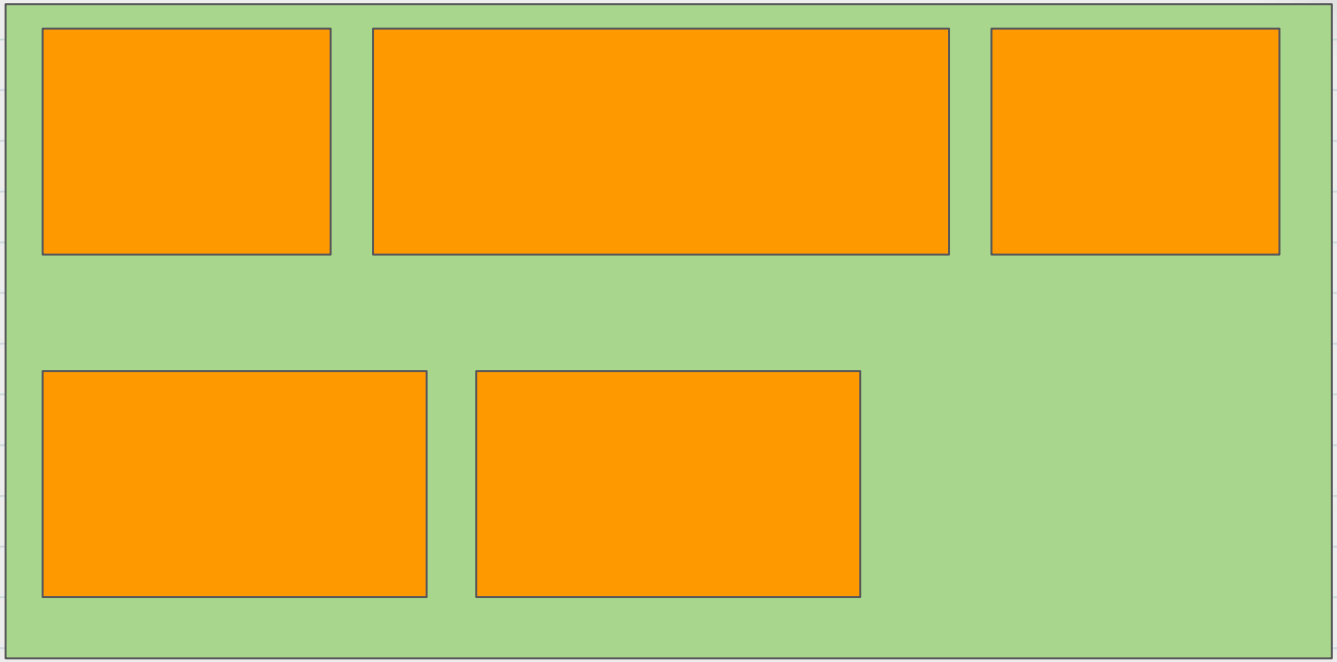
結論: max-width/min-width > flex-basis > width > content

7-7

flex-wrap

换行

Flex: flex-wrap



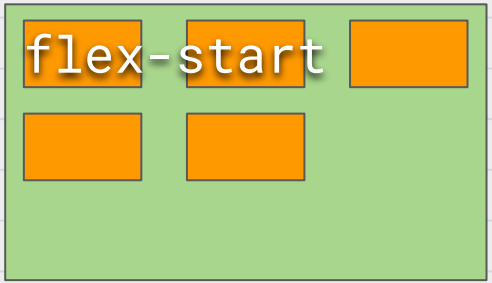
決定超過容器寬度的時候是否要換行。

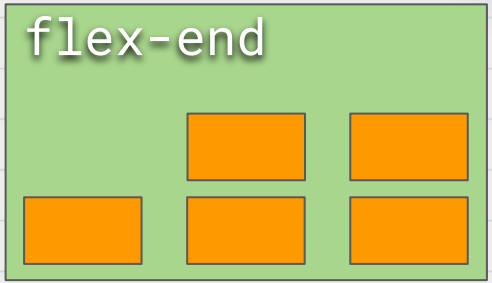
7-8

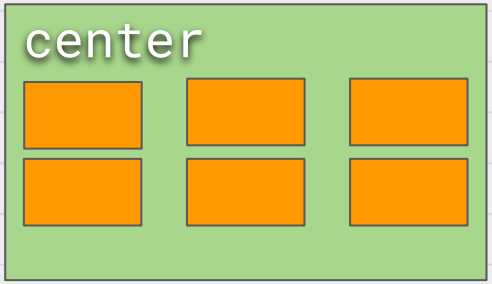
align-content

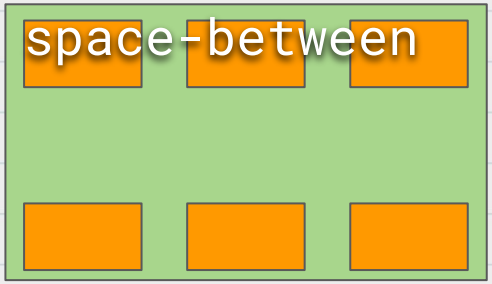
區塊排列

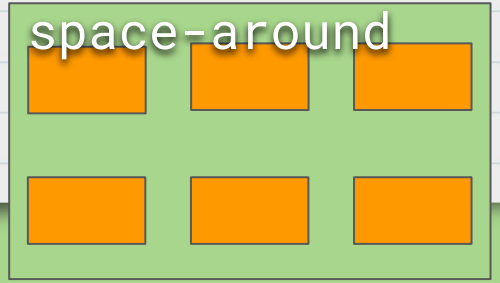
Flex: align-content

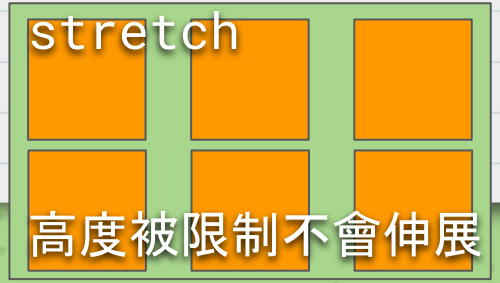










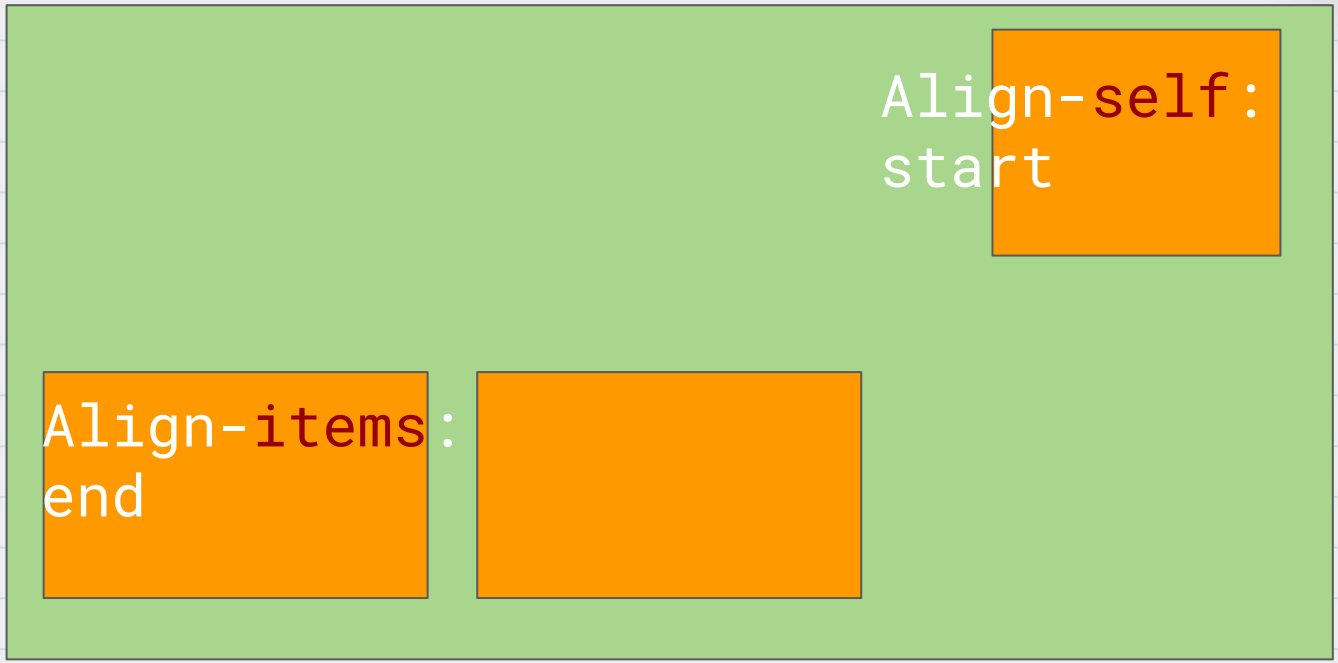


7-9

align-self

自己的次軸排列

Flex: align-self



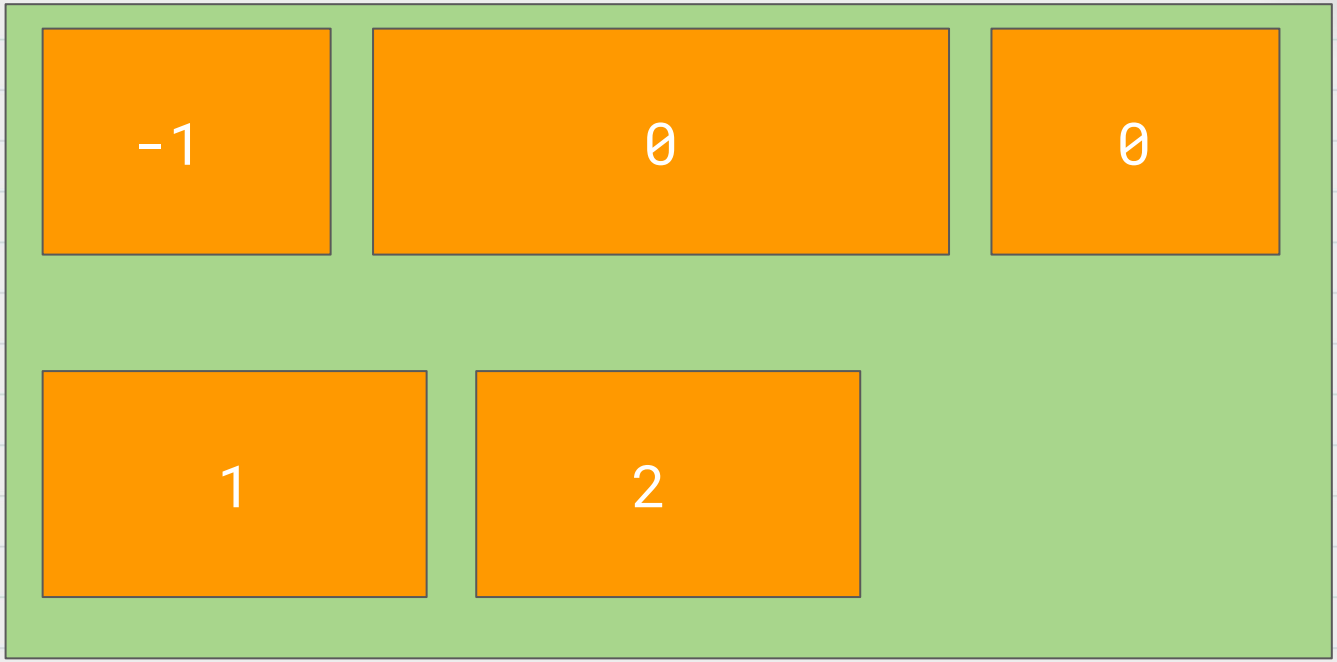
可針對單一個 item 自身改變其 align 的對齊屬性。

7-10

order

修改顺序

Flex: order



Order 會將 items 由小到大排列，預設值為 0。

嘗試用 Flex 切版

所有排列屬性都學完,

現在試著用 Flex 來挑戰排版吧

https://alameda-demo.squarespace.com/

8

何謂響應式網頁設計

RWD 是什麼可以吃嗎？

RWD VS AWD

RWD:
Responsive Web Design 響應式網頁設計

AWD:
Adaptive Web Design 自適應式網頁設計

	AWD	RWD
CSS 差異	桌機、手機...各裝置皆寫一個	全寫在一起
開發時間	可多人同時開發	不同版面樣式容易互相影響
維護難易度	較簡單	較難
SEO	有可能略微降低	略好

8-1

RWD

@media query

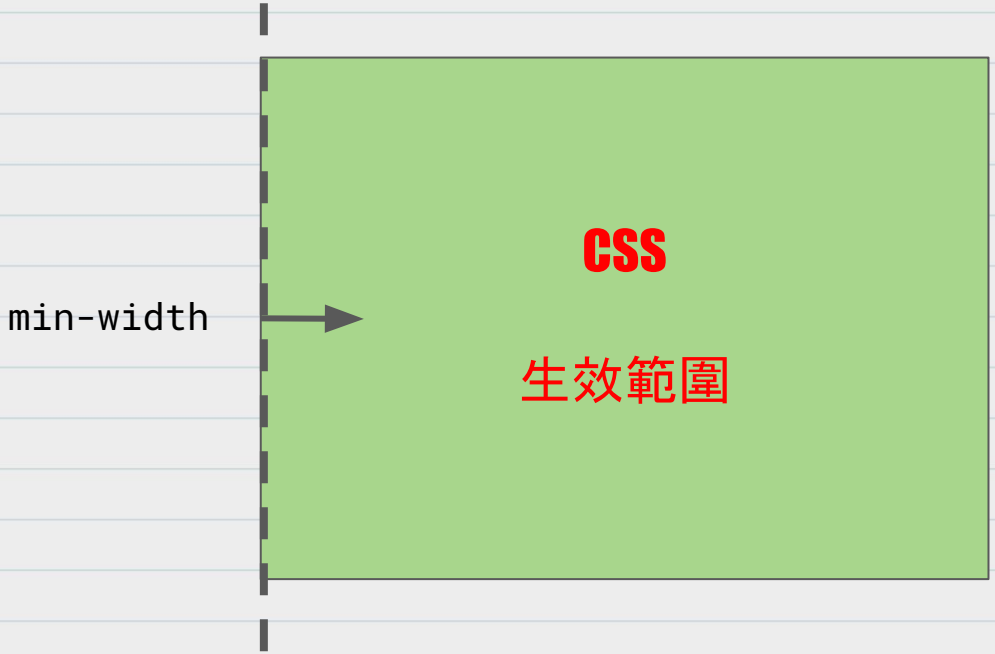
HTML5 預設的這一行很重要

```
<meta name="viewport"
content="width=device-width,
initial-scale=1.0">
```

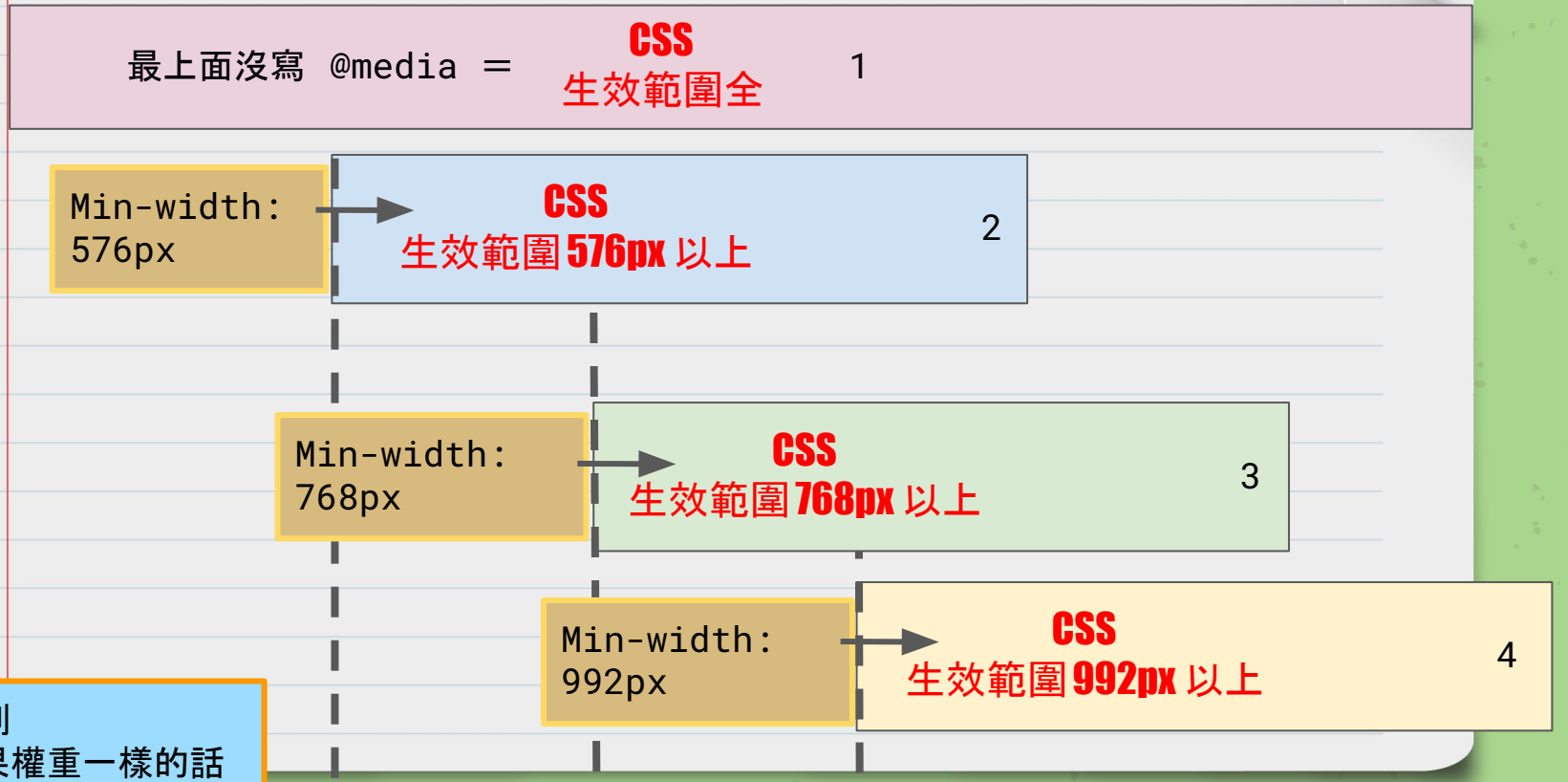
注意：沒有這一行的話，或不小心改到格式錯誤，RWD 會有問題。

@media 用來設定 css 生效範圍

手機優先(先從小螢幕開始寫)



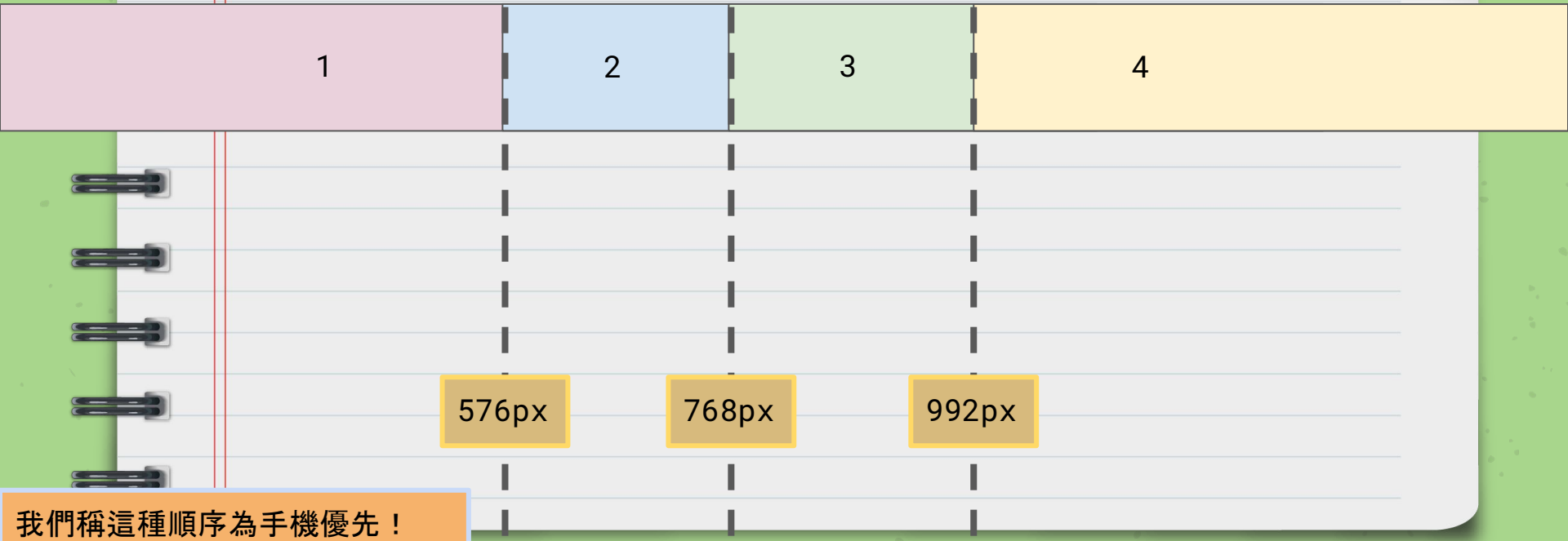
按照 CSS 寫的順序，後面蓋掉前面



依據 CSS 原則
重複的屬性如果權重一樣的話
會有「後蓋前」的特性。

@media 用來設定 css 生效範圍

依照 CSS 順序給予數字編號的話：



我們稱這種順序為手機優先！
(Mobile First)

如果想寫桌機優先...

範圍反過來做即可，但實務上不建議採用此方法