CSS 基礎及應用

語法介紹、屬性說明與應用

語法介紹

```
選擇器 {
  屬性: 設定值;
}
```

CSS 選擇器（Selector）

優先級： !important > style > ID > Class > 標籤 > *

標籤選擇器（ a ）

直接將指定元素標籤名稱當做選擇器

ID 選擇器（ #a ）

選擇使用id屬性命名的元素

Class 選擇器（ .a ）

選擇使用class屬性命名的元素

複合選擇器（ .a.b ）

設定多個class時，選擇同時擁有指定class的元素

群組選擇器（ a, b ）

利用「逗號」同時選擇多個指定的元素

後代選擇器（ a b ）

選擇所有後代元素，依序為「父元素→子元素→孫元素」

子選擇器（ a > b ）

只選擇子元素，子元素之子元素（孫元素）並不受影響

相鄰選擇器（ a + b ）

選擇有相同父元素並緊接在後的「第一個」兄弟元素

間接選擇器（ a ~ b ）

選擇有相同父元素並緊接在後的所有兄弟元素

通用選擇器（ * ）

將樣式套用於全部元素標籤中

CSS 表單常用選擇器

:checked

選擇被選取的單選或多選元素

:focus

選擇處於 focus 狀態的元素

::placeholder,
::-webkit-input-placeholder

選擇輸入框的提示文字

:enabled

選擇不含 disabled 屬性的元素

:disabled

選擇含有 disabled 屬性的元素

:-webkit-autofill,
:-webkit-autofill:hover,
:-webkit-autofill:focus,
:-webkit-autofill:active

box-shadow: inset 0 0 0 1000px rgba(255, 255, 255, 0)
transition: background-color 5000s 0s ease-in-out
用來消除表單 autocomplete 預設填入的背景

CSS 偽類選擇器

受相鄰不同標籤（如：div, p）及不同元素（如：#id123, .class321）影響

:first-child

選擇第一個相鄰元素

:last-child

選擇最後一個相鄰元素

:nth-child(odd)

選擇基數的相鄰元素

:nth-child(even)

選擇偶數的相鄰元素

:nth-child(n)

選擇第n個相鄰元素，n為數字輸入，從 1 開始

:nth-child(an)

選擇每a個相鄰元素的第a個，a為數字輸入

:nth-child(an+b)

選擇每a個相鄰元素的第b個，a與b為數字輸入

:nth-child(an-b)

選擇每a個相鄰元素的第a-b個，a與b為數字輸入

CSS 偽類選擇器

不受相鄰不同標籤影響，但一樣會受相鄰相同標籤、不同元素影響

:first-of-type

選擇第一個相鄰元素

:last-of-type

選擇最後一個相鄰元素

:nth-of-type(odd)

選擇基數的相鄰元素

:nth-of-type(even)

選擇偶數的相鄰元素

:nth-of-type(n)

選擇第n個相鄰元素，n為數字輸入，從 1 開始

:nth-of-type(an)

選擇每a個相鄰元素的第a個，a為數字輸入

:nth-of-type(an+b)

選擇每a個相鄰元素的第b個，a與b為數字輸入

:nth-of-type(an-b)

選擇每a個相鄰元素的第a-b個，a與b為數字輸入

CSS 屬性選擇器［屬性="值"］

[class="u-title"]

class 只能是 u-title 值
符合：class="u-title"
不符：class="u-title u-color-red"

[class^="u-title"]

class 必須以 u-title 字眼開頭
符合：class="u-title-lg u-color-red"
不符：class="u-color-red u-title-lg"

[class~="u-title"]

class 必須有 u-title 值
符合：class="u-title u-color-red"
不符：class="u-title-lg"

[class$="u-title"]

class 必須以 u-title 字眼結尾
符合：class="u-color-red u-title"
不符：class="u-color-red u-title-lg"

[class*="u-title"]

class 只要有 u-title 字眼就符合
符合：class="u-title-lg u-color-red"
符合：class="u-color-red u-title-lg"

[class|="u-title"]

class 只能包含 u-title 或 u-title 以 - 分隔的值
符合：class="u-title u-title-lg"
不符：class="u-color-red u-title-lg"

CSS 偽元素（Pseudo）

偽元素：設定後會自動於指定 HTML 元素內生成的區塊

於選擇的元素內最前面生成（ ::before ）　　於選擇的元素內最後面生成（ ::after ）

偽元素必要屬性為 content，可填入空值、字串或 attr(HTML屬性名稱)
如：content: ''　|　content: '$'　|　content: attr(data-label)

常見用法為 content 帶空值，並設定 display 當一般區塊應用：

```
.class-name::before {
  content: '';
  display: block;
  position: absolute;
  left: 20px;
  top: -40px;
  width: 80px;
  height: 80px;
  background-color: #eee;
}
```

樣式繼承與還原

說明 inherit, initial, unset, revert 差異

屬性的繼承與還原

屬性繼承： inherit

每一個 CSS 屬性皆有默認可繼承或不可繼承特性，大部分文字類樣式是可以繼承外層樣式，
如：font-family, letter-spacing, color 等等；
如果想要將瀏覽器預設樣式改為繼承外層樣式，可加上 inherit 值。
常用手法：

```
a {
  color: inherit;
}
```

屬性預設值： initial

每一個 CSS 屬性皆有預設樣式（非瀏覽器預設）

屬性重置： unset

如該 CSS 屬性為可繼承屬性，將套用 inherit，否則套用 initial 值

屬性還原： revert

將該 CSS 屬性還原為瀏覽器預設樣式值

是否可繼承及預設值可參考下列連結的 Properties：
https://developer.mozilla.org/zh-CN/docs/Web/CSS

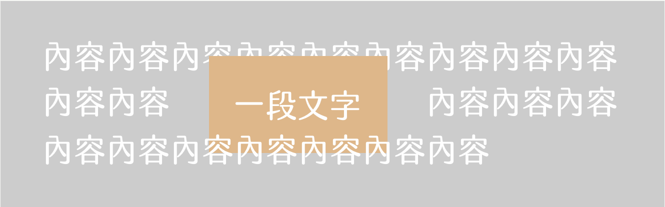
基本 Display 差異

inline, inline-block, block 差異說明

Display 特性說明

display: inline

在行列裡
設定上下 margin 或 padding 不影響
行高（不允許破壞行列）
通常是文字、圖片標籤預設的 display

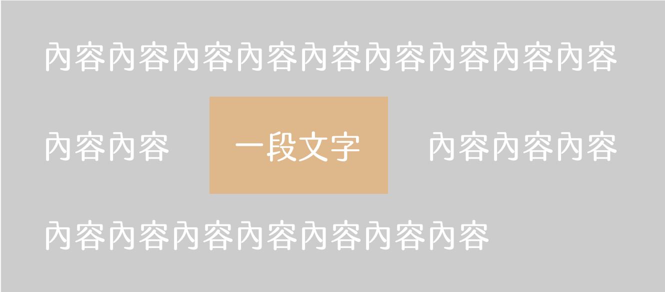


display: none

不顯示區塊
整個區塊將從畫面移除，且不佔位
動畫過渡也將失效

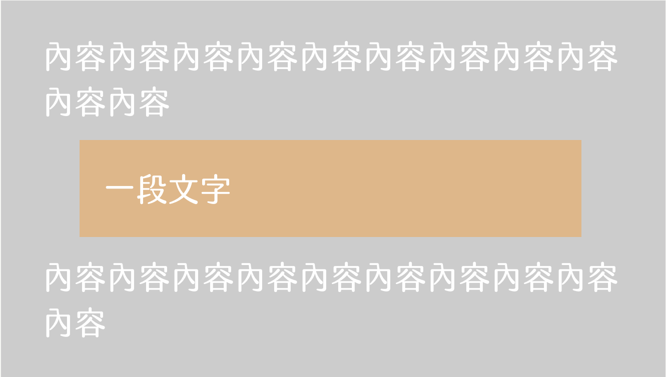
display: inline-block

在行列裡的區塊
設定上下 margin 或 padding 將影響
行距；且跟文字很像，太長會自動換行
也常用在讓區塊寬度符合內容寬度



display: block

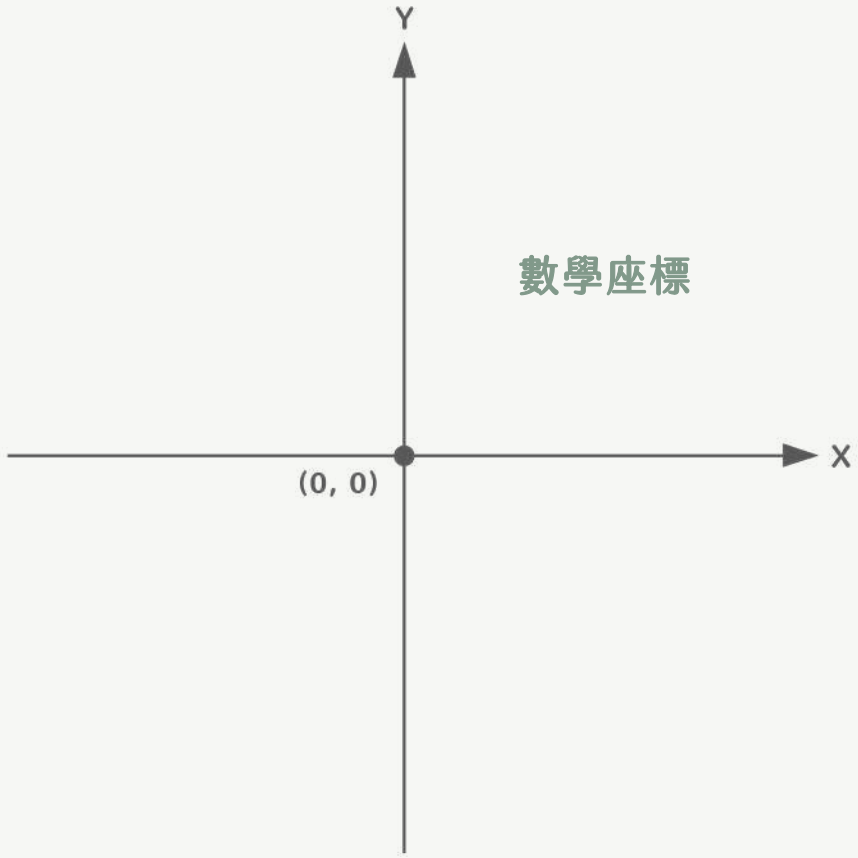
區塊
跳脫行列獨立出來，且預設寬 100%
也是大部分標籤預設的 display



內外距、邊框

分辨內外距及其設定值、邊框說明

網頁起始座標



數學座標

網頁座標

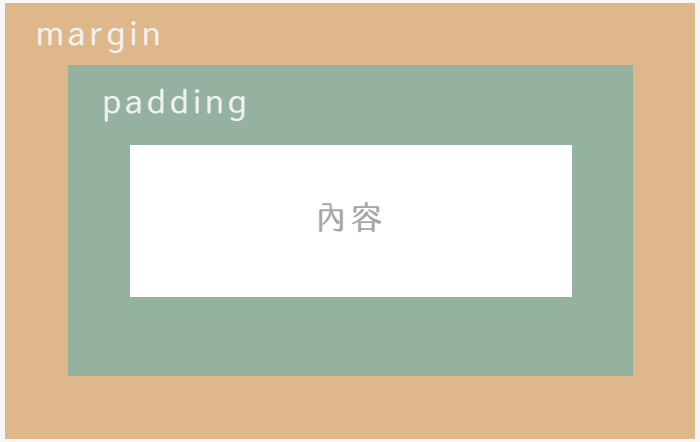
CSS 內外距、邊框

內距（padding）

於元素內容周圍加上指定大小的空間
padding-left, padding-right
padding-top, padding-bottom

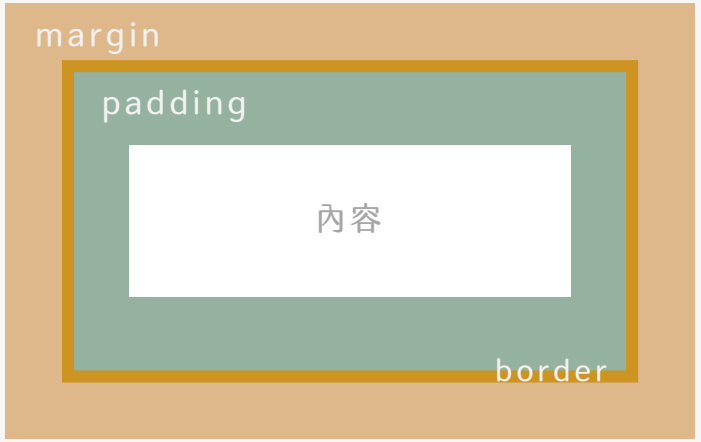
外距（margin）

指定元素與其他元素的距離
margin-left, margin-right
margin-top, margin-bottom



邊框（border）

指定元素的外圍邊框
border-left, border-right
border-top, border-bottom
四邊 -> border: 1px solid #000

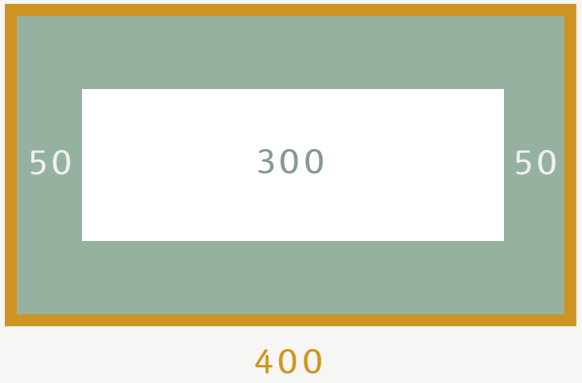



```
padding: 10px          padding: 10px 20px      padding: 8px 20px 10px      padding: 8px 0 10px 8px
四邊皆套用              上下 / 左右              上 / 左右 / 下              上 / 右 / 下 / 左
```

CSS 區塊內距計算（box-sizing）

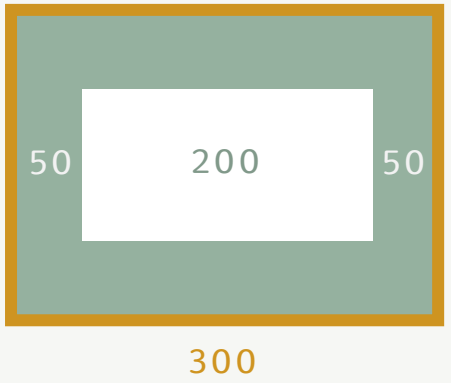
寬度不包含內距（content-box）

```
box-sizing: content-box;
width: 300px;
padding: 50px;
```



寬度包含內距（border-box）

```
box-sizing: border-box;
width: 300px;
padding: 50px;
```



專案實作

維基百科加入 CSS 樣式

定位、Display Flex

各種定位使用、Display Flex 版面編排說明

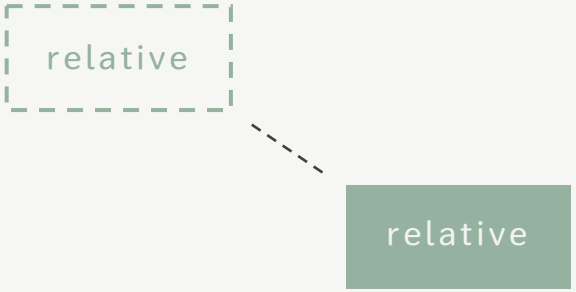
CSS 定位（position）

靜態定位（static）

預設（無法設定位置）

相對定位（relative）

相對於自己



絕對定位（absolute）

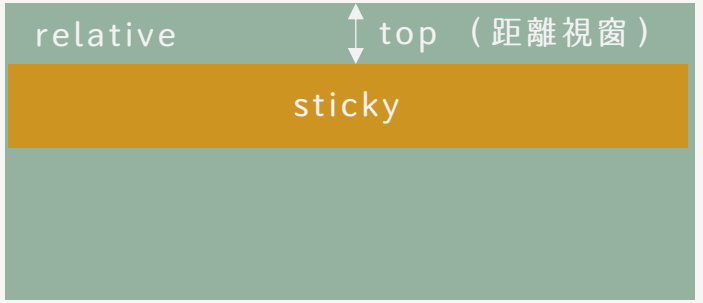
絕對於設有定位的父層；
必須設 left 或 right 及 top 或 bottom

固定定位（fixed）

固定於可視區域（父層不能設 transform）
必須設 left 或 right 及 top 或 bottom

黏性定位（sticky）

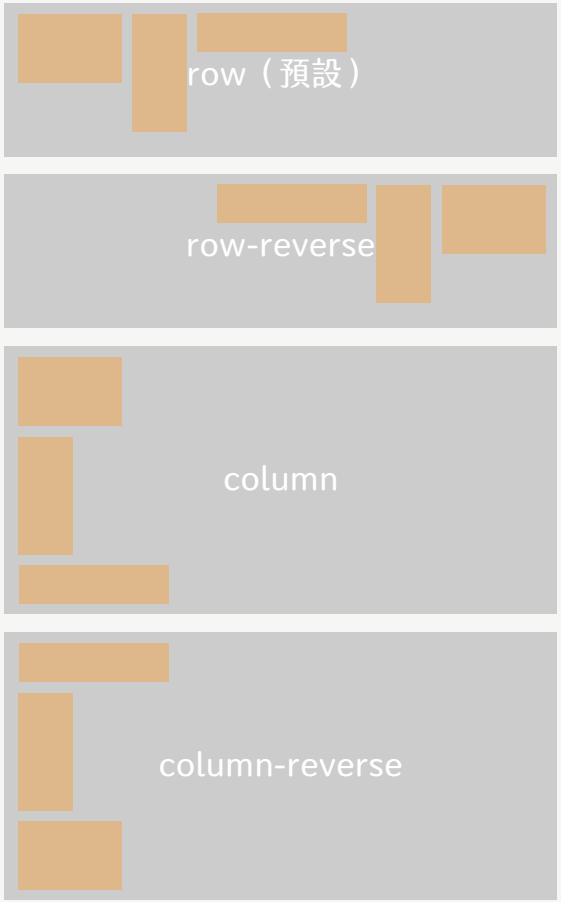
當元素 top 或 bottom 碰觸視窗上或下邊
且於父層（不能設 overflow）區塊範圍內



Display Flex 排版（設在父層，決定子層如何排列）

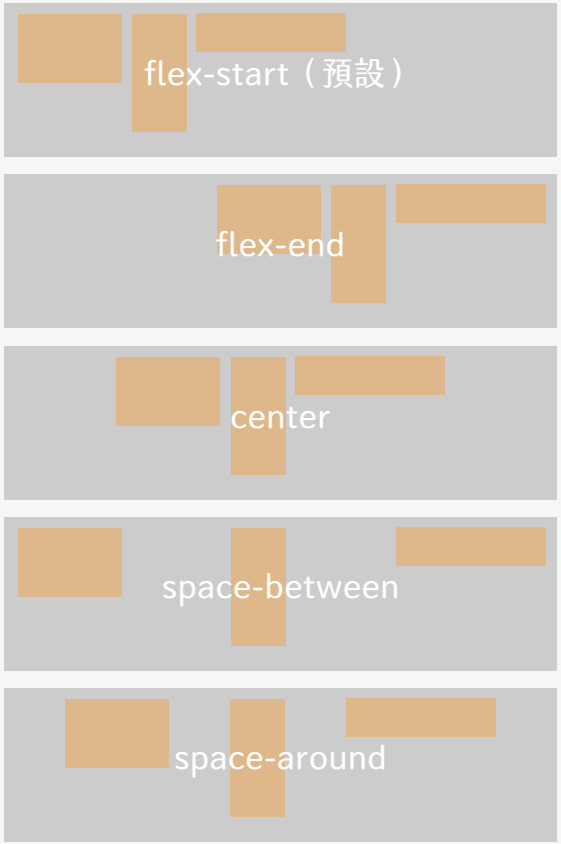
flex-direction:

改變軸線方向



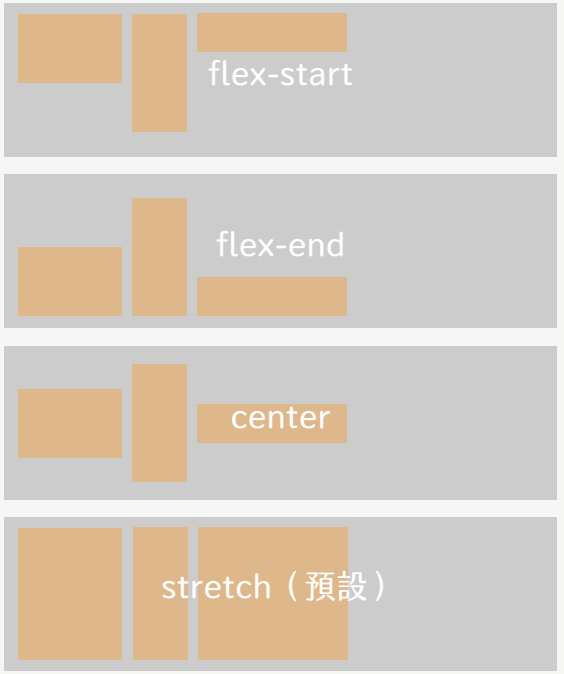
justify-content:

主軸對齊



align-items:

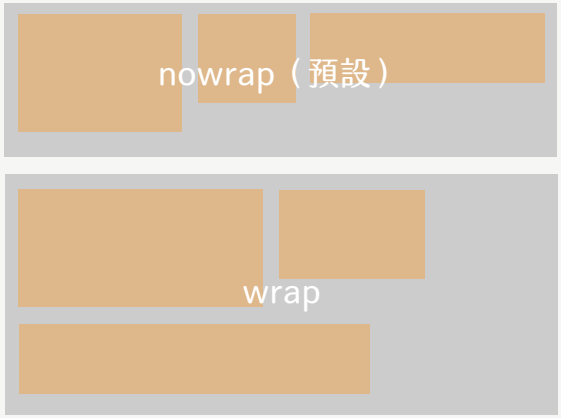
副軸對齊



Display Flex 排版（設在父層，決定子層如何排列）

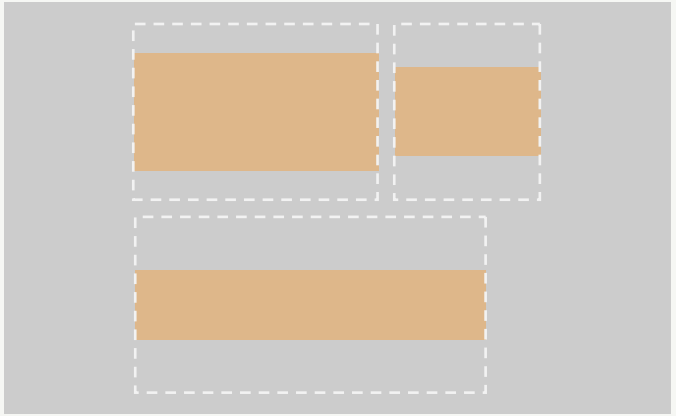
flex-wrap:

子元素超出範圍時自動換行



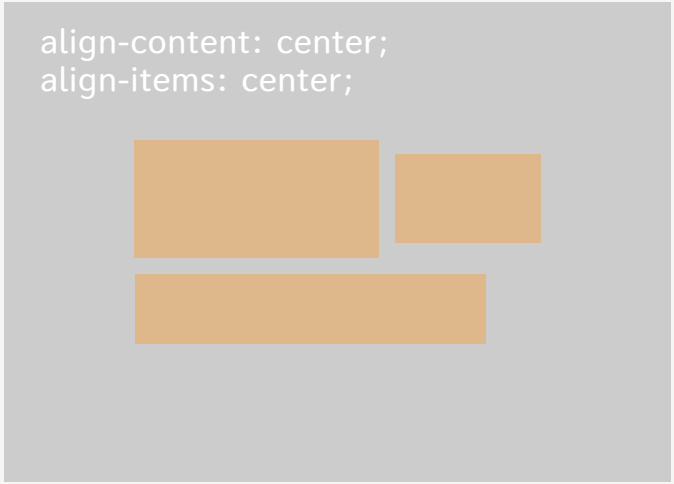
flex-wrap + align-items

設定 wrap 時會因為 align-items 特性
讓元素平均分配多出來的空間；
設 center 會依照自己的原高度做置中



align-content

使用 wrap 時 align-content 解決了
align-items 空間問題；
且可設定數值與 justify-content 一樣

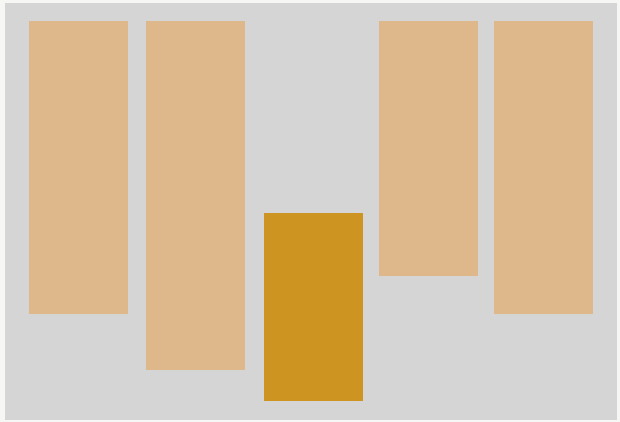


Display Flex 排版（設在子層，個別控制）

align-self:

副軸設置單一元件位置

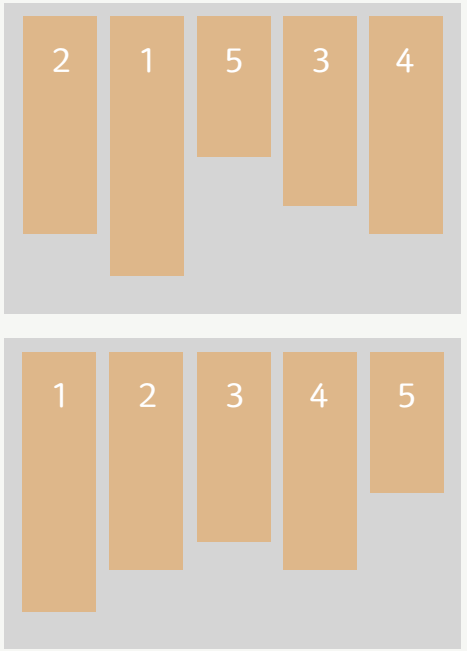
flex-end



order:

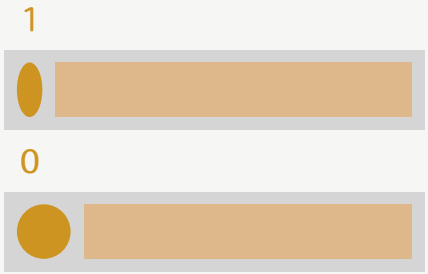
依據數值大小排列，越小越前

輸入數值



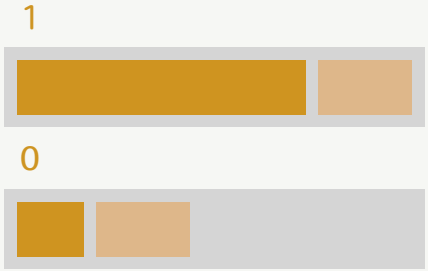
flex-shrink:

元件壓縮比



flex-grow:

元件伸展比



專案實作

製作網頁小卡及表單

Z-index

了解及設定 html 圖層關係

z-index 用來設定元素前後位置（需設 position 定位）

HTML 圖層特性是兄弟層結構上到下，越下面顯示越前面（越靠近我們）

```
<div>
  <div class="block block-1"></div>
  <div class="block block-2"></div>
  <div class="block block-3"></div>
</div>
```

透過 z-index 並設有 position 可以自由控制前後關係（預設 0，數字越大越前面，可以設負數）

```
.block {
  position: relative;
}
.block-1 {
  z-index: 10;
}
```

z-index 父子層關係

```
<div class="block-wrap">
  <div class="block block-1"></div>
</div>
<div class="block-wrap">
  <div class="block block-2"></div>
</div>
<div class="block-wrap">
  <div class="block block-3"></div>
</div>
```

當父層沒有設定 z-index 的情況下，各層級可以相互排序

```
.block-wrap, .block {
  position: relative;
}
.block-wrap:nth-child(3) {
  z-index: 5
}
.block-2 {
  z-index: 10;
}
```

當父層設有 z-index 的情況下，將以父層同級元素為優先排序

```
.block-wrap:nth-child(2) {
  z-index: -1
}
.block-wrap:nth-child(3) {
  z-index: 5
}
.block-2 {
  z-index: 10;
}
```