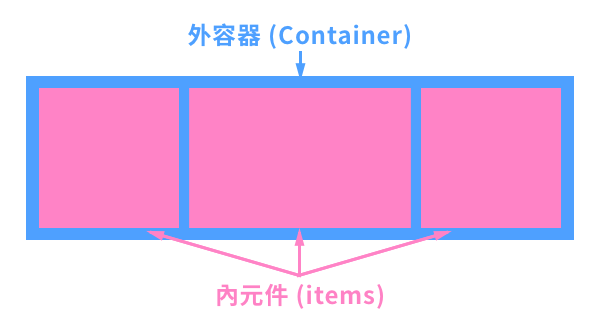
**CSS Flex**

## **Flex 的外容器與內元件**

屬性放錯地方就沒有作用，學習的過程中如果可以先了解正確的擺放位置，可以減少許多除錯的時間；Flex 中分為外容器與內元件，下方這張圖就是解釋這兩個的差異。

Flex 外容器屬性：

* display
* flex-flow
  + flex-direction
  + flex-wrap
* justify-content
* align-items

Flex 內元件屬性：

* flex
  + flex-grow
  + flex-shrink
  + flex-basis
* order
* align-self

## **外容器**

### **display**

### 一開始要宣告為 flex 才能使用，如果沒有宣告為 flex，本篇大部分的屬性都無法作用，不過除了 flex 外，還有一個 inline-flex，作用類似於 inline-block + flex。

### .flex-container {

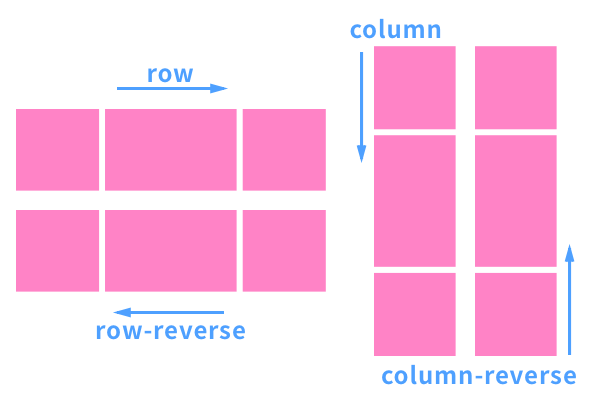
### display: flex | inline-flex;

}

### **flex-direction**

### Flex 決定內元件排序方向的重要屬性，這也直接影響了 flex 的軸線 (主軸及交錯軸)，flex 中的對齊屬性都與此有很大的相關聯，下圖中是預設狀態的軸線方向。

flex-direction 就是來改變上圖中的軸線方向，可以將軸線做水平反轉、轉為垂直、垂直反轉等，下圖是四種軸線排列的示意圖：



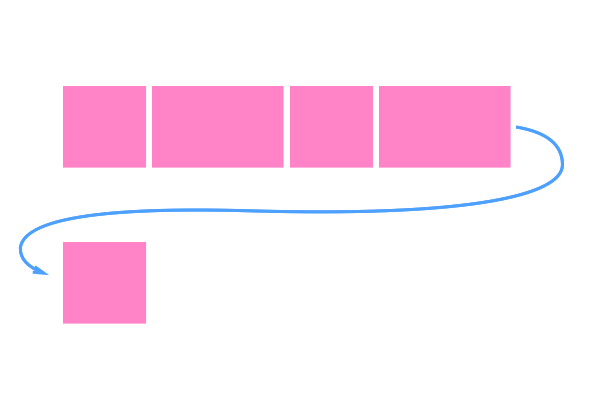
.flex-container {

flex-direction: row | row-reverse | column | column-reverse;

}

Example：[Flex Direction範例](https://codepen.io/Wcc723/pen/yXdVXz)

### **flex-wrap**

超出範圍時是否換行的屬性，分為換行、不換行、換行時反轉。

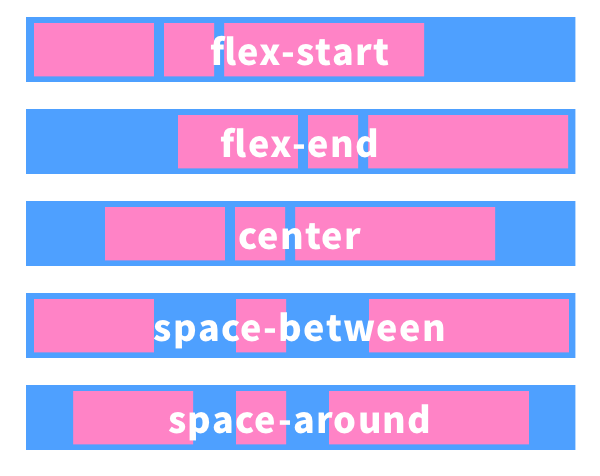
.flex-container {

flex-wrap: nowrap | wrap | wrap-reverse;

}

### **justify-content**

很接近水平對齊的設定，但嚴格說來是稱為主軸對齊，主軸的設定是上方的 flex-direction；所以實際是水平或垂直，要依主軸的方向而定喔。



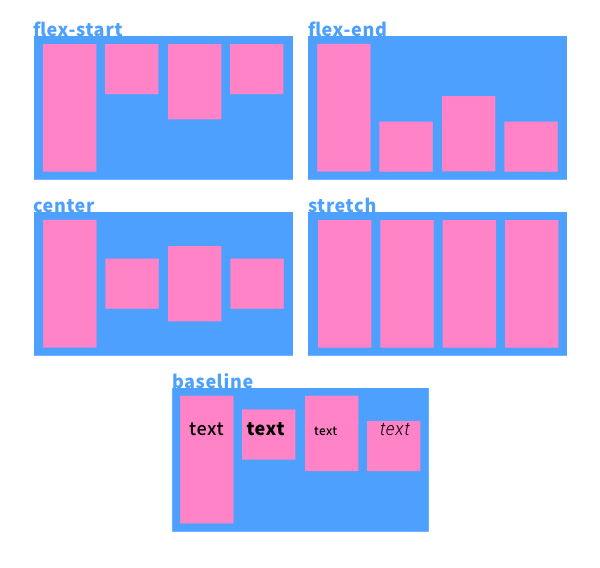
.flex-container {

justify-content: flex-start | flex-end | center | space-between | space-around;

}

* align-items

相對於上一個屬性，這是交錯軸的對齊設定。





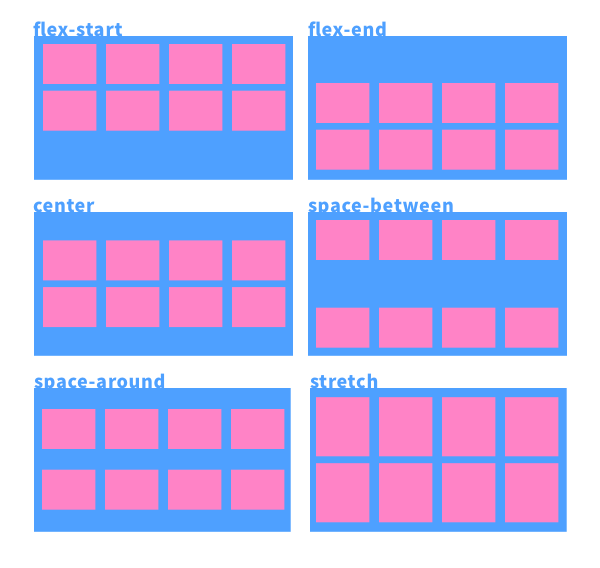
.flex-container {

align-items: flex-start | flex-end | center | baseline | stretch;

}

* align-content

上一個屬性的多行版本，排列方式也有多一些，但注意 stretch 在高度被限制的情況下不會正常伸展。



.flex-container {

align-content: flex-start | flex-end | center | space-between | space-around | stretch;

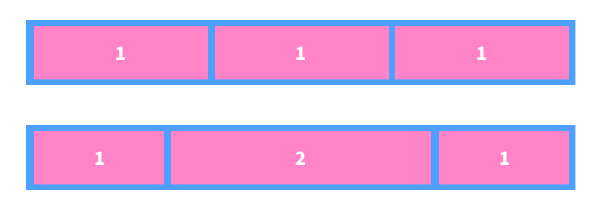
}

## **內元件**

內元件裡面依序包含三個屬性 flex-grow、flex-shrink 和 flex-basis，如果只設定一個則是 flex-grow。

* flex-grow: 元件的伸展性，是一個數值，當空間分配還有剩餘時的當前元件的伸展性，預設值為 0，如果設置為 0 則不會縮放。
* flex-shrink: 元件的收縮性: 元件的伸展性，是一個數值，當空間分配還不足時的當前元件的收縮性，預設值為 1，如果設置為 0 則不會縮放。
* flex-basis: 元件的基準值，可使用不同的單位值(flex item 的初始大小，可設定數值，預設 ‘auto’。)。

以下圖為例，伸展值為 2 的空間會佔更多。



[flexbox layout - JSFiddle - Code Playground](https://jsfiddle.net/camel2243/j657ntf3/10/)

## **align-self**

align-self 可以調整內元件交錯軸的對齊設定(主軸線則不能另外做設定)，且可以個別設定單一元件的值。

## **Order**

這是一個相當特別的屬性，可以重新定義元件的排列順序，順序會依據數值的大小排列。



**POSITION**

position 目前可以使用的值有 5 種，列舉於下

* static (靜態定位)
* relative (相對定位)
* fixed (固定定位)
* absolute (絕對定位)
* sticky (黏貼定位)

Position 的用途

在網頁中 position 的用途是設定「物件定位時所要的參考對像」，預設狀態下物件的位置是依據資料流來做排列，也就是跟隨資料做排列，如果對物件添加了不同的 position 之後，就能改變物件所參考的空間對像，進而改變物件的位置。

### **Static**

「static 靜態定位」是定位中很少用到的，此定位是將其他定位特性取消，回到最原始的狀態，一般來說除非有特殊狀況，不然不會用到，多數的網頁物件預設也都屬於此種定位。

### **Relative**

「relative 相對定位」的效果是將設定的物件，將其參考空間參考自身原始的資料流位置，且此種定位值並不會將物件獨立一層，搭配其它 CSS 屬性，如 top 、 right 、 bottom 、 left 可做到「顯示位置的偏移」。若想看實際的效果

**Absolute**

「absolute 絕對定位」的效果是將設定的物件，將其參考空間設定為「階層離自身最近且具備定位設定的父層空間」，倘若沒有任何一個父層空間俱備定位設定，最終該物件將採用視窗空間座位參考空間，而父層空間只要俱備了以下四種定位類型，就可被絕對定位物件做為參考空間

### **Fixed**

「fixed 固定定位」的效果是將設定的物件，將其參考空間設定為「視窗」，也就是說當我對一個物件設定了 position: fixed; 之後，該物件的參考空間就直接對視窗的範圍了

### **Sticky**

「sticky 黏貼定位」是所有定位中最年輕也最新的一個定位了，這個值非常的棒，它同時結合了 relative 與 fixed 兩大特性

[position: sticky (codepen.io)](https://codepen.io/deathhell/pen/BaprWYp)