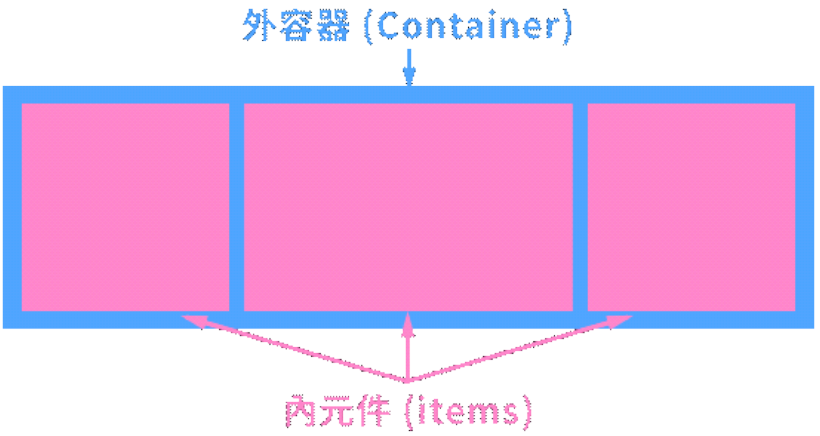
3.3.Flexbox排版

一、Flex 的外容器與內元件

屬性放錯地方就沒有作用，學習的過程中先了解正確的擺放位置，就可以減少除錯的時間；Flex 中分為外容器與內元件，如下圖。



- Flex 外容器屬性：
  - display：flex, inline-flex 必備屬性。
  - flex-flow
    - flex-direction：決定 flex 軸線。
    - flex-wrap：決定換行的屬性。
  - justify-content：主要軸線的對齊。
  - align-items：交錯軸線的對齊。

- Flex 內元件屬性：
  - flex：
    - flex-grow：伸展比（其數值與其它物件可分配伸展比有關）。
    - flex-shrink：收縮比。
    - flex-basis：絕對值。
  - order：排序。
  - align-self：單一個物件的交錯軸對齊。

（一）display

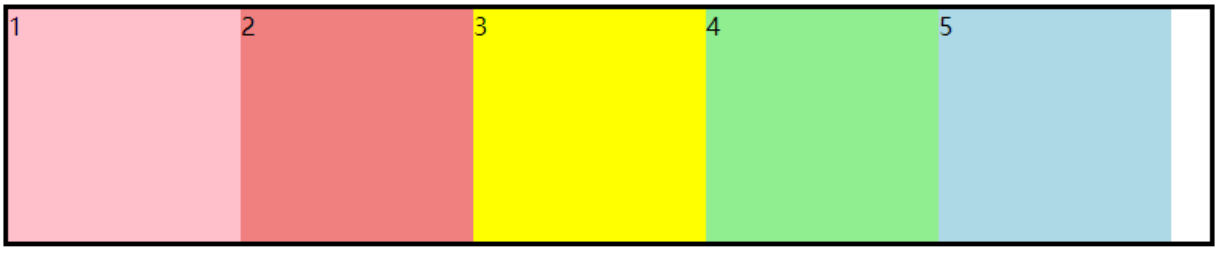
一開始要宣告為 flex 才能使用，如果沒有宣告為 flex，大部分的屬性都無法作用，不過除了 flex 外，還有一個 inline-flex，作用類似於 inline-block + flex。

範例：

Ex:

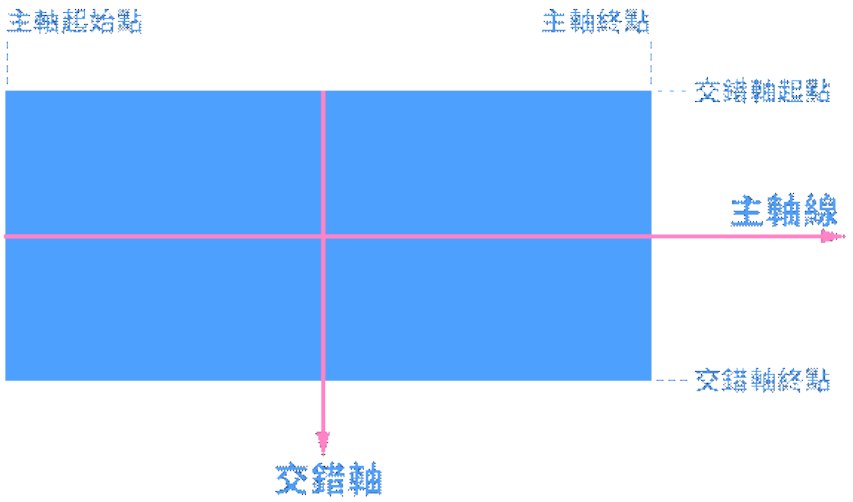
```html
<div class="container">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
    <div class="box box4">4</div>
    <div class="box box5">5</div>
</div>
```

```css
.container{
  border: 3px solid black;
  display: flex;
}
.box {
  width: 150px;
  height: 150px;
}
.box1 {
  background-color: pink;
}
.box2 {
  background-color: lightcoral;
}
.box3 {
  background-color: yellow;
}
.box4 {
  background-color: lightgreen;
}
.box5 {
  background-color: lightblue;
}
```

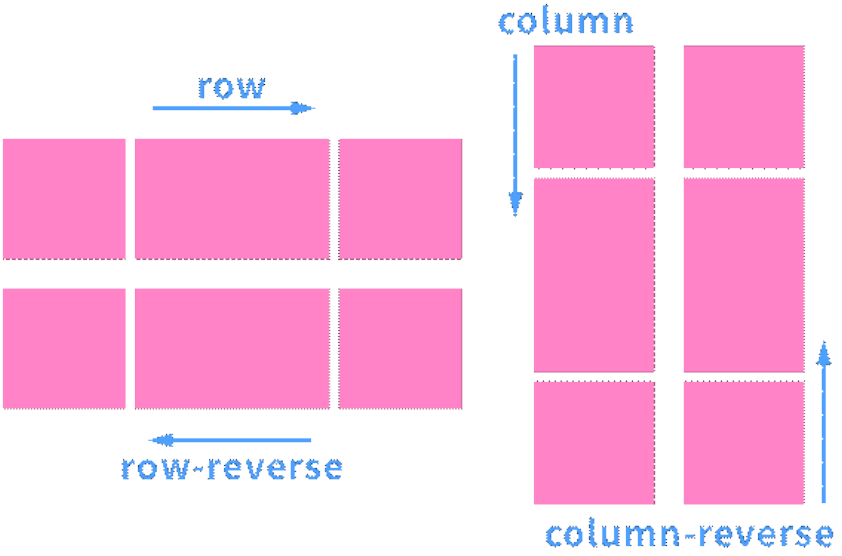


(二) flex-direction

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。



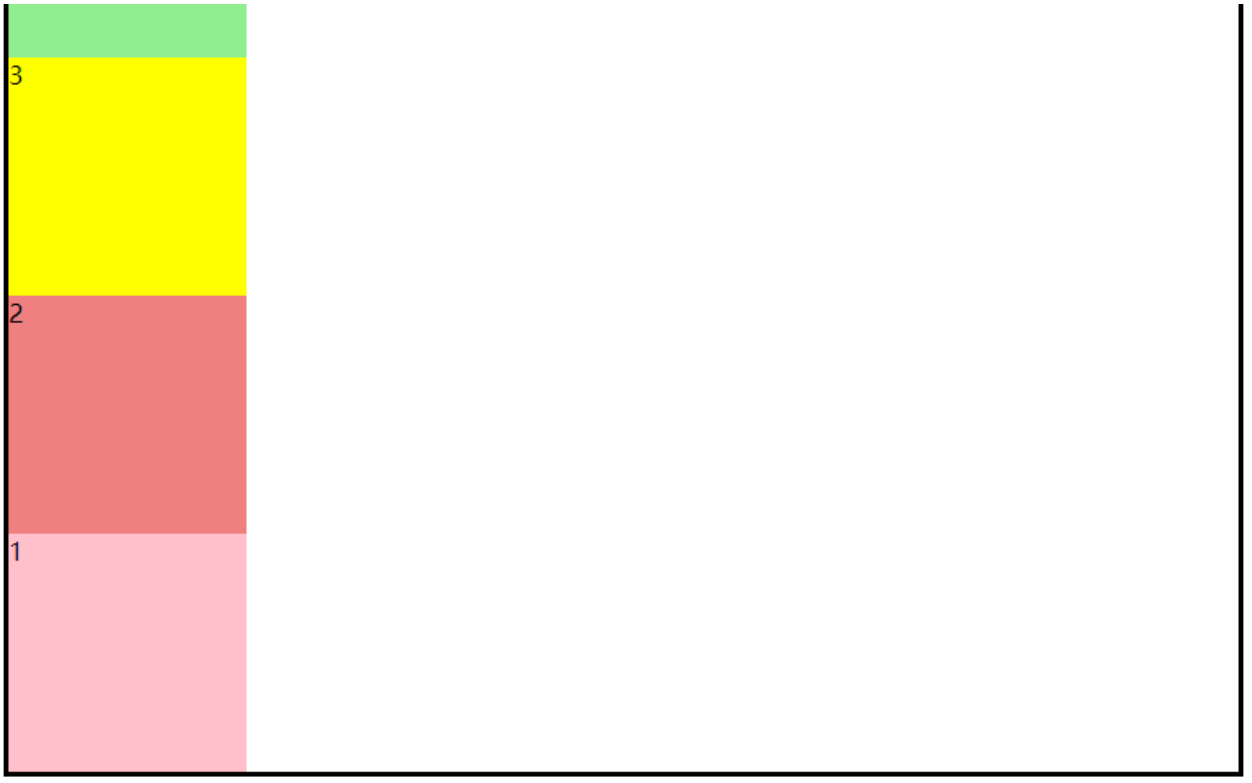
flex-direction 就是來改變上圖中的軸線方向，可以將軸線做水平反轉、轉為垂直、垂直反轉等，下圖是四種軸線排列的示意圖：



範例：

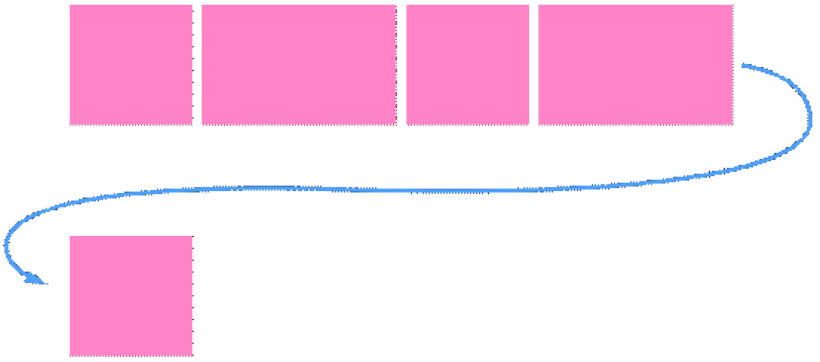
Ex:

```
flex-direction: column-reverse;
```



(三) flex-wrap

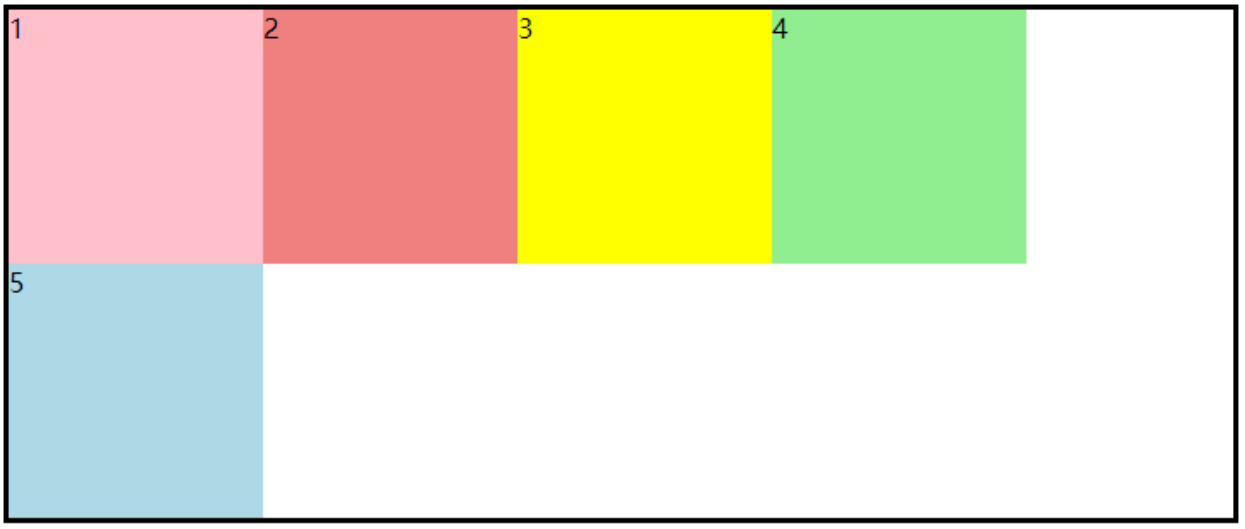
超出範圍時是否換行的屬性，分為 wrap（換行）、no-wrap（不換行）、wrap-reverse（換行時反轉）。



範例：

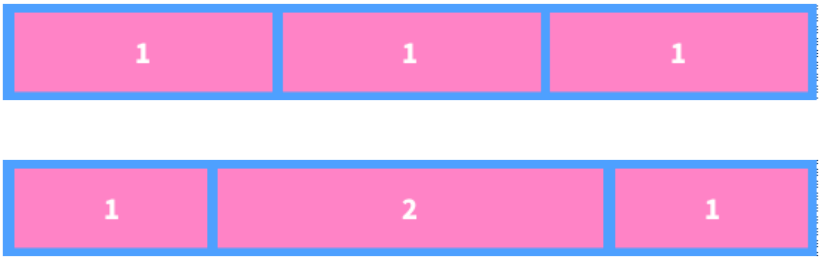
Ex:

```
flex-wrap: wrap;
```



(四) flex-grow

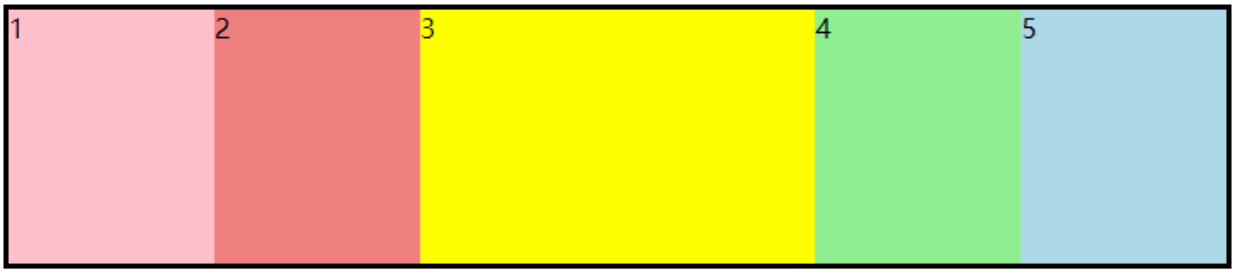
元件的伸展性，是一個數值，當空間分配還有剩餘時的當前元件的伸展性，預設值為 0，如果設置為 0 則不會縮放。



範例：

Ex:

```css
.box {
  /* width: 150px; */
  height: 150px;
}
.box1 {
  flex-grow: 1;
  background-color: pink;
}
.box2 {
  flex-grow: 1;
  background-color: lightcoral;
}
.box3 {
  flex-grow: 2;
  background-color: yellow;
}
.box4 {
  flex-grow: 1;
  background-color: lightgreen;
}
.box5 {
  flex-grow: 1;
  background-color: lightblue;
}
```



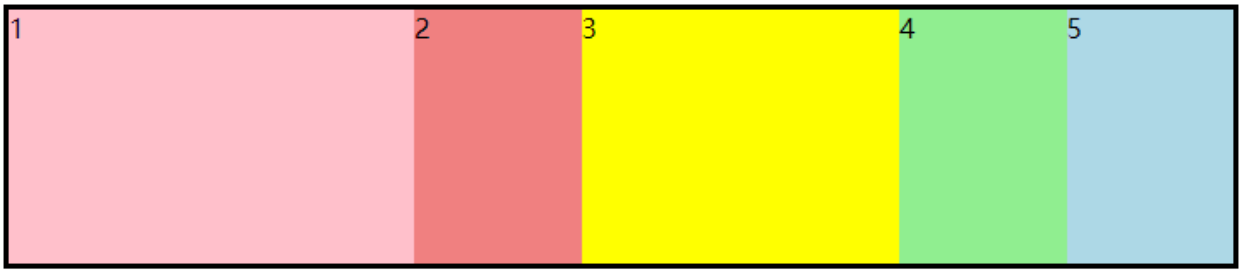
(五) flex-basis

元件的基準值，可使用不同的單位值。

範例：

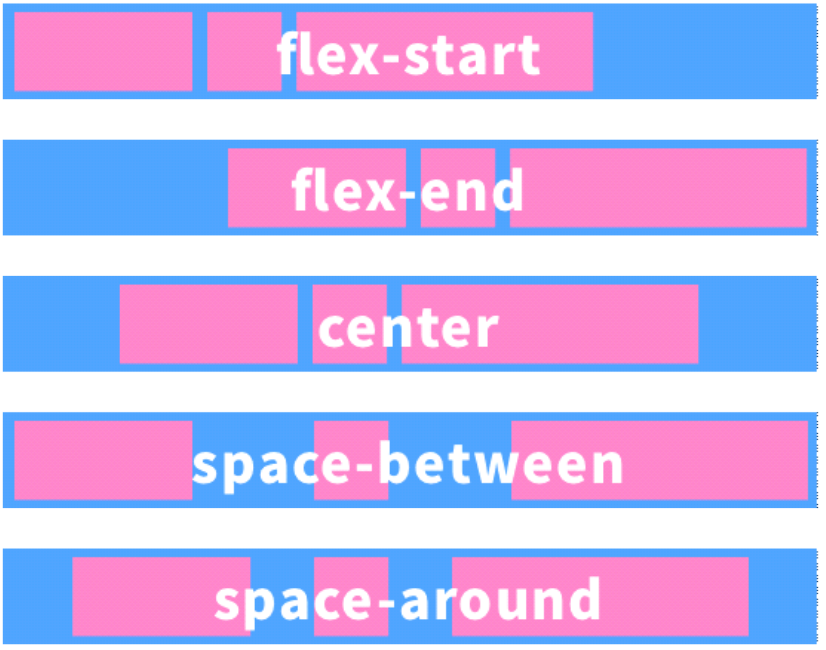
Ex:

```css
.box1 {
  flex-grow: 1;
  flex-basis: 150px;
```



(六) justify-content

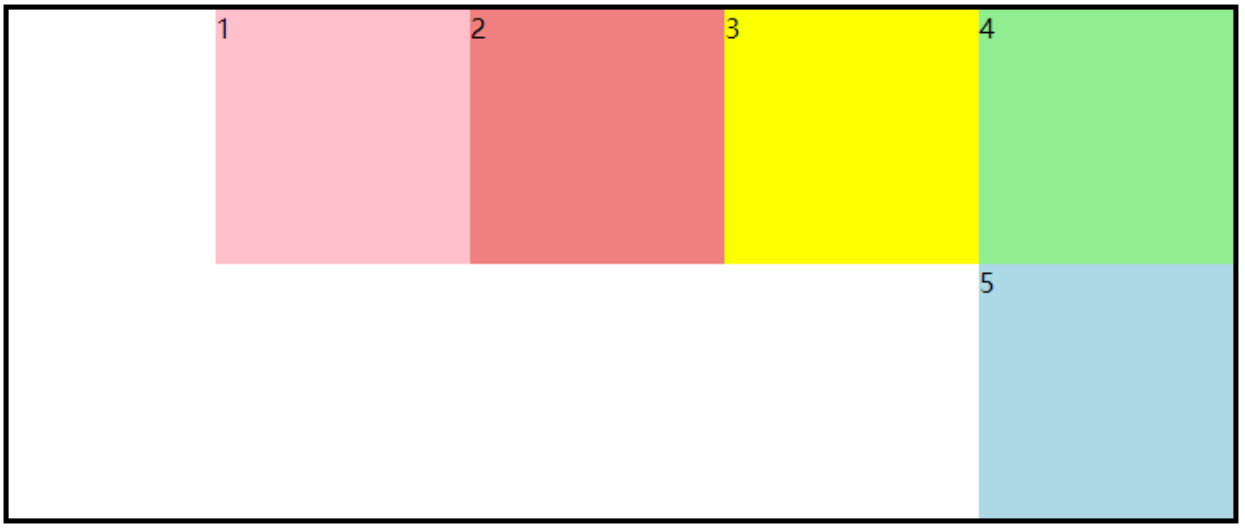
這個屬性很接近水平對齊的設定，但嚴格說來是稱為主軸對齊，主軸的設定是上方的 flex-direction；所以實際是水平或垂直，要依主軸的方向而定。



範例：

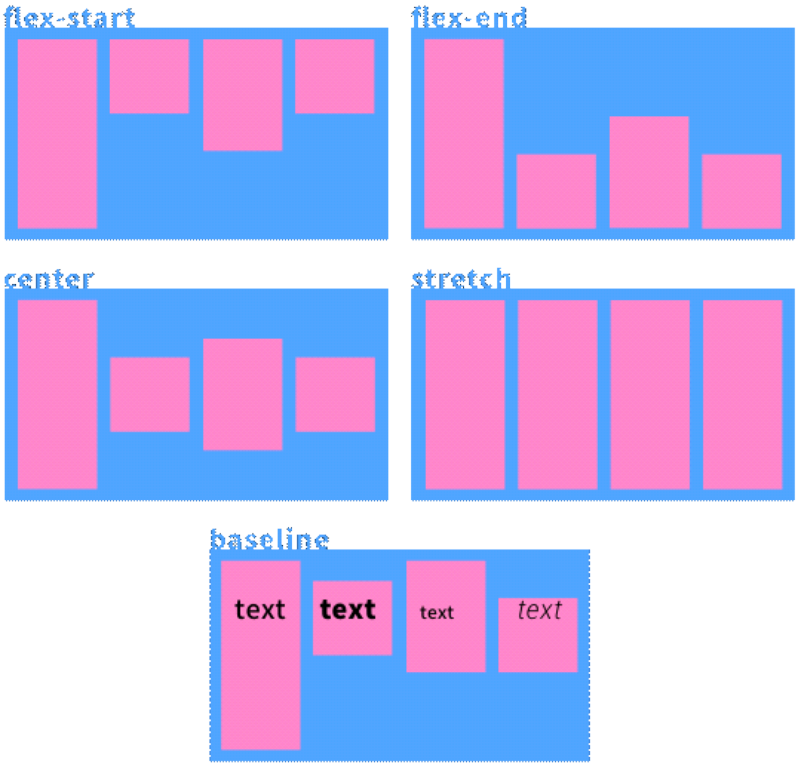
Ex:

```
justify-content: flex-end;
```



(七) align-items

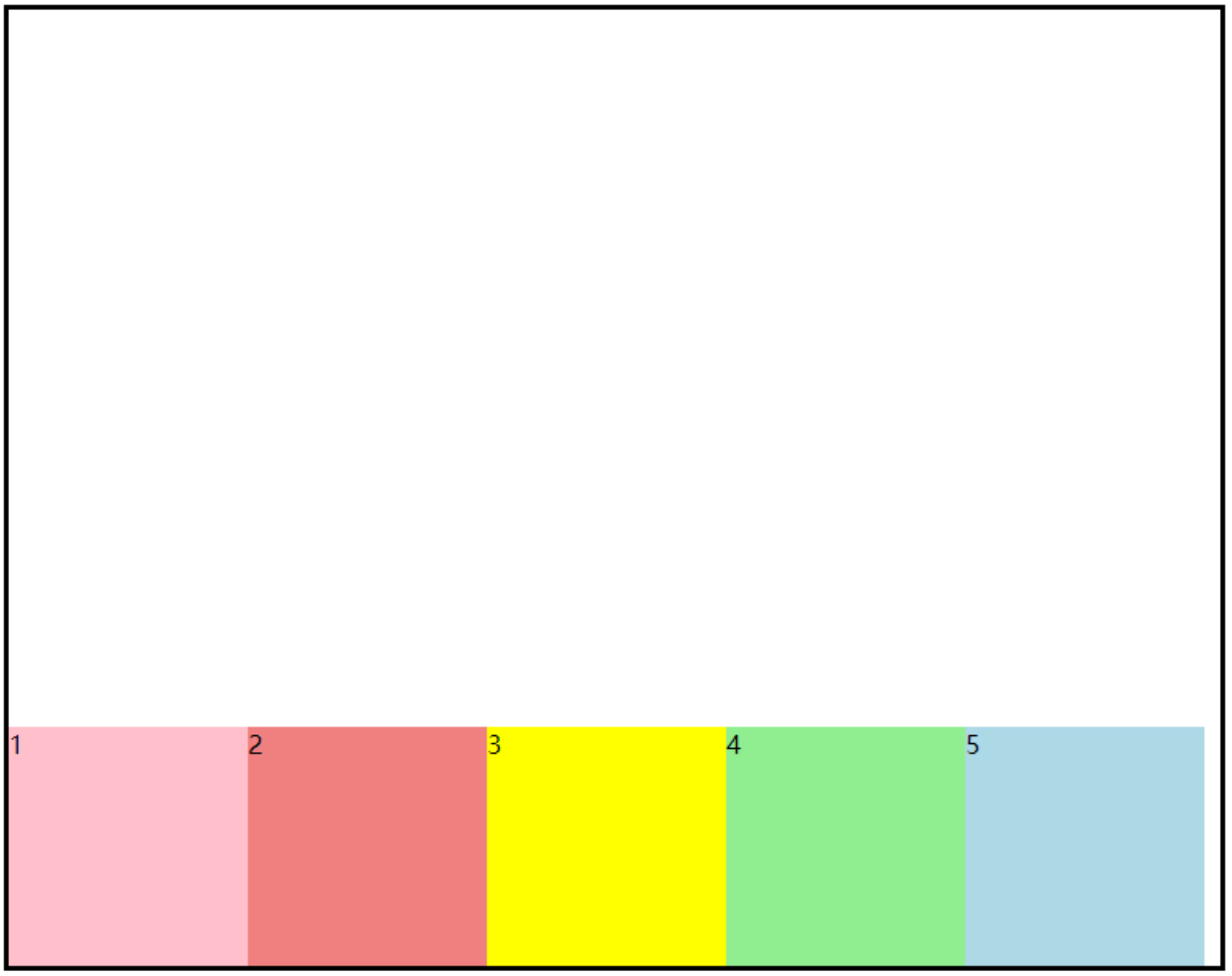
交錯軸的對齊設定。



範例：

Ex:

```
height: 600px;
align-items: flex-end;
```



二、Flex 空間計算規則

flex 是縮寫，裡面依序包含三個屬性 flex-grow、flex-shrink 和 flex-basis。

- 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，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 0，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。

grow, shrink 的計算概念，這兩者所填入的皆是整數，運作的概念上非常接近，都是按比例分配剩餘空間，只是兩者是在相反的情境下運作，先來介紹比較常使用到的 flex-grow，接下來再用相同的概念來理解 shrink。

（一）Grow 伸展值，分配剩餘的空間

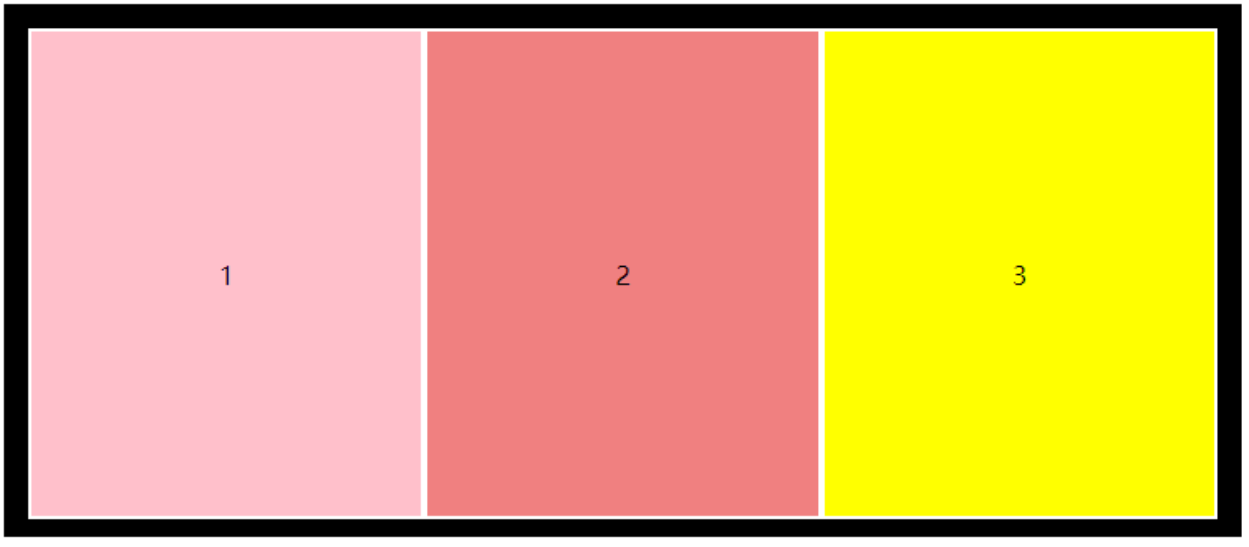
先建立一個簡單的基礎頁面，在一個外部容器上加上 display: flex;，內部則補上 flex: 1;，會得到如下的結果：三個等比切分的內元素（三個寬度各佔 33%）。

範例：

Ex:

```
<div class="container">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
</div>
```

```css
.container{
  background-color: black;
  display: flex;
  height: 300px;
  padding: 15px;
}
.box {
  height: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
  color: black;
  border: 2px solid white;
}
.box1 {
  flex: 1;
  background-color: pink;
}
.box2 {
  flex: 1;
  background-color: lightcoral;
}
.box3 {
  flex: 1;
  background-color: yellow;
}
```



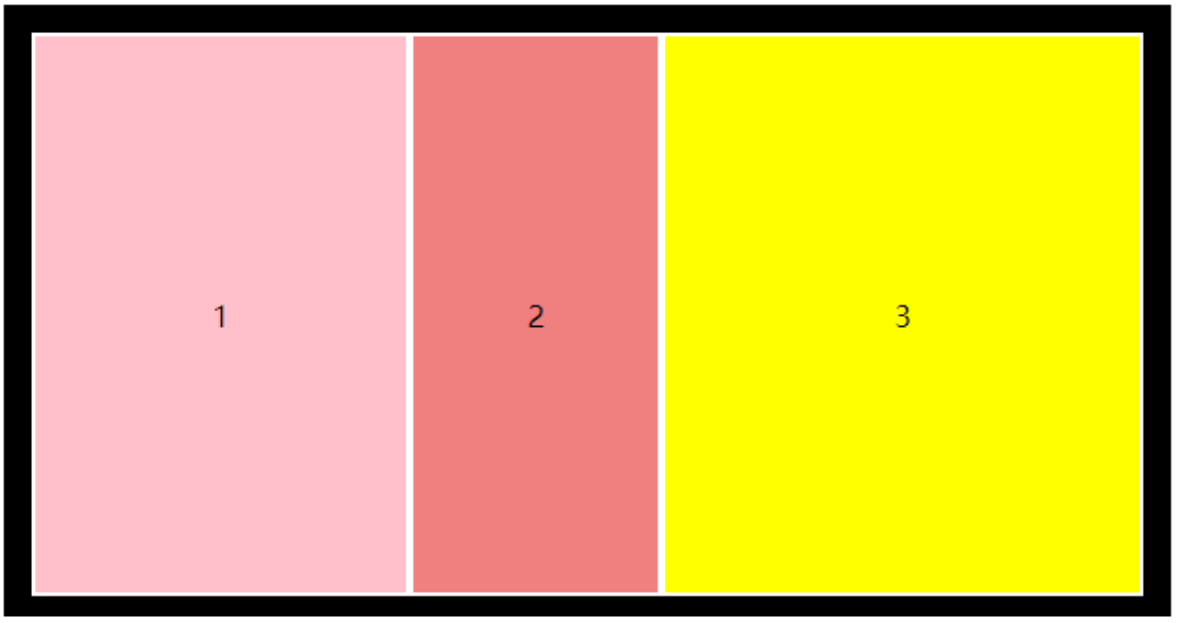
再補上一些設定：

- 外層容器設定為 600px
- box1 的元素設定 200px 寬度（flex-basis 強制設定）
- box2 及 box3 的元素設定 flex-grow: 1

```css
.box1 {
  flex-basis: 200px;
  background-color: pink;
}
.box2 {
  flex-grow: 1;
  background-color: lightcoral;
}
.box3 {
  flex-grow: 1;
  background-color: yellow;
}
```

接下來，將最 box3 的元素 flex-grow 改為 2

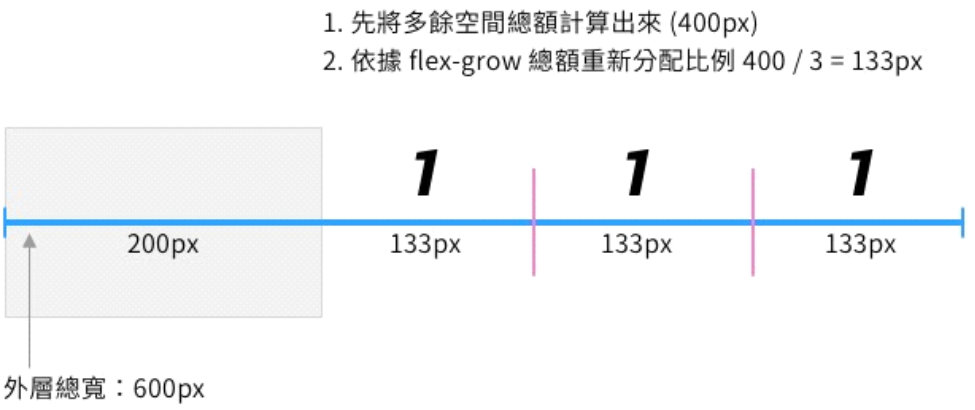
```css
.box3 {
  flex-grow: 2;
```

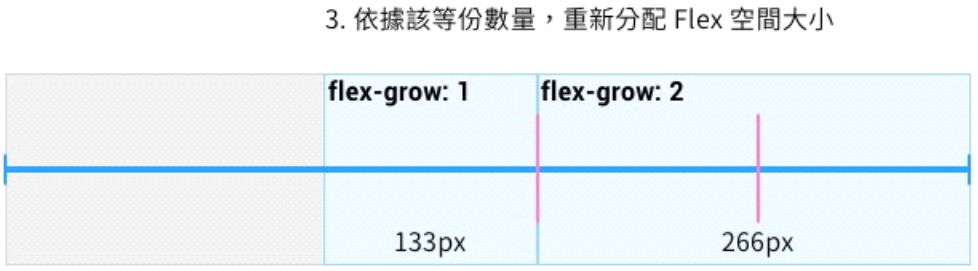


由於 Flex 是按比例分配剩餘空間，因此在上面的範例中，兩者 flex-grow 都是 1 的情況下，會呈現 1:1 的結果，這次改為 1:2 那麼寬度的計算上也會改為 1:2，重點如下：

- 分配空間是依據比例（flex-grow 的總和值再重新分配）。
- 是分配剩餘的空間，已經佔用的空間不會重新分配。
- 因此，以上方的範例來說：
  總寬度為 600px，最左方的元素佔用 200px 寬度，因此剩餘 400px，flex-grow 分別為 2、1，因此總和為 3，比例分配上為 400px / 3 = 133.33333px，左方寬度為 133 * 1，右方元素為 133 * 2 = 266。

1. 先將多餘空間總額計算出來 (400px)
2. 依據 flex-grow 總額重新分配比例 400 / 3 = 133px





（二）Shrink 收縮值，分配多餘的空間

flex-shrink 與 flex-grow 運作上則是相反，shrink 是將超出的部分重新分配，確保元素不會被裁切（如果足夠被分配完）。

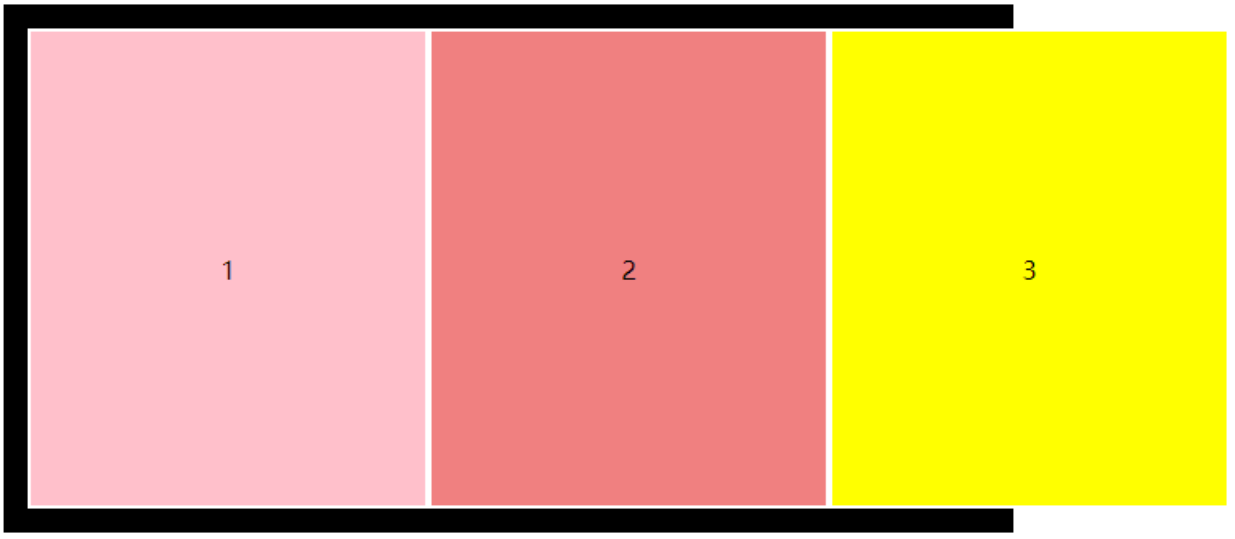
以下範例透過三個 flex-basis: 250px 來超出範圍，並且統一設定 flex-shrink 與 flex-grow 皆為 0（禁止伸展、收縮），可以得到以下的結果：內元素超出了外容器。

範例：

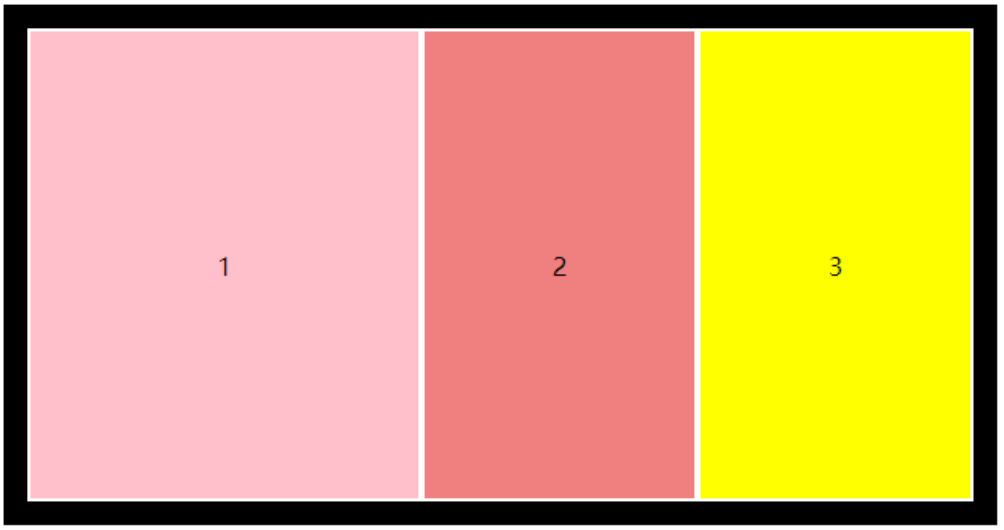
Ex:

```
<div class="container">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
</div>
```

```css
.box1 {
  flex-grow:0;
  flex-shrink: 0;
  flex-basis: 250px;
  background-color: pink;
}
.box2 {
  flex-grow:0;
  flex-shrink: 0;
  flex-basis: 250px;
  background-color: lightcoral;
}
.box3 {
  flex-grow:0;
  flex-shrink: 0;
  flex-basis: 250px;
  background-color: yellow;
}
```

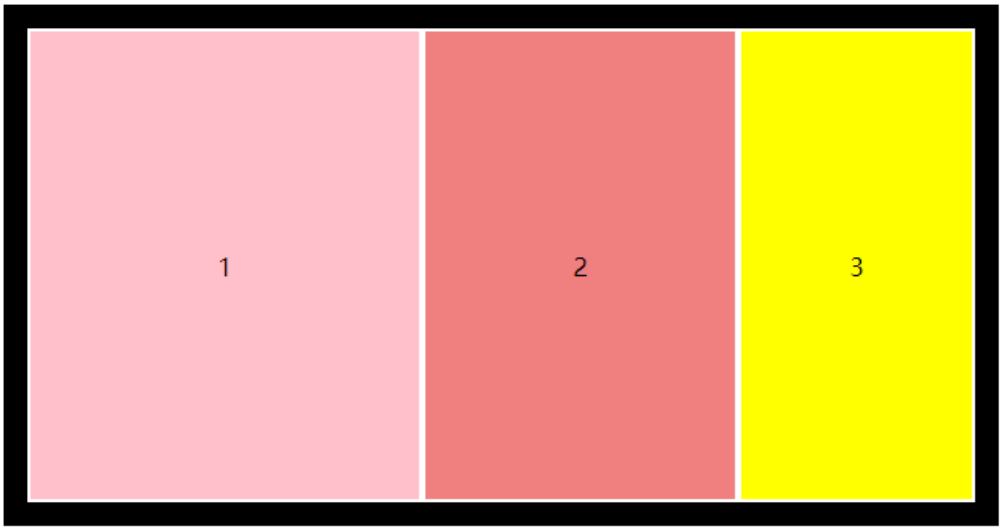


將 box2 與 box3 兩個元素 flex-shrink 設為 1，會看到兩個元素多餘的內容收縮到外容器內部並維持相同的寬度，最左方的元素則維持原本的寬度不變。



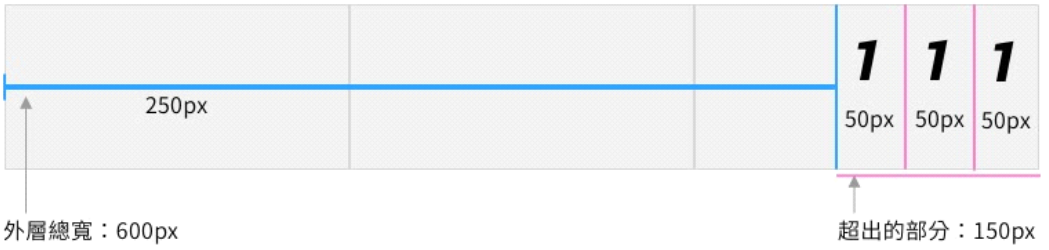
接下來，一樣進行 2:1 的 shrink 設置實驗，box2 為 flex-shrink: 1，box3 為 flex-shrink: 2

```
.box3 {
  flex-grow: 2;
```



結果兩者的尺寸竟然不是 2: 1 ！shrink 是分配多餘的空間，因此不是全部的空間上都重新分配，所以運作的結果如下：

1. 先將多餘的總額計算出（150px）
2. 依據 shrink 總數計算切分比例（50px）



3. 原始寬度依據 shrink 數值扣除