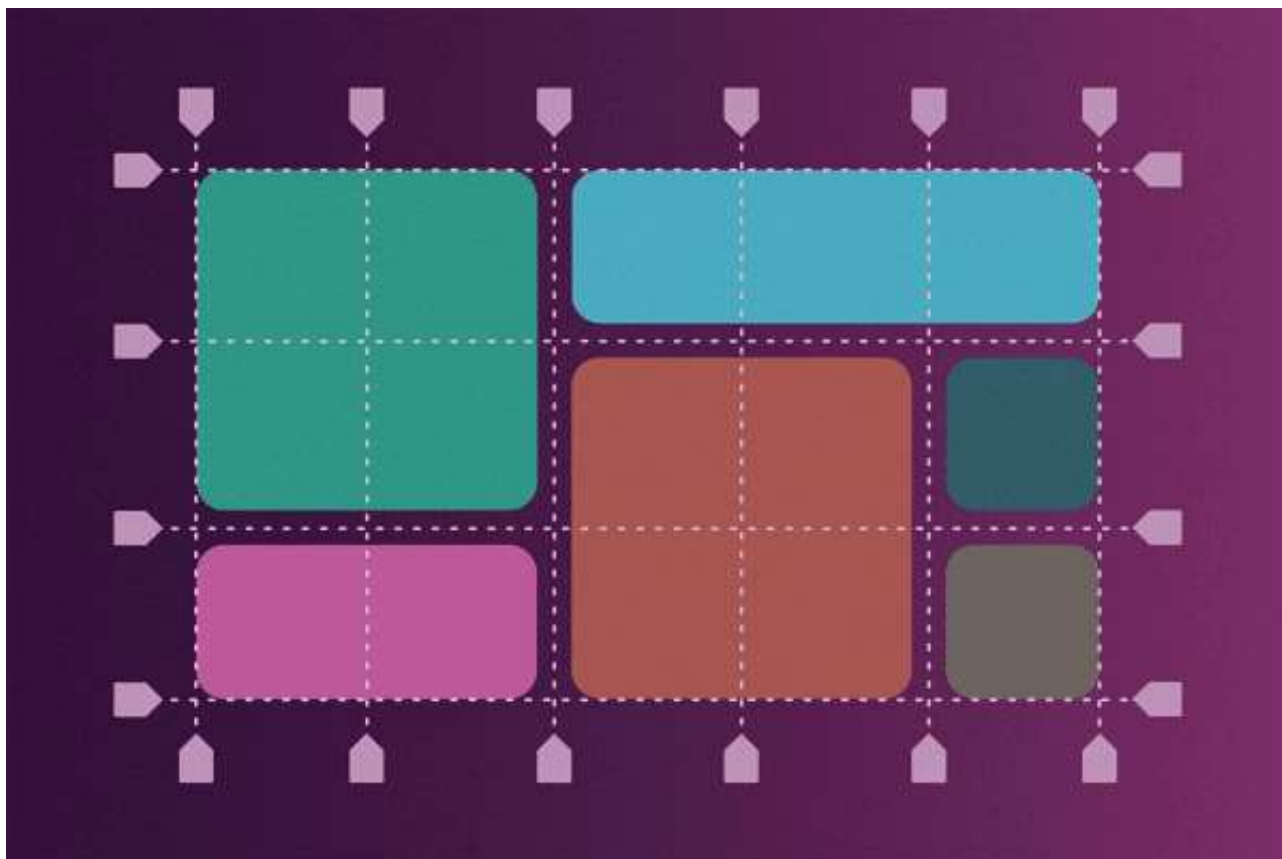


[首頁](#) > CSS

Css Grid 概念介紹及使用教學

呂 Partical Weng 10月 06, 2019 3 留言

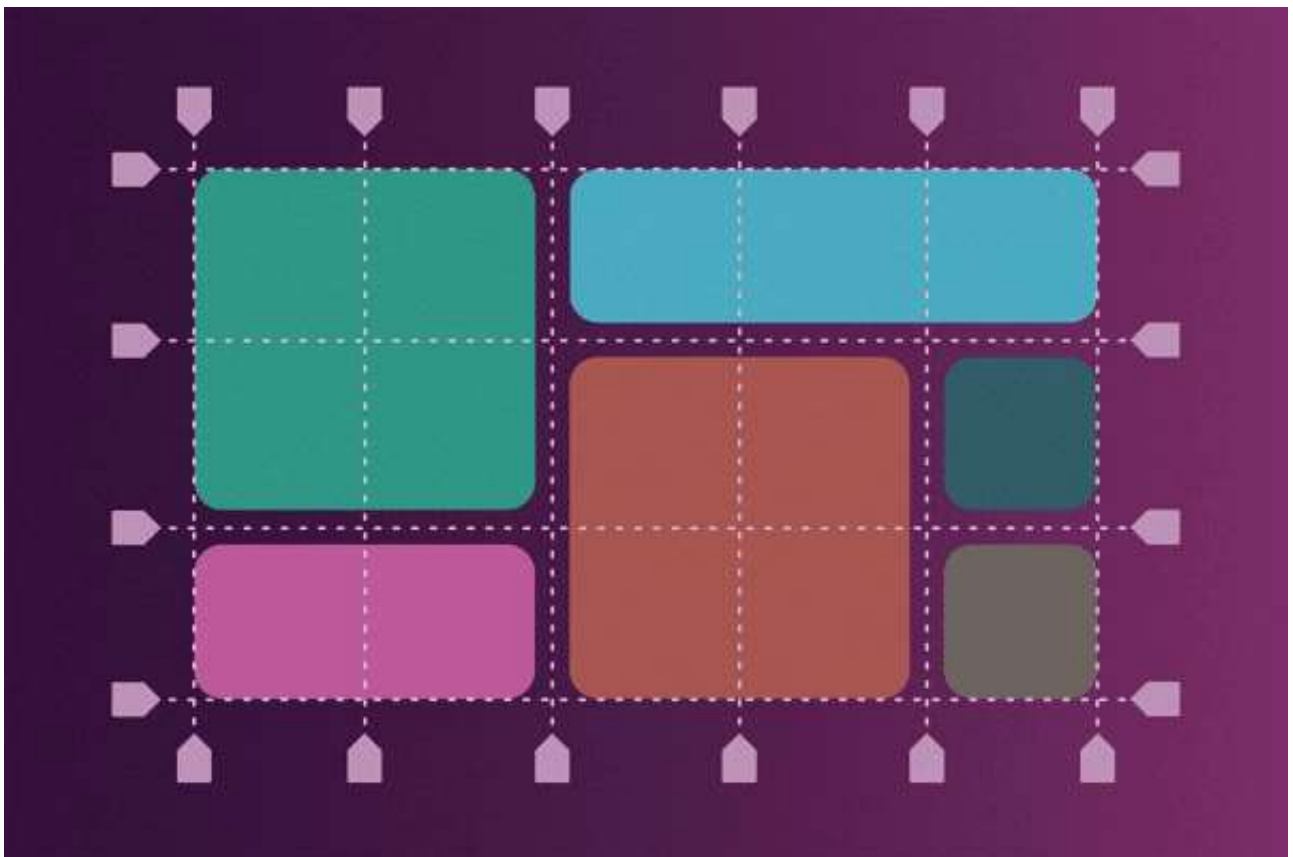


Css Grid 在使用上與 Flexbox 有許多相像之處，而最大的不同點在於，Css Grid 是以二維的角度來操作，而 Flexbox 則是一維。

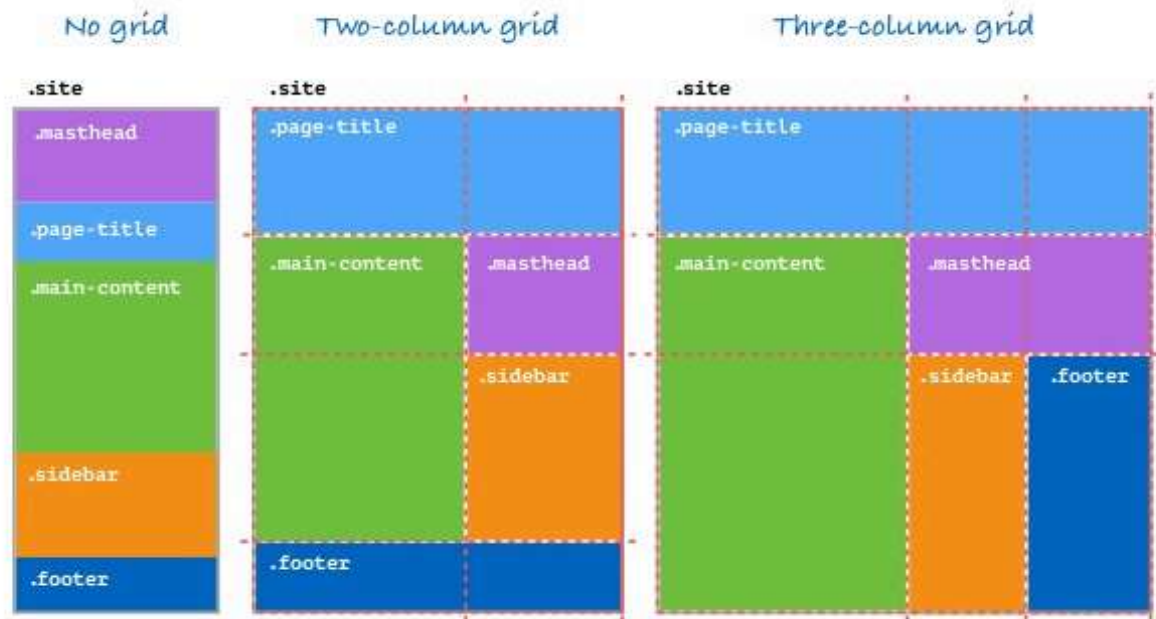
Css Grid 屬於相對較新的 Css 技術，在使用前，最重要的是確認目前瀏覽器的支援程度。而由下圖可看到，現今主流的幾大瀏覽器幾乎都能夠完整的支援了。



Css Grid 就像是平面繪圖軟體，有許多輔助線幫助你定位，擁有非常靈活自由的排版方式；他又像是Excel，以表格的形式排列基本單位。



而如果你想看 **Css Grid** 的實際應用例子，下圖就是一個很好的例子。對於現代網頁，**RWD** 已是基本配備，而透過 **Css Grid**，你可以更精準的分配每一項組件的位置，更靈活的排版方式，也更加美觀。

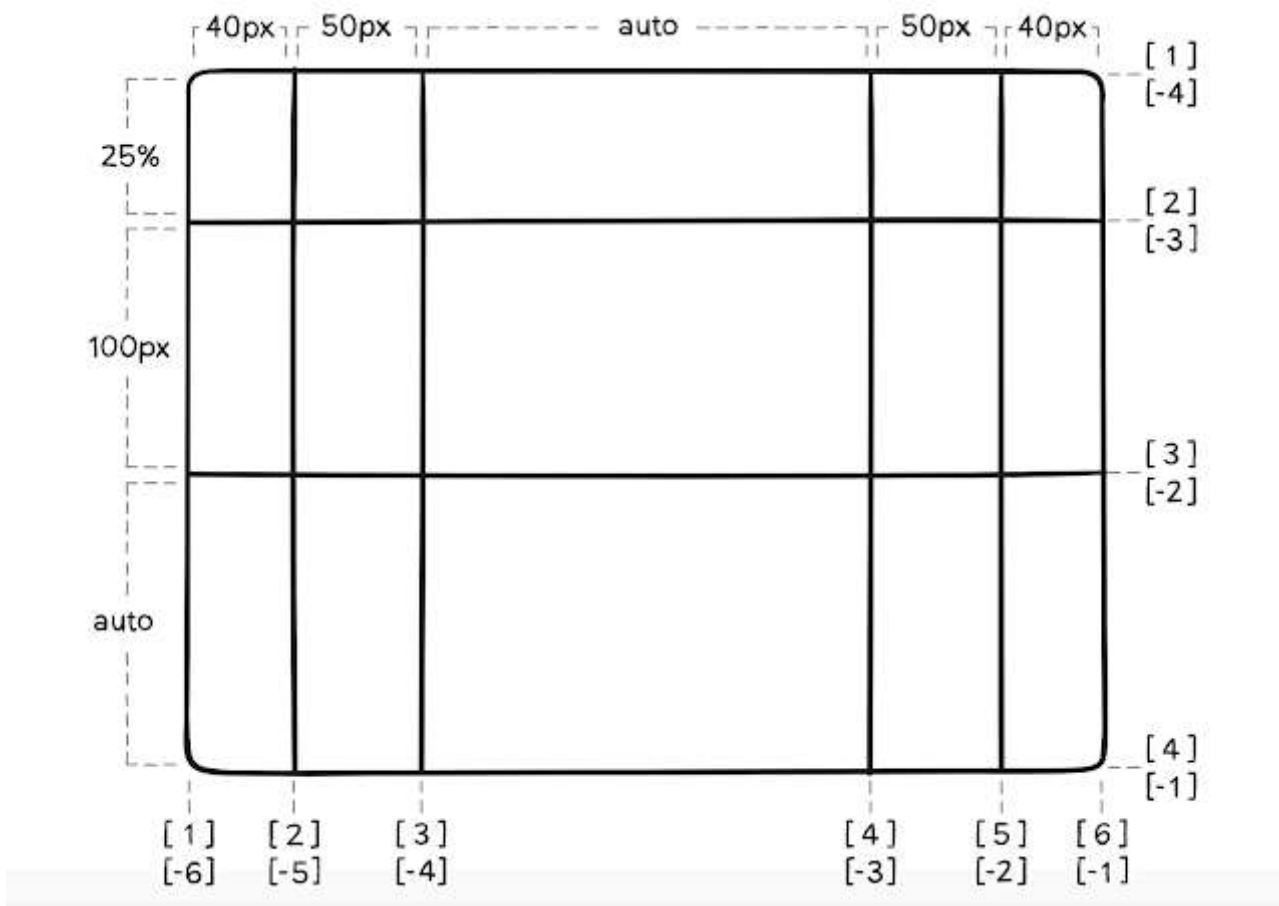


教學開始：

首先，你必須了解 **Css Grid** 的組成，一個 **Grid** 有兩個部分：**Container** (容器或框架) 與 **Item** (元件)。

基本概念是，由 **Container** 定義出框架大小，再將 **Item** 分配至其中。

```
<div class="container">
  <div class="item c">A</div>
  <div class="item b">B</div>
  <div class="item c">C</div>
</div>
```



Display

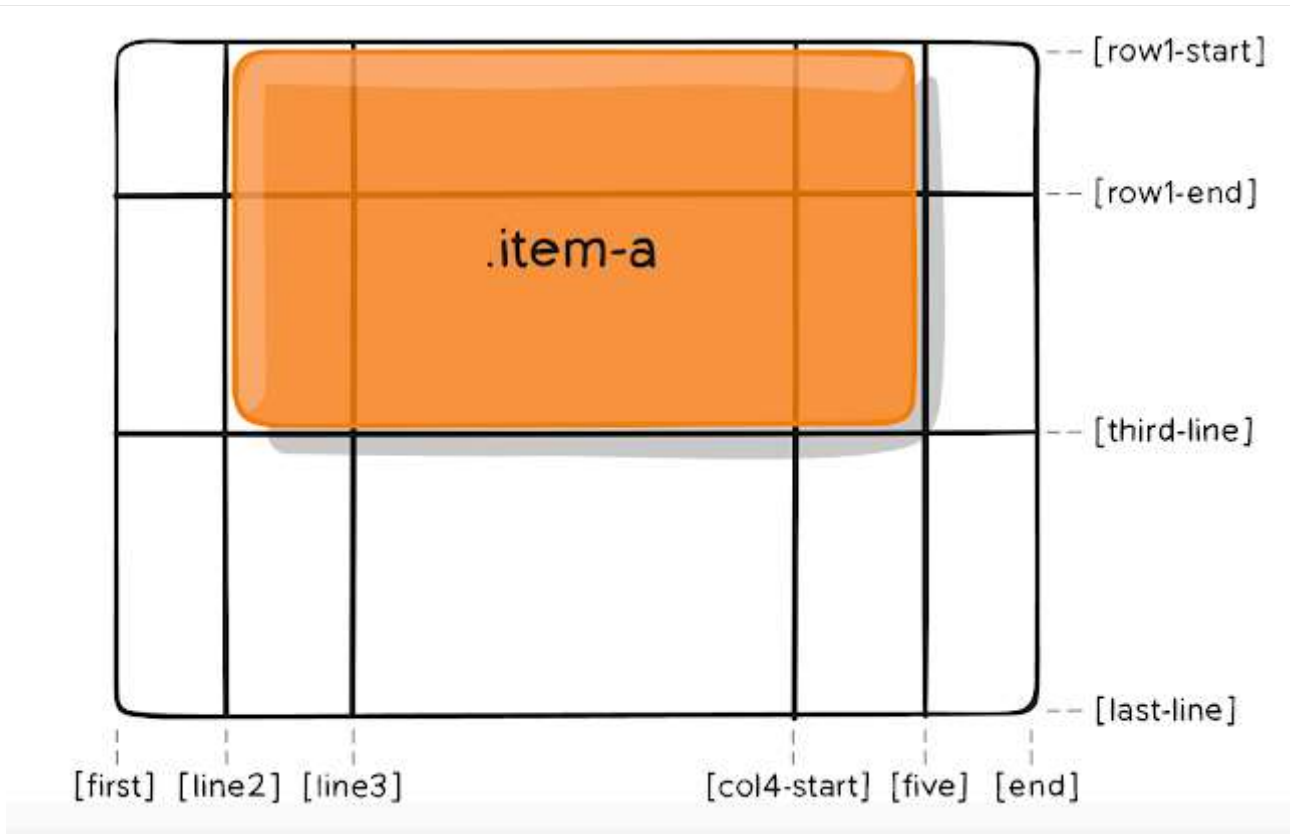
無論是哪種方式，在使用前，都必須在 **container** 的屬性加上此行，你可以選擇橫式排列或是縱式排列。

container :

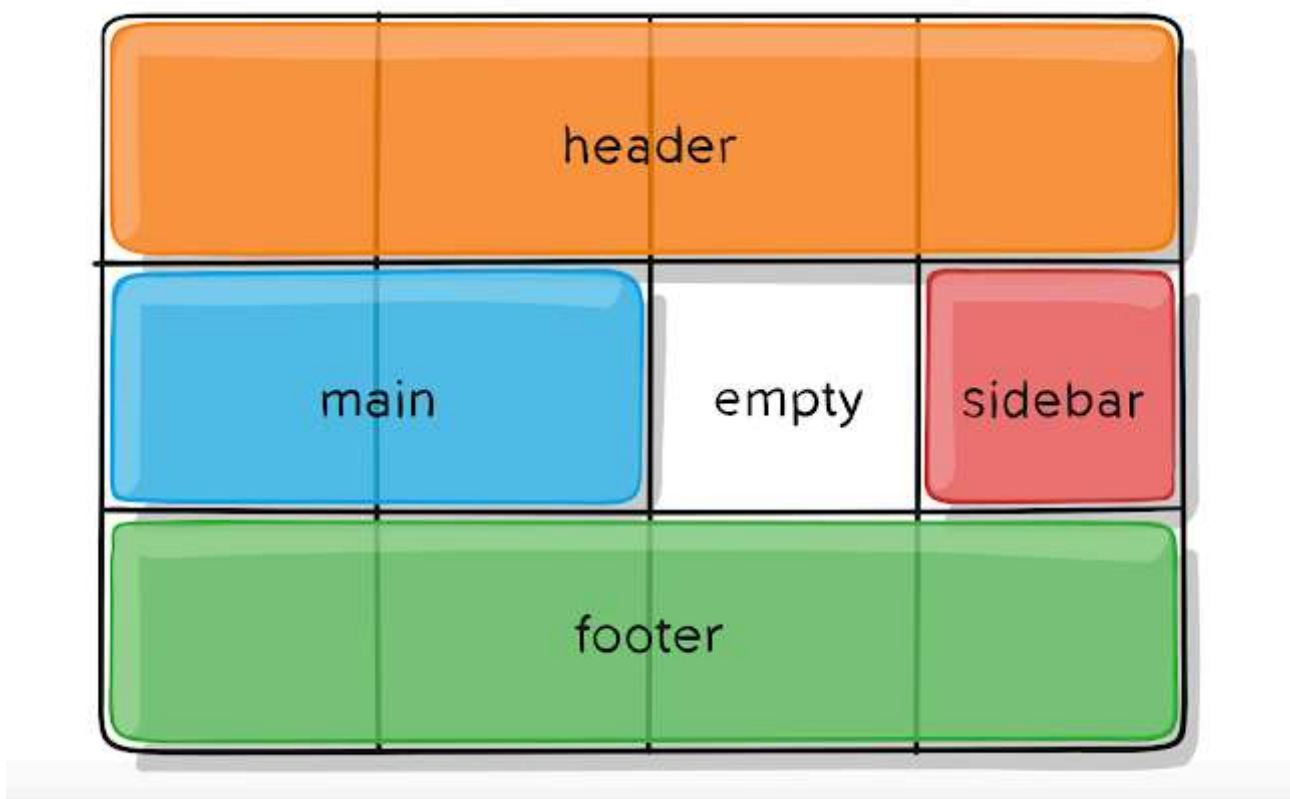
```
display : grid; /*(縱列)*/
         : inline-grid; /*(橫列)*/
```

接下來介紹幾種操作方式，因為 **Css Grid** 本身許多屬性操作是重複的，意思是你可以針對個別元素操作(像是 **grid-template-rows** & **grid-template-columns** 分開設定，也可以直接以 **grid-template** 一次設定兩者)。

1. **Template Layout** : **container** 設定各虛擬線間之距離，**item** 個別設定區域大小



2. Template Area : **container** 設定區域及區域名稱，**item** 只需設定 **container** 設定好的名字



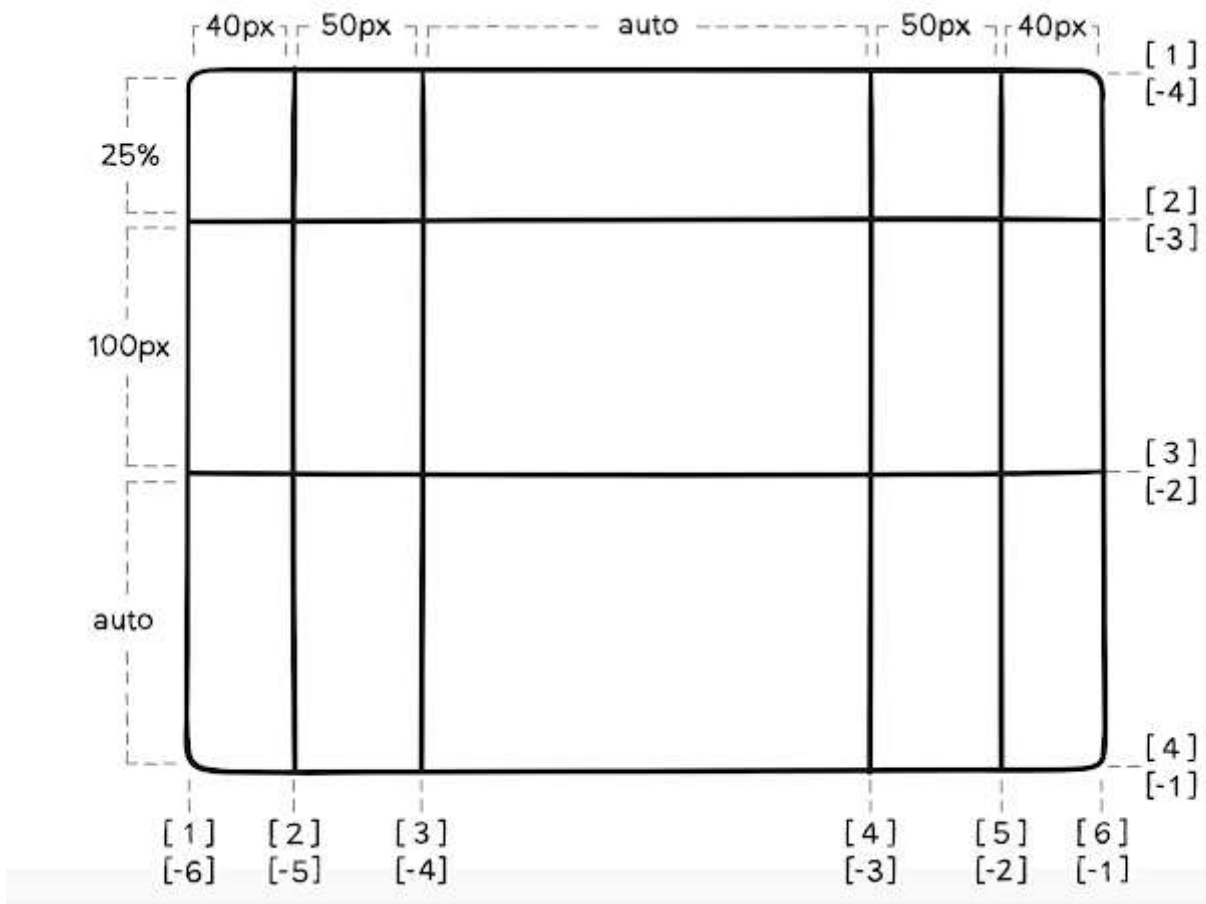
1. Template Layout

第1種非常好理解，由 **container** 定義出每條線之間的間隔，包含橫線與直

線。

container :

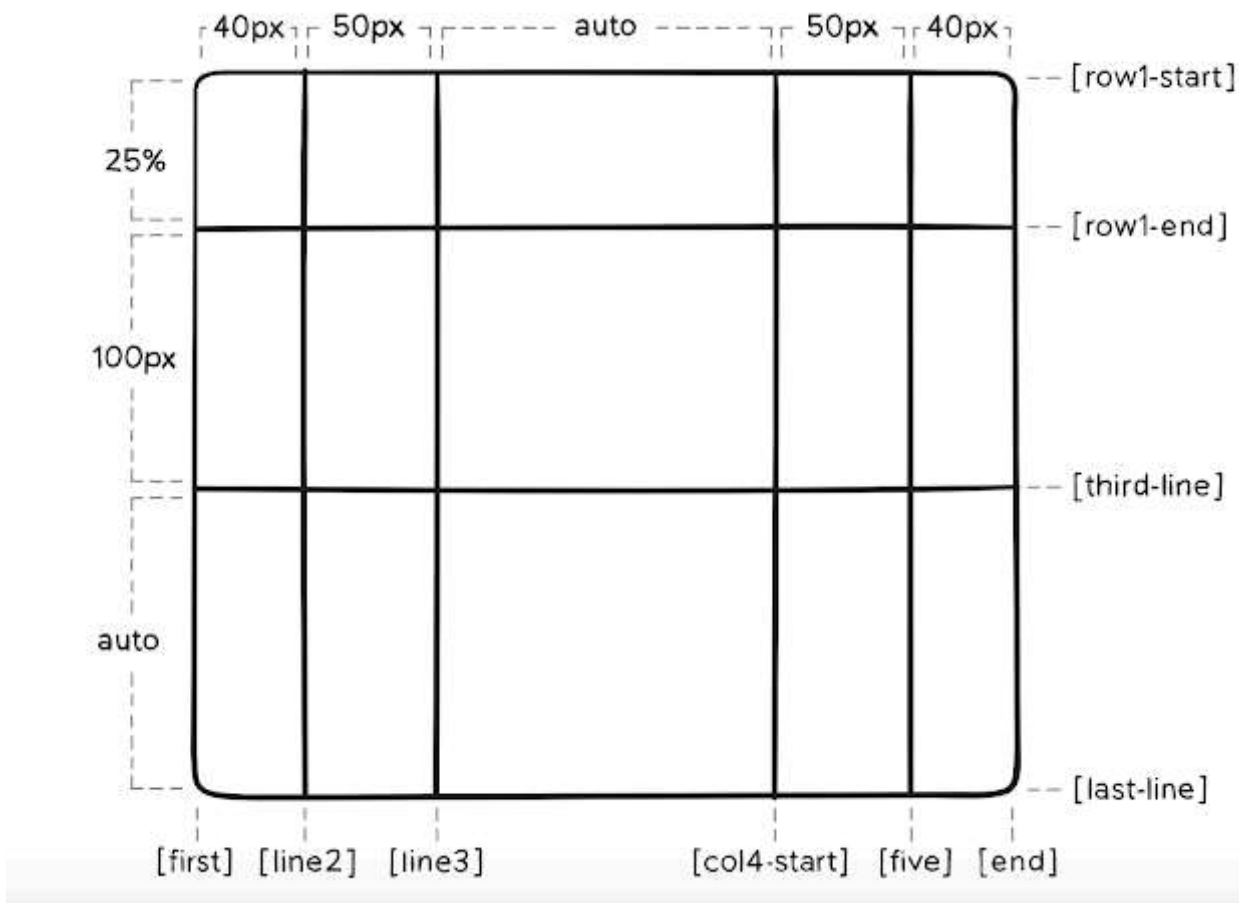
```
grid-template-columns: 40px 50px auto 50px 40px;  
grid-template-rows: 25% 100px auto;
```



在單位之前，可以為每一條線命名，[lineN]，[] -> 此括弧很重要，代表為線命名。

container :

```
grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start]  
grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto
```



分配欄位時，還有一個 `repeat` 的 function 可以使用，分配重複的距離定義

container :

```
grid-template-columns: repeat(3, 20px [col-start]);
/* 相當於 20px [col-start] 20px [col-start] 20px [col-start] */
```

如果有多條同名稱的線，分配給 **item** 時，可以在後方加上數字作為 index

item :

```
grid-column-start: col-start 2;
/* (col-start)的第2條 */
```

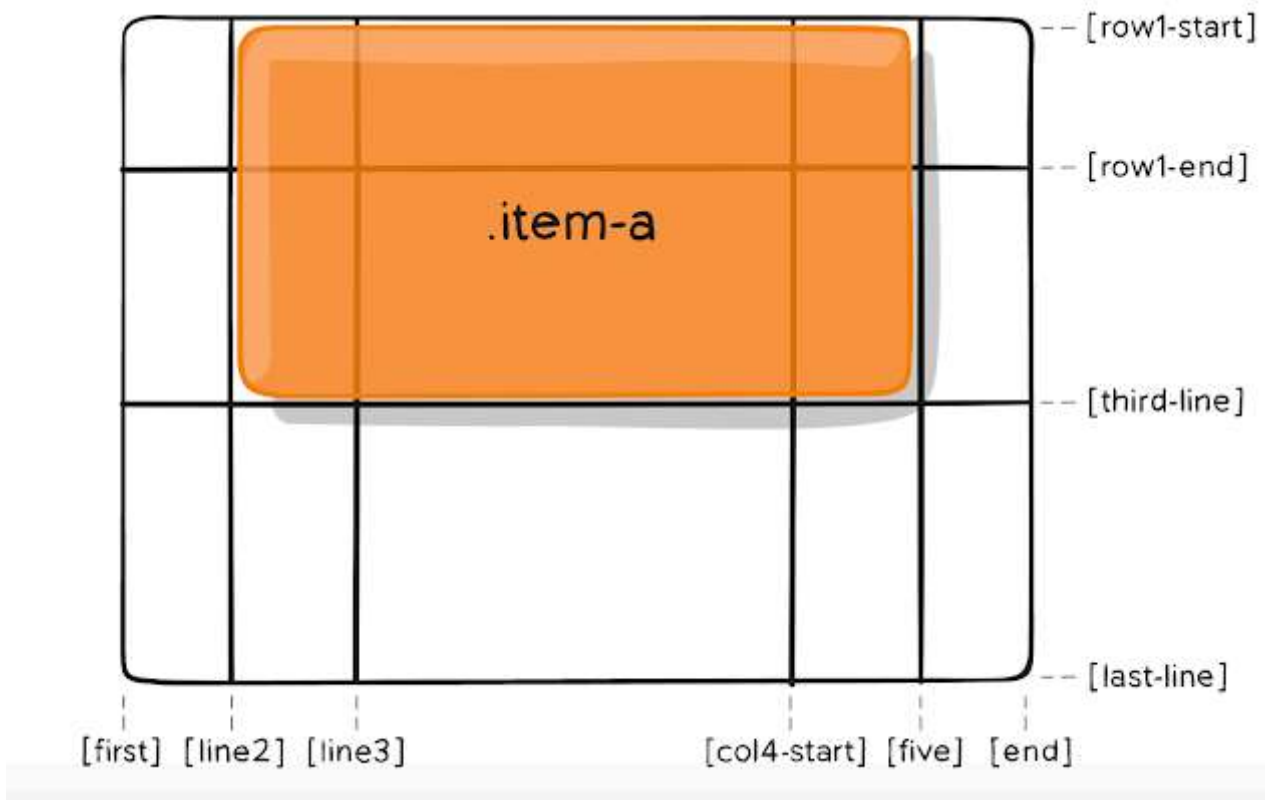
在形程表格的樣式後，就可以由 **item** 個別定義大小，定義上下左右4條線的位置，即可構成一個區域。

item :

```
grid-column-start: 2; (2nd line)
grid-column-end: 5; (5th line)
grid-row-start: 1; (1st line)
grid-row-end: 3; (3rd line)
```

/ 更簡單的方式 */*

```
grid-column: 2 / 5; /* (<startLine> / <endLine>) */
grid-row: 1 / 3; /* (<startLine> / <endLine>) */
```



也可以一次設定4條線的位置，一個屬性就可以解決，相當方便。

item :

```
/* <row-start> / <column-start> / <row-end> / <column-end> */
grid-area: 1 / col4-start / last-line / 6;
```


如果只加上一個單位，意指從此線開始至下一條線，一個單位。

item :

```
grid-column : 2
```

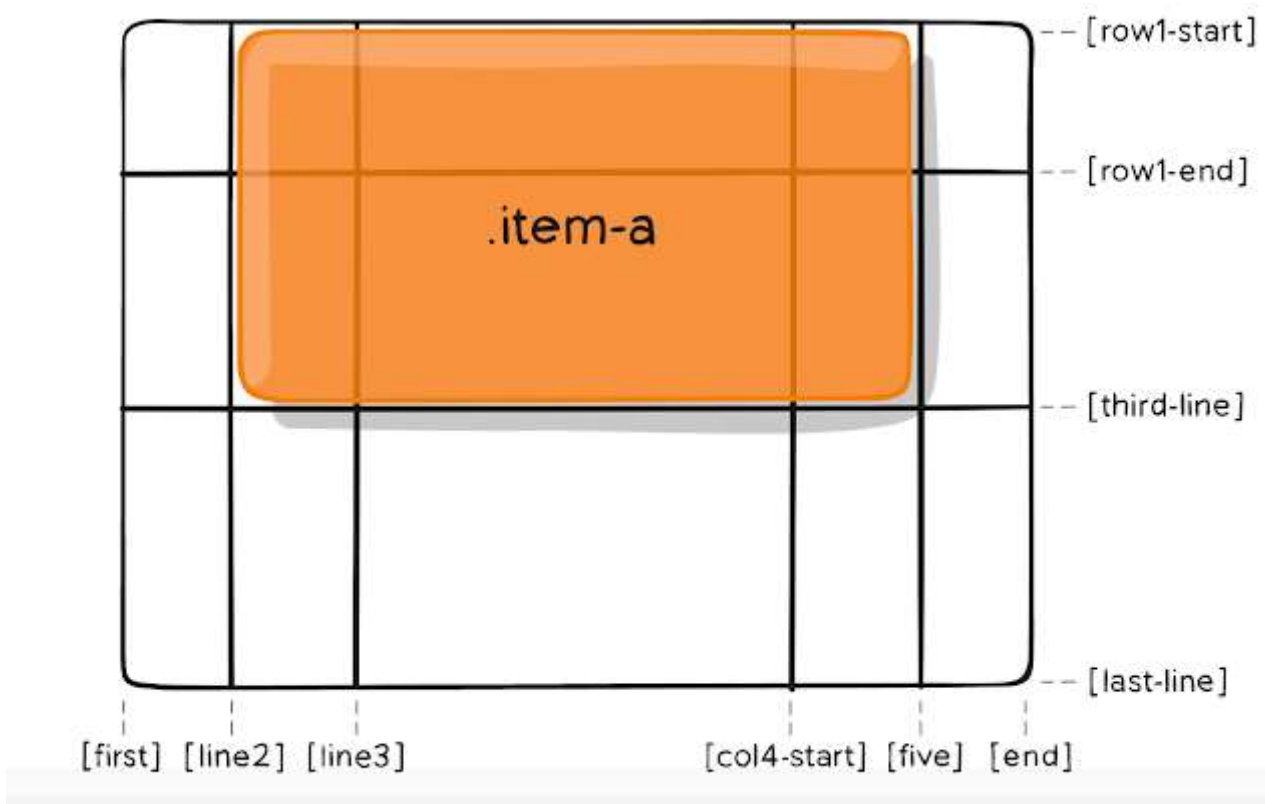
```
/*    == 2 / 3 , 第2條線至第3條線的區域 */
```

除了可以用數字代表第幾條線之外，也可以使用先前為 **Line** 定義的名稱（使用名稱時不用加 [] 此括弧）

item :

```
grid-column: line2 / five;
```

```
grid-row: row1-start / third-line;
```

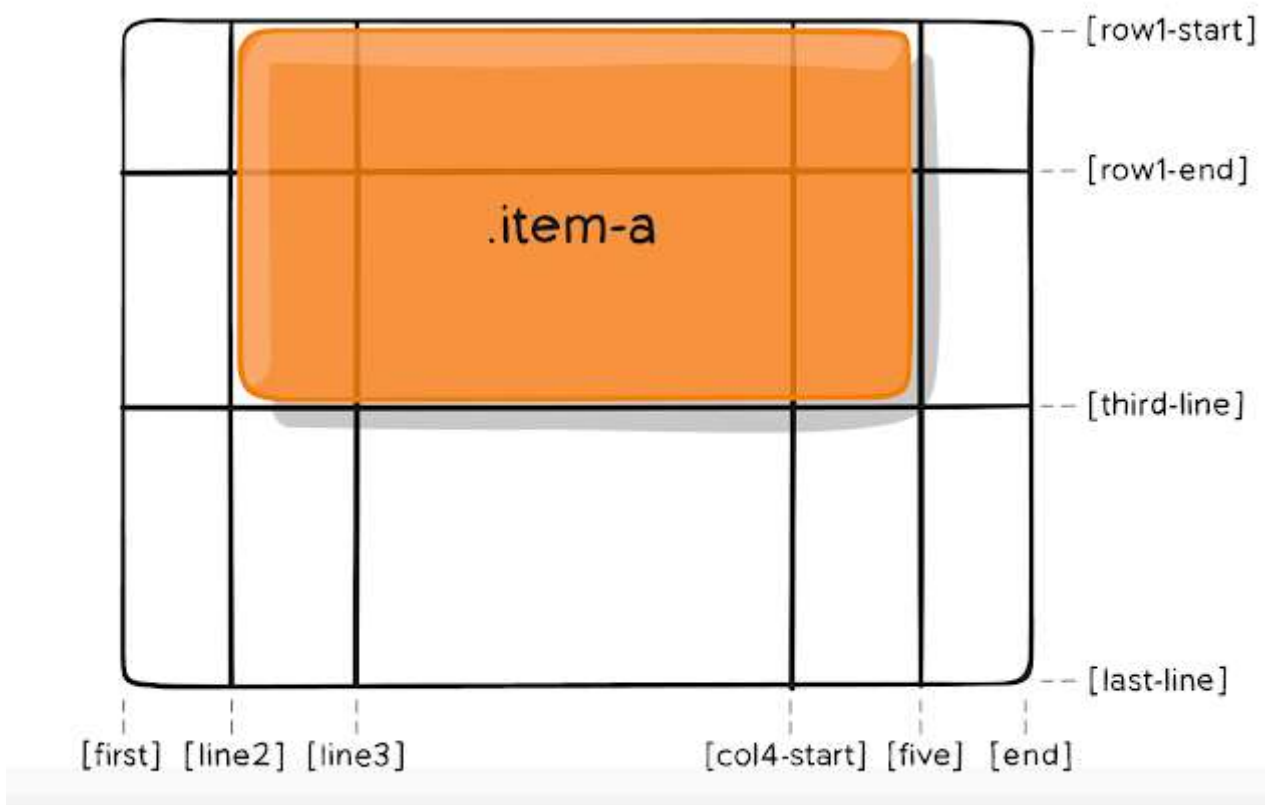


單位前可以加上 **span**，意指擴展一條線的距離

startLine 會往前一條(**startLine** - 1)，**endLine**會往後一條(**endLine** + 1)

item :

```
grid-column: span 3 / span 4;  
/*  
  == 2 / 5  
  == span line3 / span col4-start  
  == line2 / five  
*/
```



2. Template Area

第2種方式是我個人較為偏好的，當你分配好 **item** 的 **area name**，只要在 **container** 就可以輕鬆改變排版與配置。

(小提示：記得在分配名字之前，也必須像 **Layout** 一樣為版面設定每條線之間的間隔。)

item 只定義 **area name**

定義位置則完全由 **container** 分配

container :

```
grid-template-areas: "<grid-area-name> | . | none | ..."
```

在此屬性中，有三個比較特別的：

name -> 指定這個位置的 **item** 名稱

“.” -> 代表一個“空”的區域，也會佔有**area**，不過是**empty**的

none -> 未指定的區域

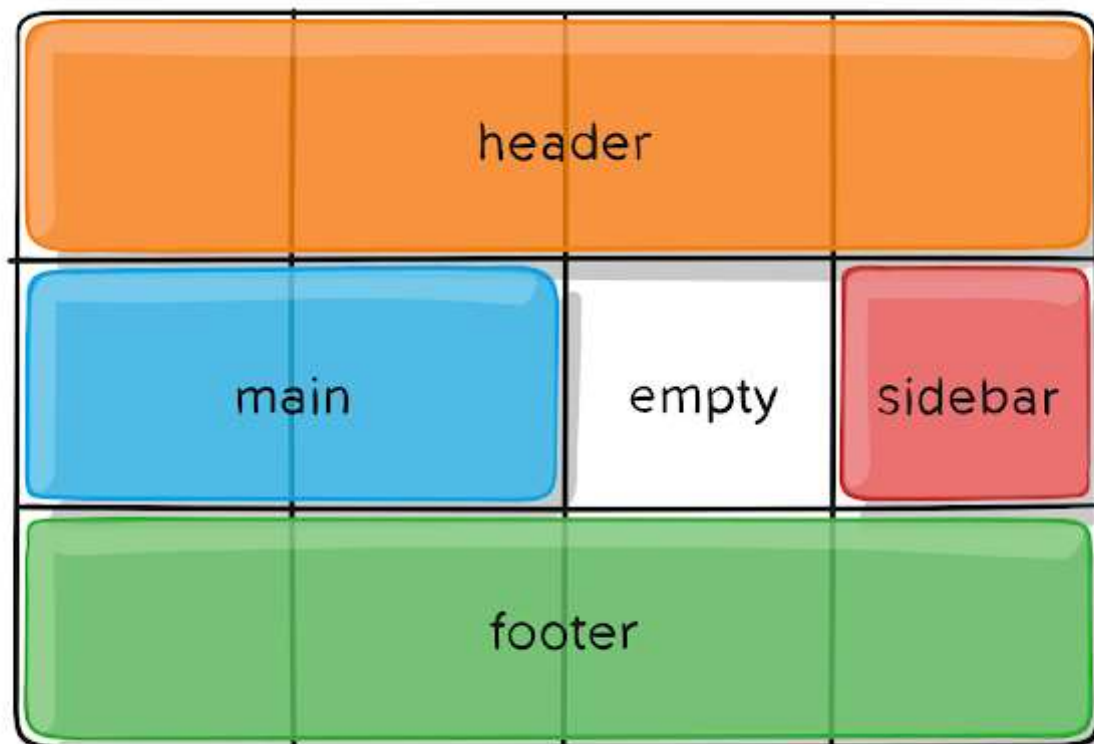
“.” 和 **none**的區域，如有其他未分配 **area name** 之 **item**，將會自動填滿空的部分

container :

```
grid-template-areas:  
  "header header header header"  
  "main main . sidebar"  
  "footer footer footer footer";
```

item :

```
grid-area: header;  
grid-area: main;  
grid-area: sidebar;  
grid-area: footer;
```



小提示：

在 **Area** 中，被分配的 **item** 不能是分段的或是非長方形

以下兩個皆為無法分配的例子

container :

ex.

grid-template-areas:

"header main header header"

ex2.

grid-template-areas:

"header1 main header2 header2"

"main main header2 header2"

單位

順帶一提，在 **Css Grid** 中有個特殊單位 - **fr**，它會將分配剩下的空間均分，如果父元素分配大小為 **300px**，而設定 **container** 為：

```
grid-template-columns : 1fr 1fr 1fr ;  
/* 1fr = 100px */
```

如果為

```
grid-template-columns : 1fr 50% 3fr 50px ;  
/*  
    50% = 150px  
    300px-150px-50px = 100px  
    而還剩下4fr，所以由4fr均分100px  
    1fr = 25px  
*/
```

基本上概念與%十分相似，但更方便是你無需計算佔有大小或百分比，你只需要輸入比例，像是 $1:2 = 1fr\ 2fr$ 。

也可以使用 **auto**，即為自動分配大小。

而其他常見單位，**css** 支援的各種單位都可以用上，**px, em, rem, % ...**

最重要的是，**各單位可以混用！**
非常方便與靈活的調整。

對齊與間隔

以 **Layout** 與 **Area** 兩種方式排版，已經能夠靈活地玩出許多花樣了。而當然不只這樣，**Css Grid** 連對齊與間隔都幫你想好了，這麼簡單強大的排版還不學嗎？

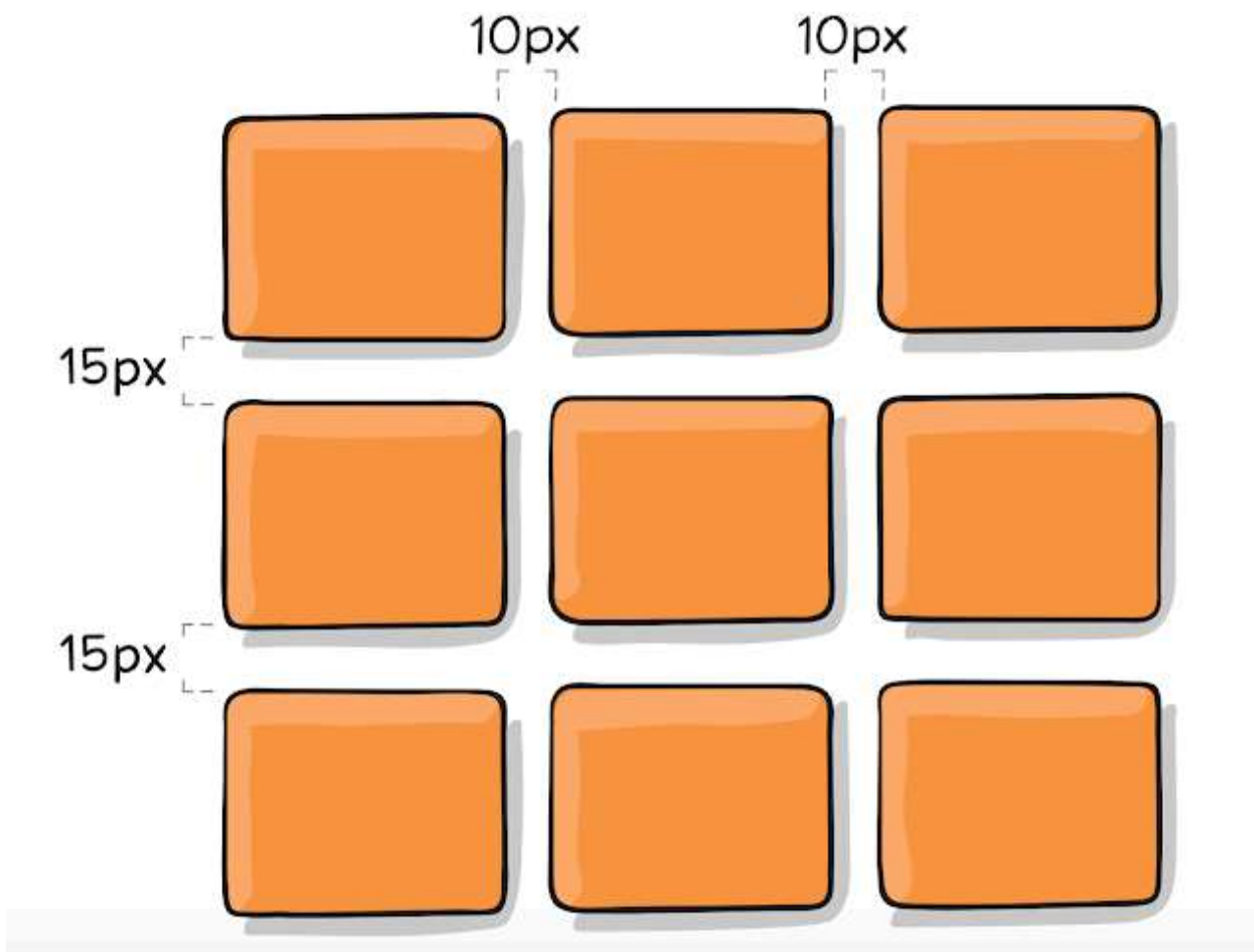
間隔

也有兩種方式可以設定 **gap** (間隔寬度)
可以獨立分開設定或是一次設定

container :

```
grid-column-gap: 10px  
grid-row-gap: 15px
```

```
/* grid-gap: <grid-row-gap> <grid-column-gap> */  
grid-gap: 15px 10px;
```



對齊

你可以為個別 item 設定對齊方式，也可以由 container 統一設定所有 item。

-items 結尾的為 container 一次設定所有 items

-self 結尾的為 item 個別設定自身的對齊方式

container :

```
justify-items:  
align-items:  
place-items:
```










Item:

justify-**self**:
align-**self**:
place-**self**:

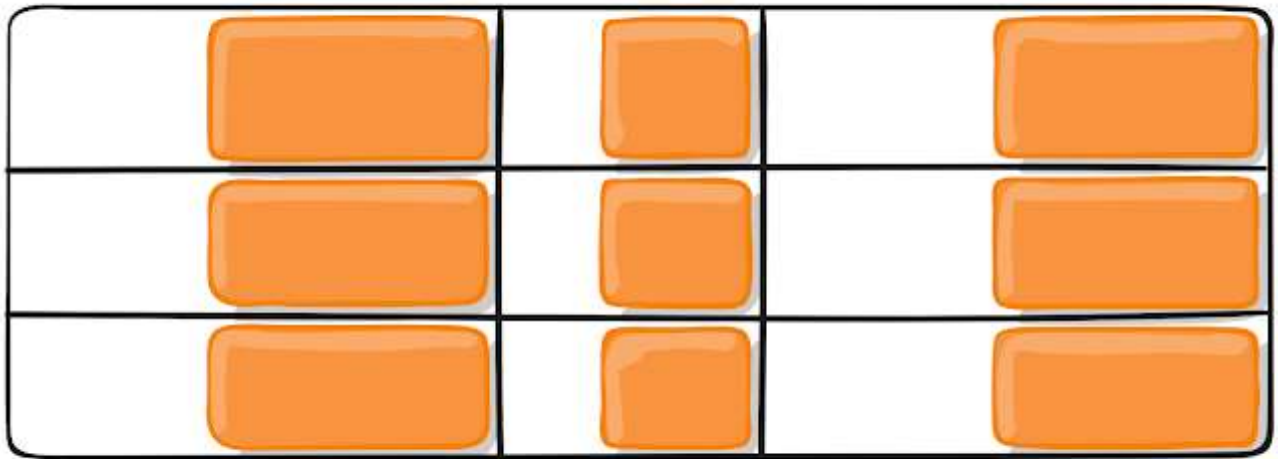
justify-self
justify-items

就只有4種屬性值

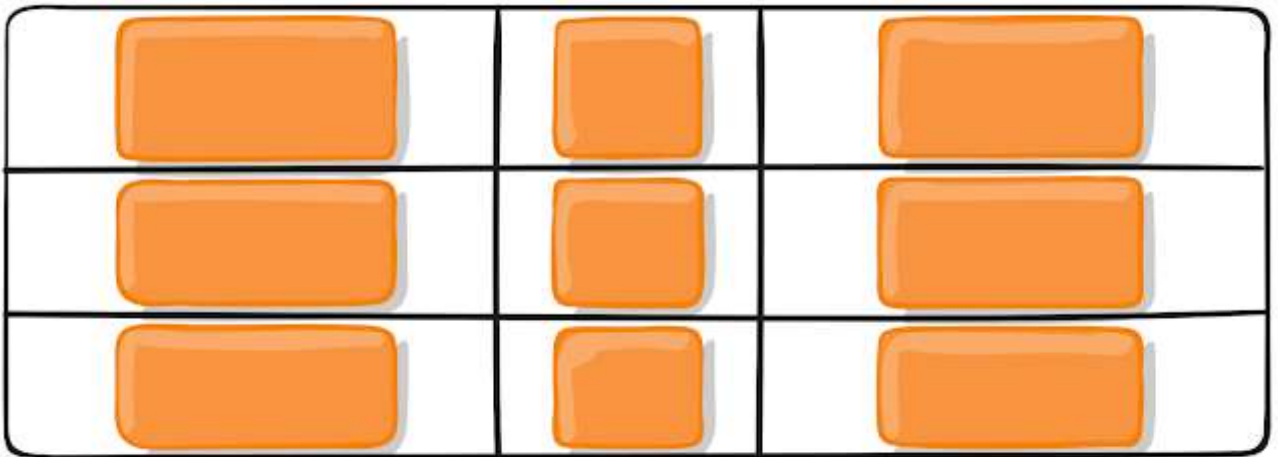
start : 相當於 float: left , 對齊startLine

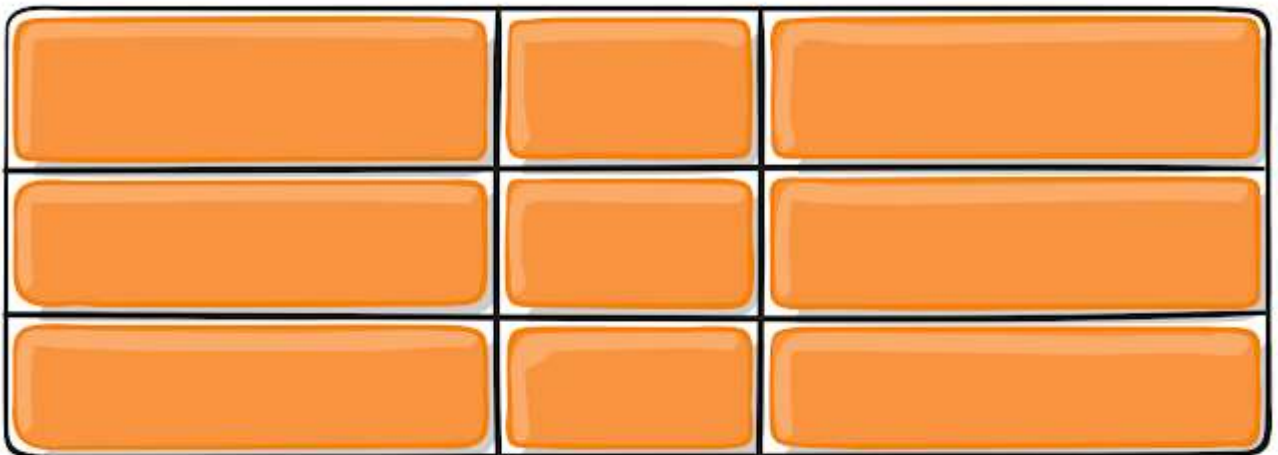
end : 相當於 float: right , 對齊endLine



center : 相當於 float: center , 置中對齊



stretch : 延展至整個 item , 為預設屬性



item :

```
justify-self: center;
```

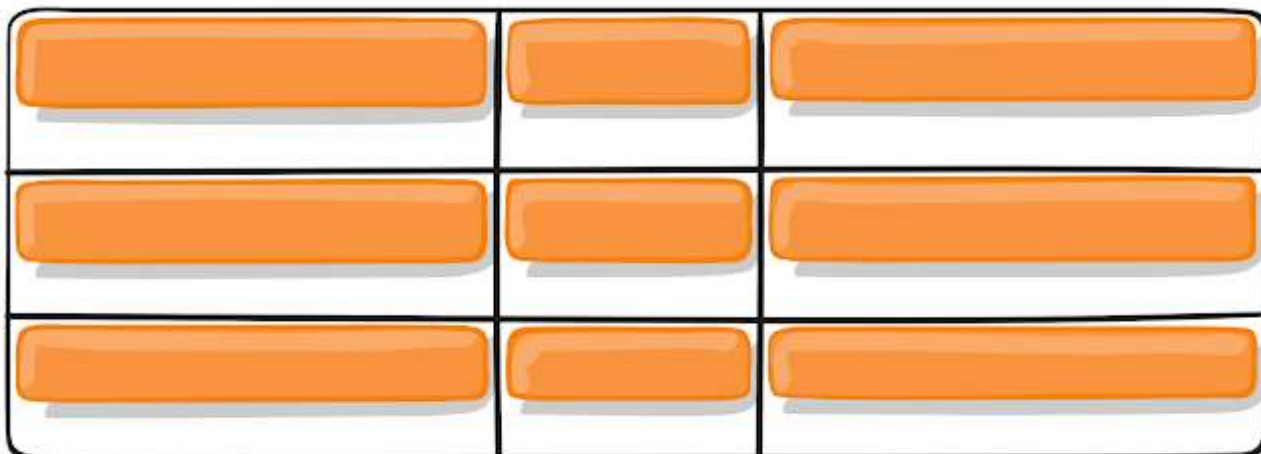
container :

```
justify-items: center;
```

align-self
align-items

這也完全相同，只有4種屬性值

start : 相當於 float: top，對齊 startLine



end : 相當於 float: bottom，對齊 endLine

(實際上是沒有 float top or bottom 的，但意思相同)

item :

```
align-self: center;
```

container :

```
align-items: center;
```

place-self place-items

當然還有一種整合兩者的屬性，也就是一次調整 justify-self 和 align-self

container :

```
/* place-items : <align-items> / <justify-items> */
```

```
place-self: center stretch;
```

```
/*
```

```
如果只設定一個值
```

```
則為同時設定兩個屬性
```

```
*/
```

```
place-items: center;
```

```
/* == center center */
```

除了 item 中的對齊外，還有所有 items 之於 container 的對齊方式，直接看下圖片介紹會更加清楚。

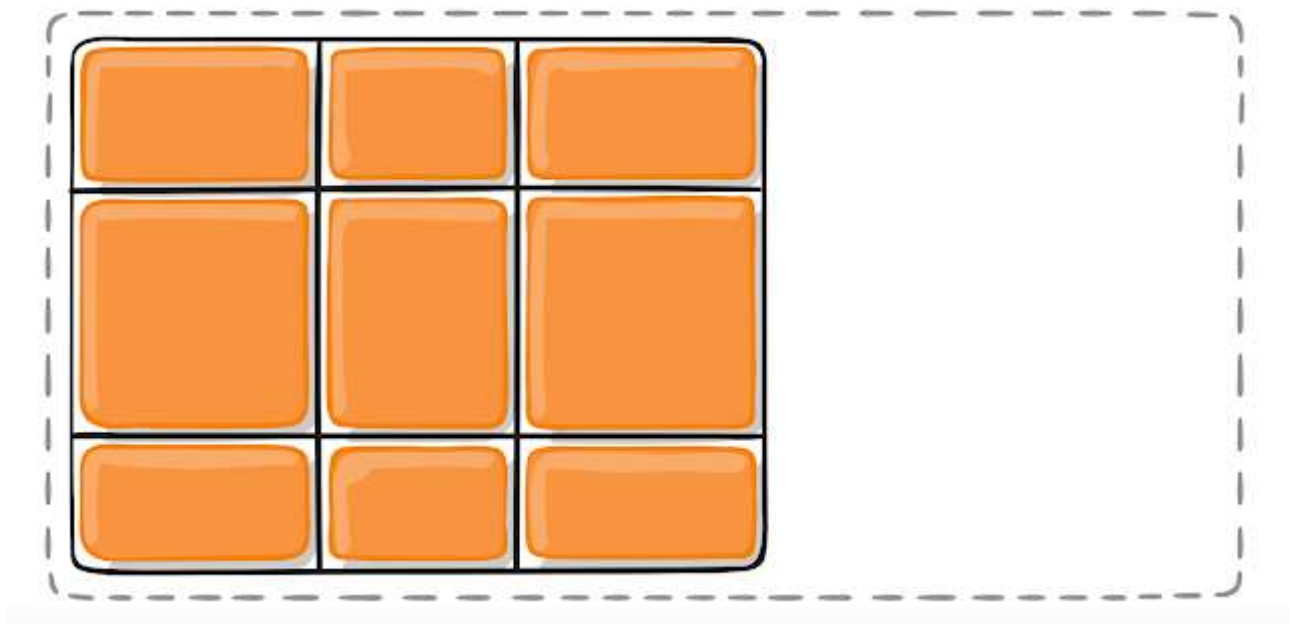
結尾為 -content 的屬性是以整體的形式對齊。

簡單來說，就是變成以行列為單位處理

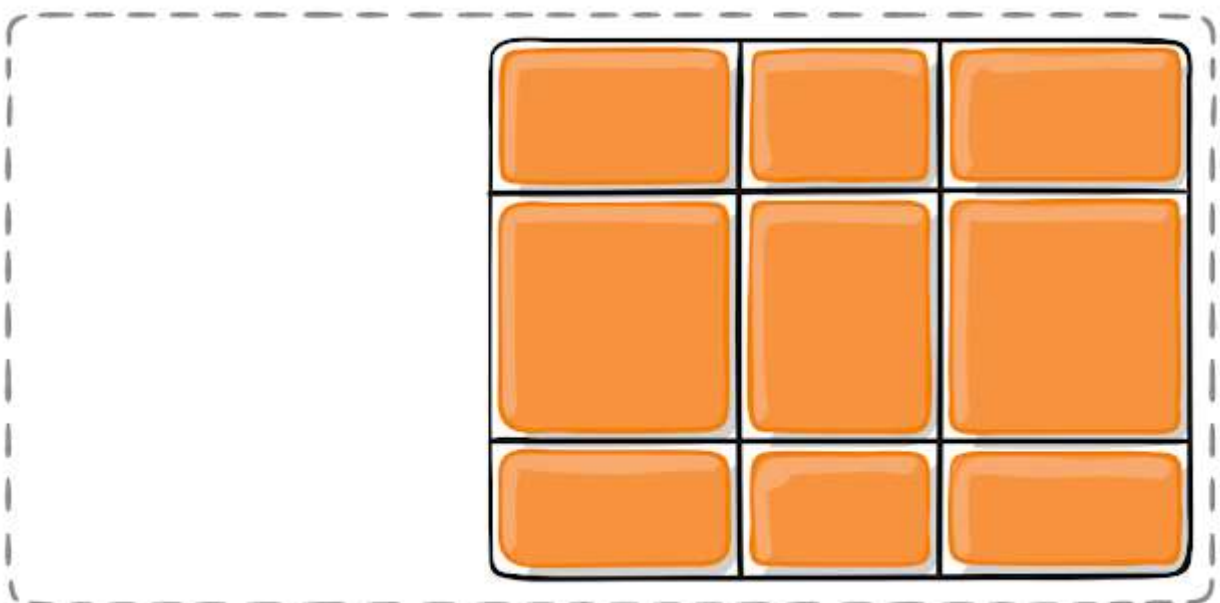
除了原本的4種屬性之外，額外增加了3種特別的屬性，控制行列間的間距值：

justify-content

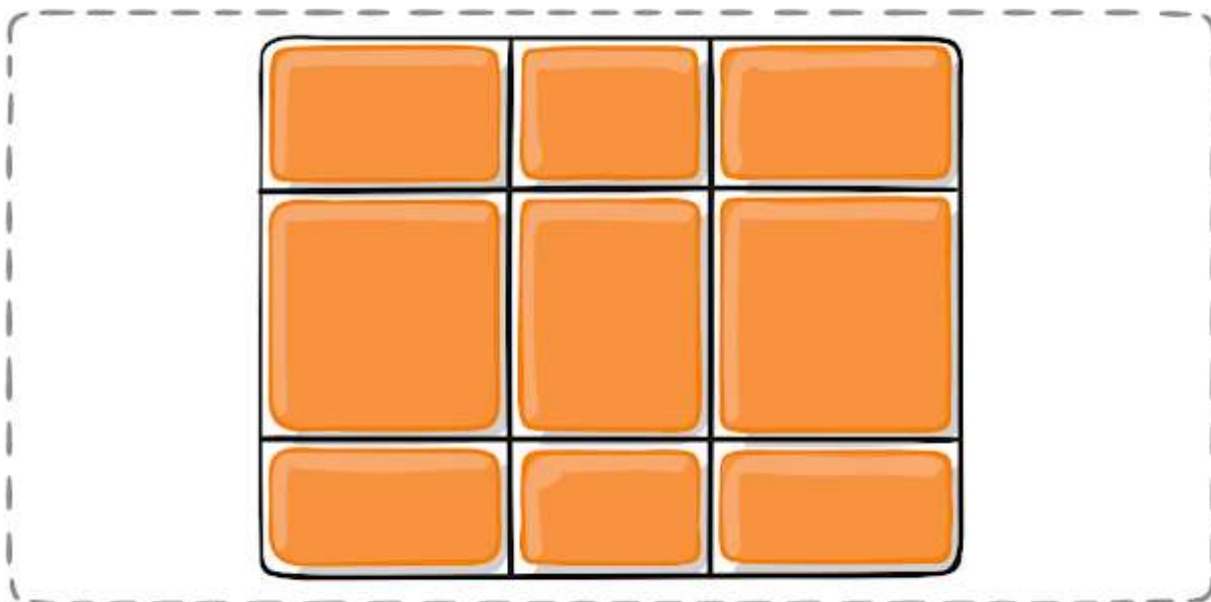
start



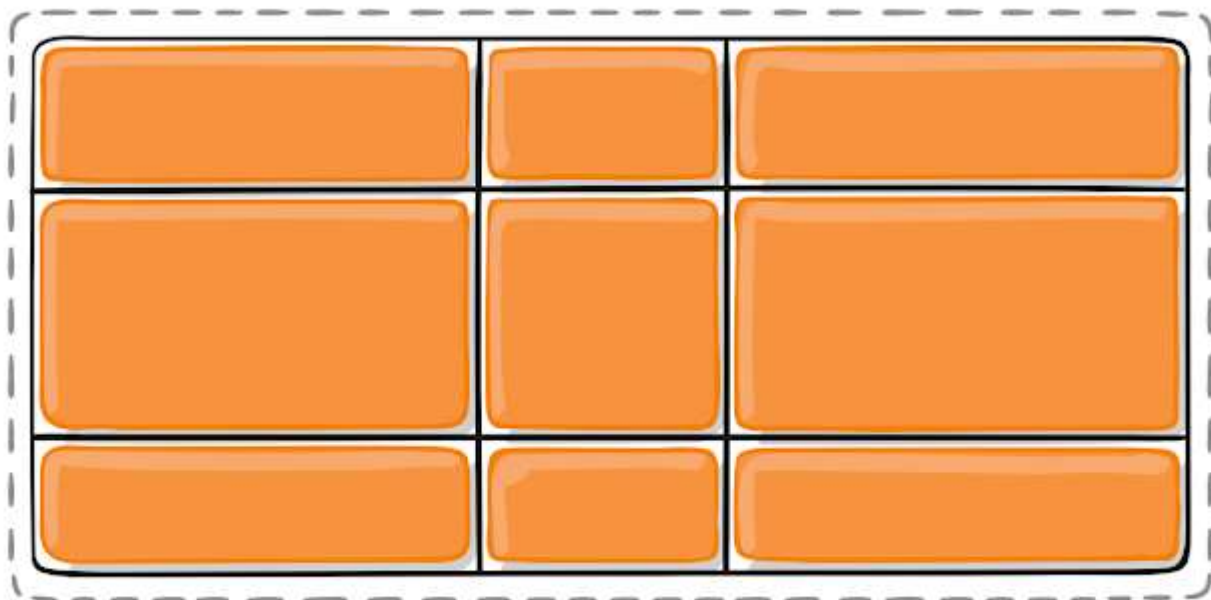
end



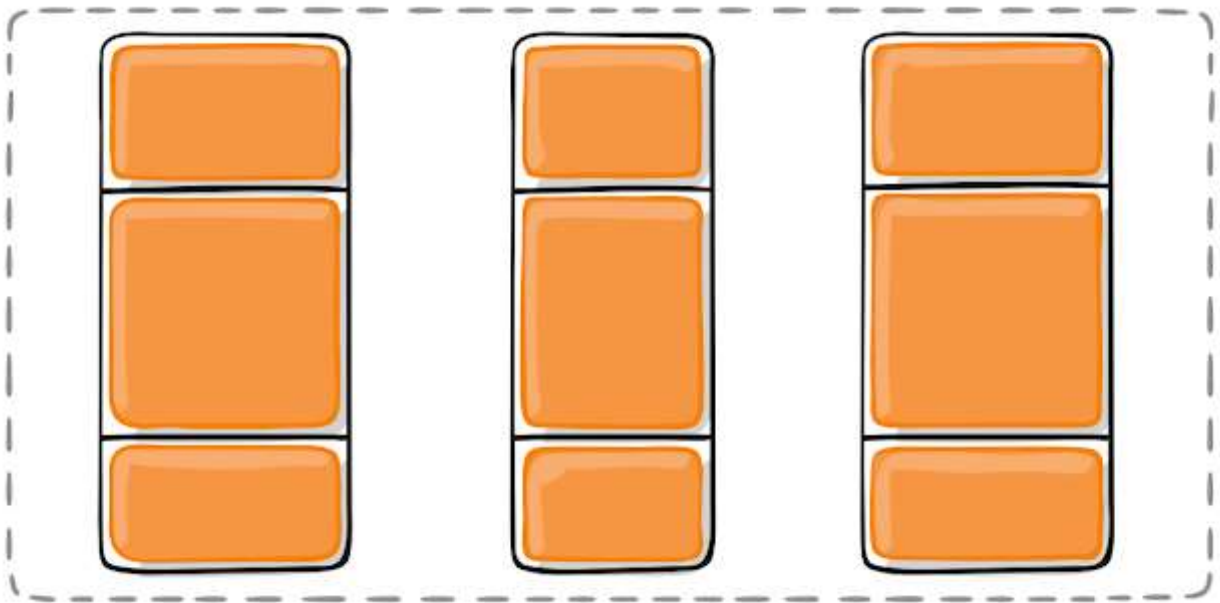
center



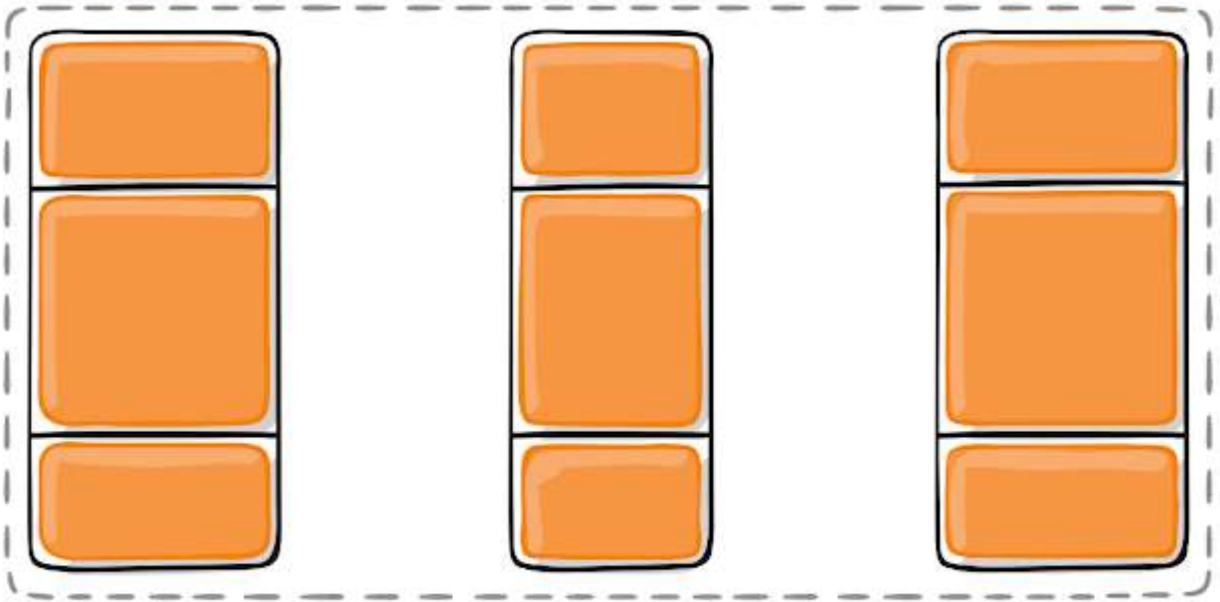
stretch



space-around : 首尾的空間，僅為行列間間距值的一半

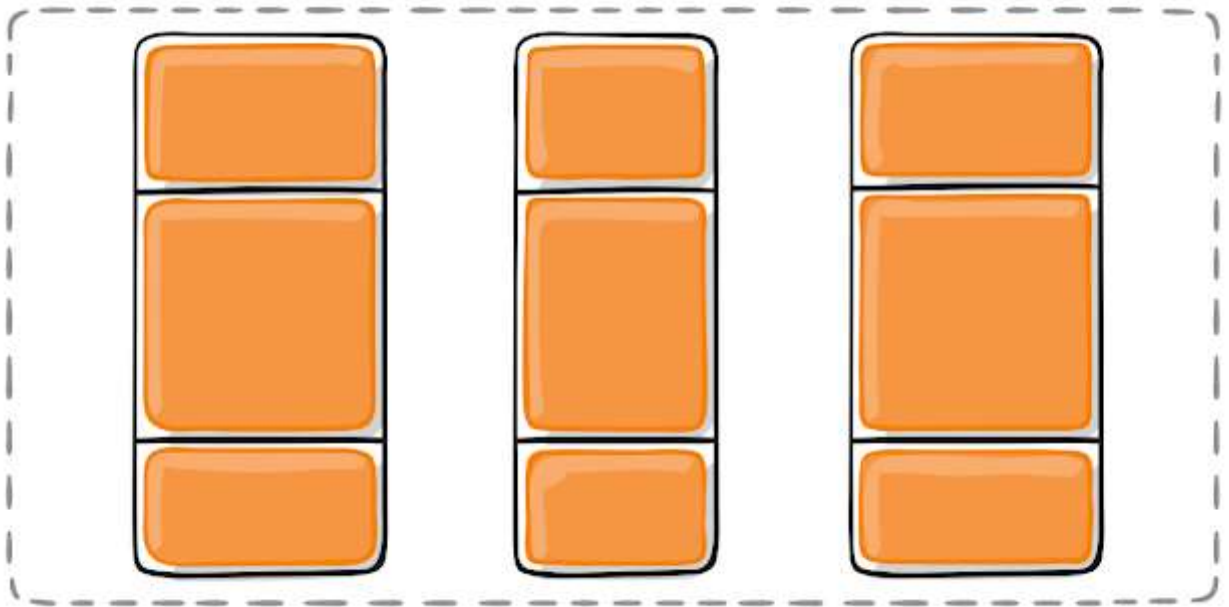


space-between：首尾的空間，無行列間間距值



space-between：首尾的空間，無行列間間距值

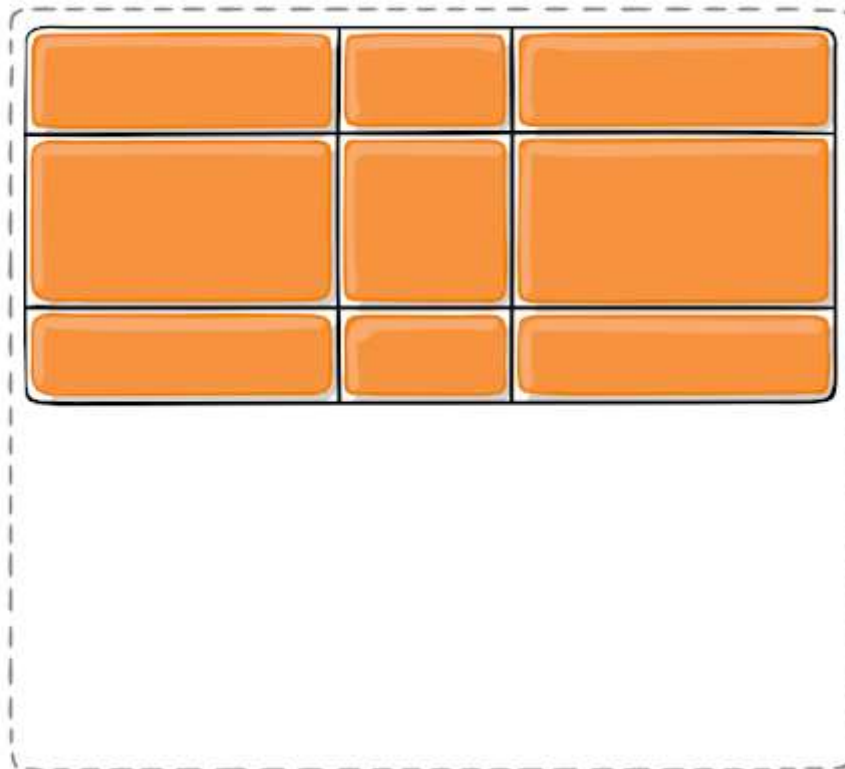
space-evenly：首尾的空間，同為行列間間距值



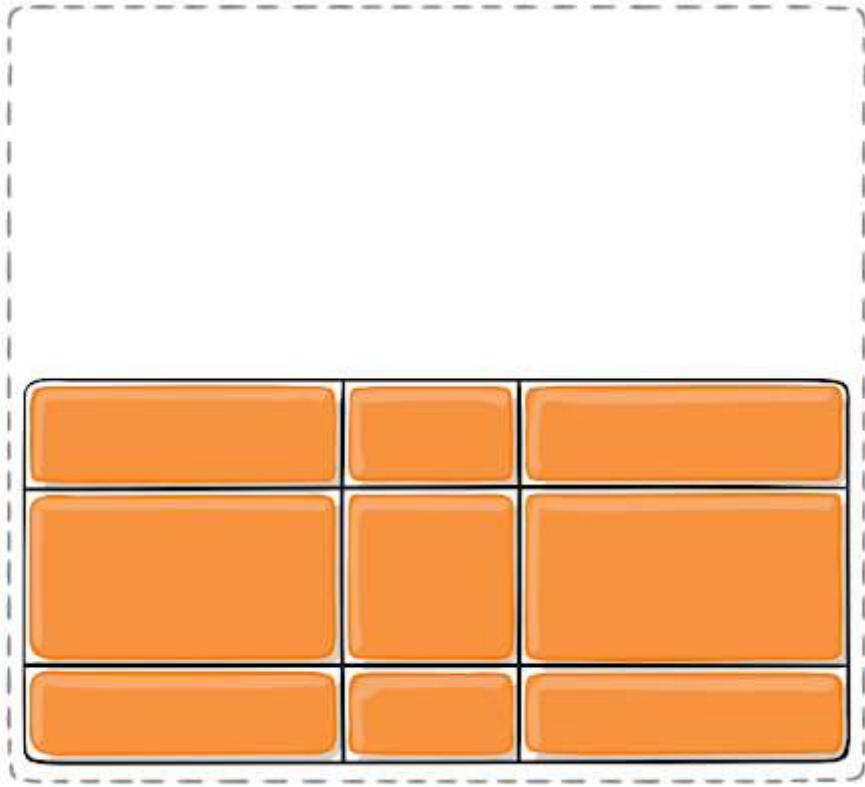
container :

justify-content: space-evenly;

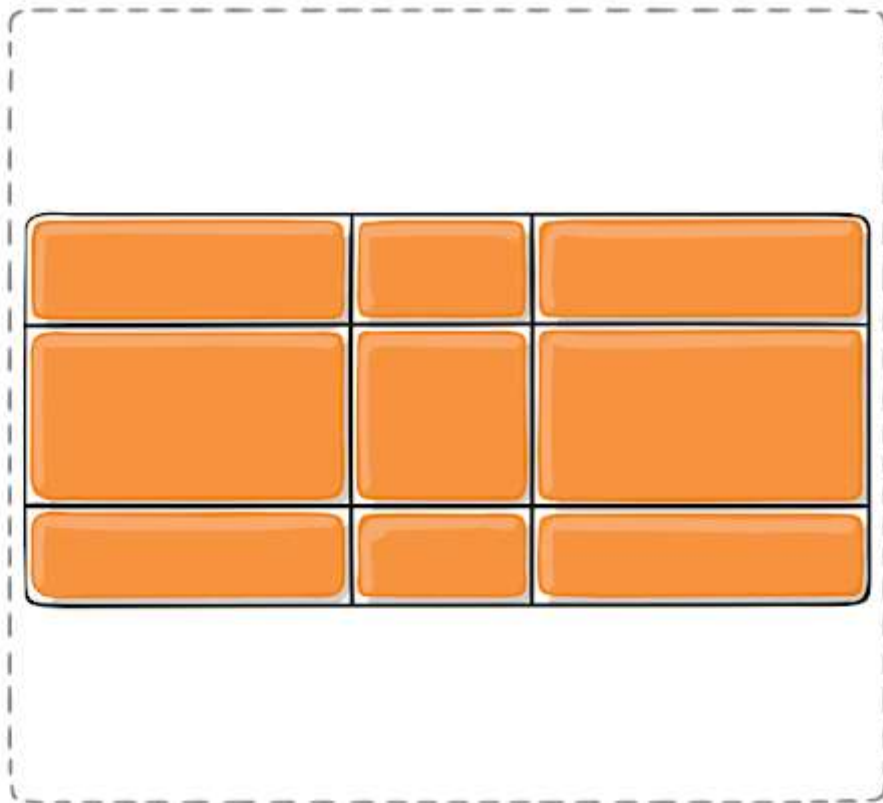
align-content
start



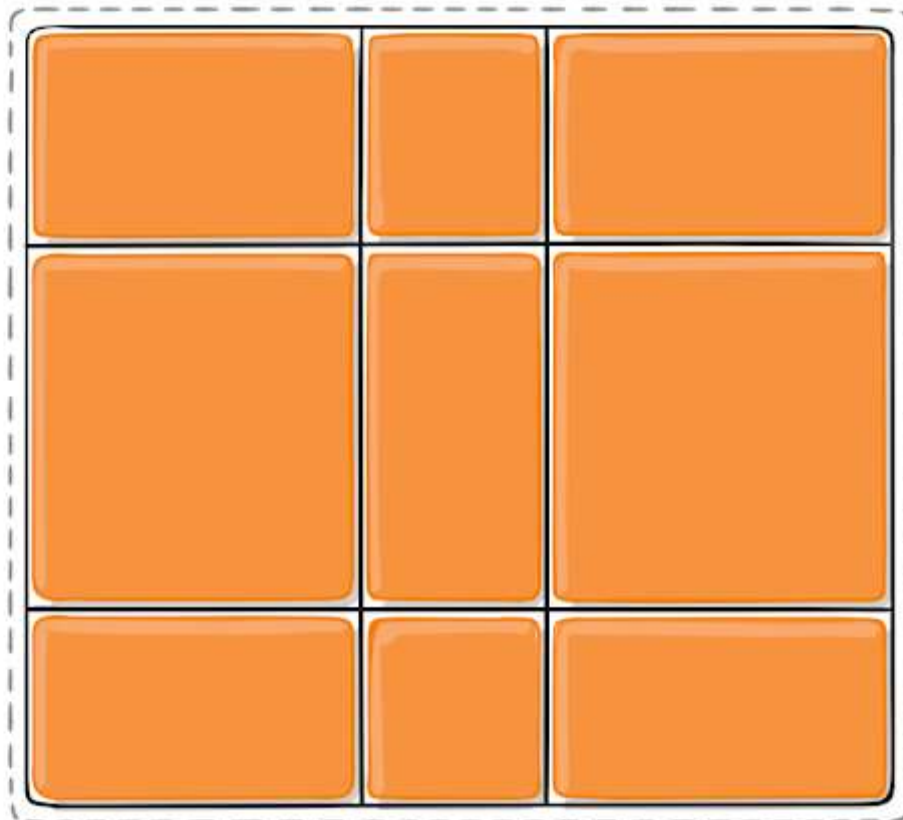
end



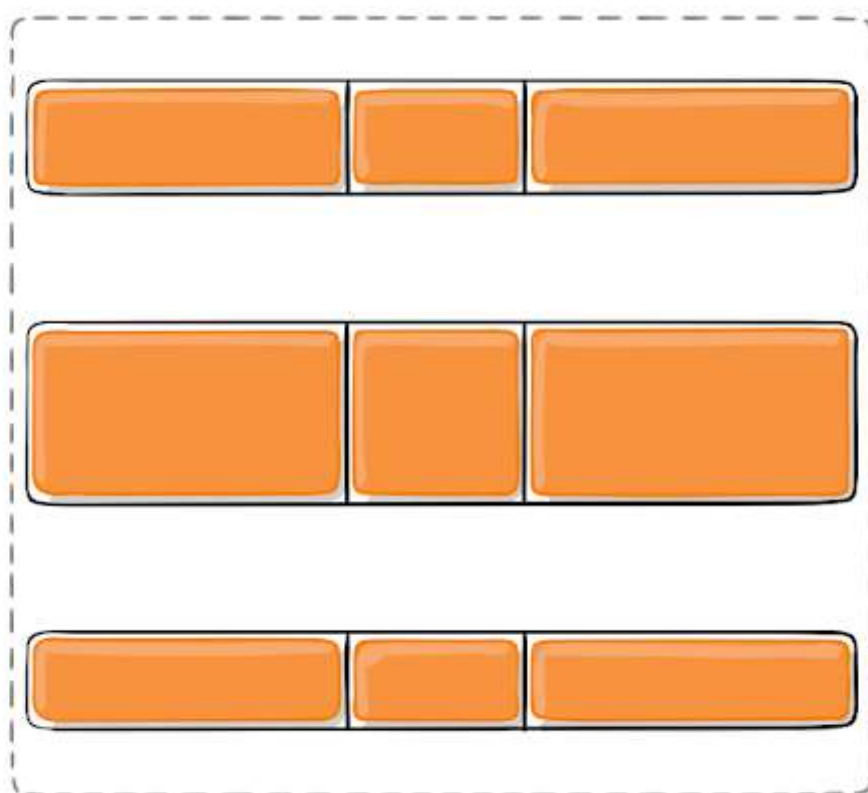
center



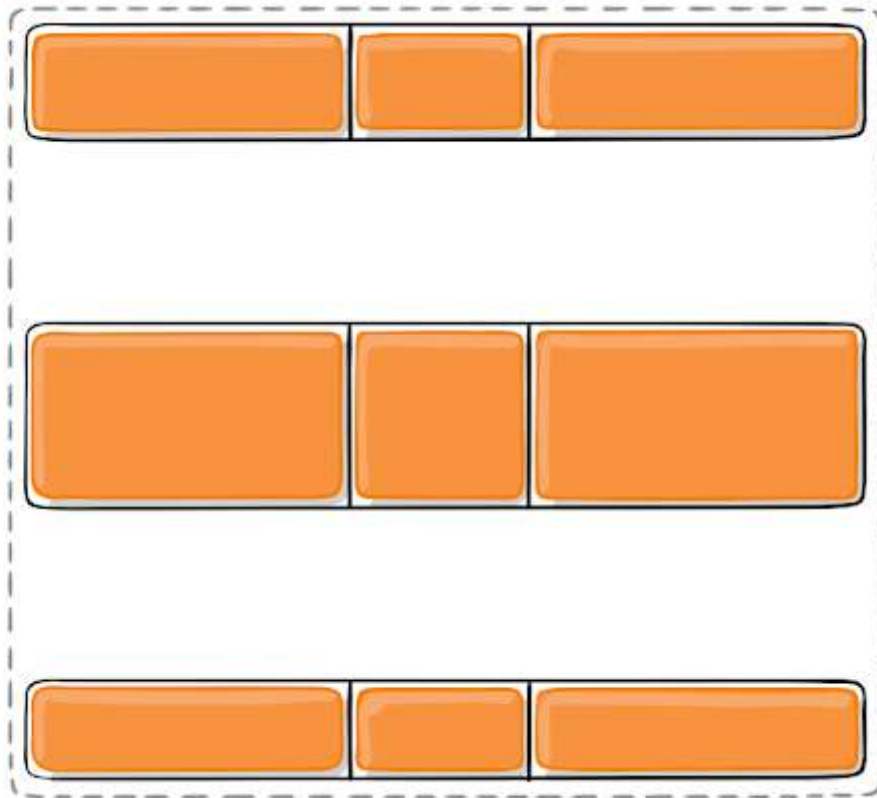
stretch



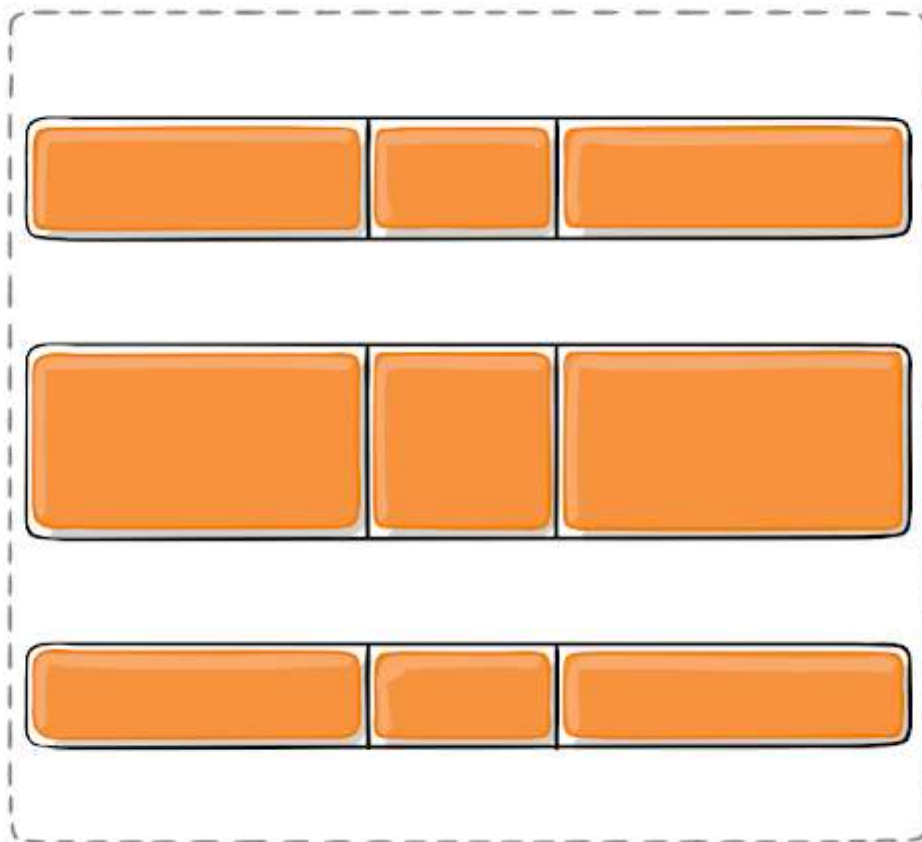
space-around : 首尾的空間，僅為行列間間距值的一半



space-between : 首尾的空間，無行列間間距值



space-evenly : 首尾的空間，同為行列間間距值



container :

```
align-content: space-between;
```

place-content

當然這邊也有一個整合的屬性

container :

```
/* <align-content> / <justify-content> */  
place-content: space-around;
```

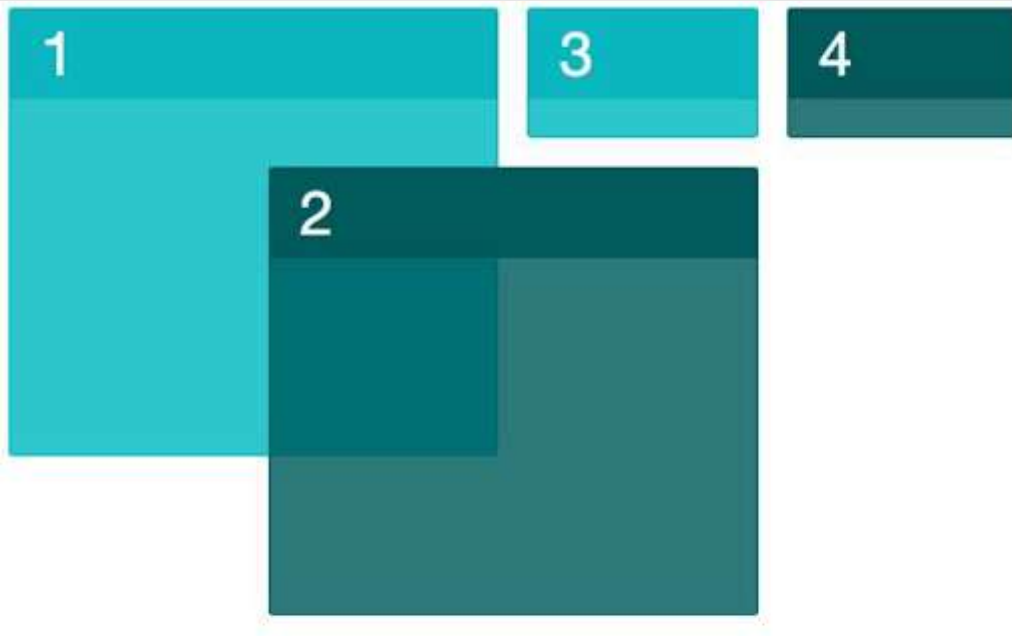
Even More

還有一些屬性，個人認為沒那麼實用，或是只是我還沒想到應用方式，列在下方也可以去搜尋看看。

```
grid-auto-columns  
grid-auto-rows  
grid-auto-flow
```

Overlap

Css Grid 方便的2維屬性，可以輕鬆地做到 **overlap** 的效果，只要設定好區域涵蓋的範圍。



Codepen 範例1

HTML

CSS

Result

EDIT ON
CODEPEN

```
body {  
  margin: 40px;  
}  
  
.container {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 100px 100px 100px;  
  background-color: #fff;  
  color: #444;  
}  
  
.box {  
  background-color: #444;  
  color: #fff;  
  border-radius: 5px;  
  font-size: 150%;  
}
```

Resources1×0.5×0.25×Rerun

Codepen 範例2

Source :
Tutorial

- [Complete Guide Grid](#)

Image

- [Layout Perfection with CSS Grid and Flexbox Fallbacks](#)
- [CSS Grid Changes Everything About Web Layouts: WordCamp Europe 2017](#)

Tags

CSS

Tutorial

Web

#Partical Weng



Facebook



Twitter



Partical Weng

Cyberpunk極致愛好者. 追求著腦動大開創意科技, 以製作出真 . 立體投影為目標.



You might like

[閱讀更多](#)

Css Grid 概念介紹及使用教學

🕒 October 06, 2019

3 留言



Ho.Chun

2021/2/19 下午4:19

很清楚! 感謝您

回覆



匿名

2021/5/25 下午1:08

good !

回覆



匿名

2021/5/25 下午1:11

作者已經移除這則留言。



輸入留言

< 較新的

較舊 >

ABOUT US

技術教學，科技觀點，社會觀察