



HTML



CSS

CSS

設定

- 使用 style 屬性

```
<label style="color: orange; font-weight: bold;">Hello, World!</label>
```

- 使用 class 屬性

```
<style>  
  .font-attribute {  
    color: green;  
    font-weight: bold;  
  }  
</style>  
<body>  
<label class="font-attribute">Hello, World!</label>  
</body>
```

文字

- 常見的文字設定

```
color: rgb(59, 119, 133);    /* 前景顏色 */
background-color: azure;      /* 背景顏色 */
font-weight: bold;           /* 粗體 */
font-size: 20px;              /* 字體大小 */
font-family: Arial, Helvetica, sans-serif; /* 字型 */
text-align: right;            /* 對齊 */
```

- 字型可以設定多個，從左邊開始，瀏覽器找不到就下一個，都找不到就用預設的
- 文字如果是在 inline 類型的標籤中（例如<label>），對齊無效

邊線

- 可以為標籤加上邊線

```
border-width: 1px;      /* 線條粗細 */  
border-radius: 5px;     /* 圓角 */  
border-style: solid;    /* 線條種類 */  
border-color: gold;     /* 線條顏色 */
```



Hello, World!

- 可簡寫，無順序，表示 border-width、border-style、border-color

```
border: 3px dashed burlywood;
```

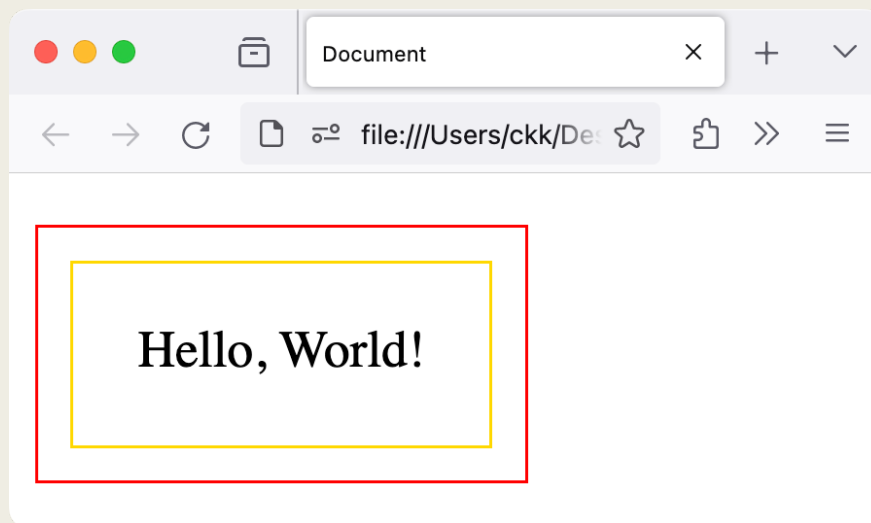


Hello, World!

距離

- Padding：元件內容與元件邊界距離 => 黃線與黑字間的距離。
- Margin：元件邊界往外擴增多少距離 => 黃線與紅線間的距離

```
<div>
  <label>Hello, World!</label>
</div>
```



```
<style>
label {
  display: inline-block;
  border: 1px gold solid;
  padding: 20px;
  margin: 10px;
}
div {
  display: inline-block;
  border: 1px red solid;
}
</style>
```

單位

- CSS 的度量單位種類繁多，螢幕顯示使用三種即可
- 絕對大小：px
 - 非實際 pixel，依裝置解析度調整
- 相對大小：em
 - 根據父元件的 font-size 為倍率基準
 - 若沒有任何父元件設定 font-size，最後以瀏覽器為基準，通常為 16px
- 相對大小：%
 - 依原尺寸大小按比例調整

```

```

```
<style>
.parent {
    font-size: 20px;
}

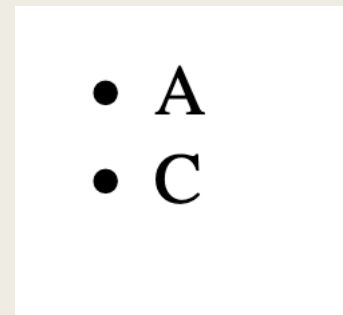
.child {
    font-size: 1.0em;
}
</style>
```

隱藏元件

■ display: none

- 整個元件從元件樹 (DOM tree) 中移除，相當於網頁上沒有這個元件

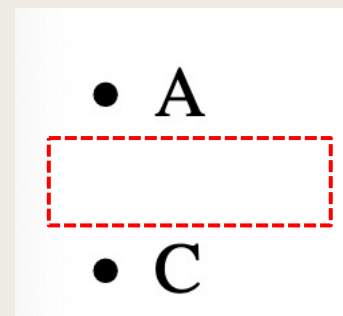
```
<ul>  
  <li>A</li>  
  <li style="display: none">B</li>  
  <li>C</li>  
</ul>
```



■ visibility: hidden

- 隱藏元件。元件還是佔據網頁上的空間，只是看不見而已

```
<ul>  
  <li>A</li>  
  <li style="visibility: hidden">B</li>  
  <li>C</li>  
</ul>
```



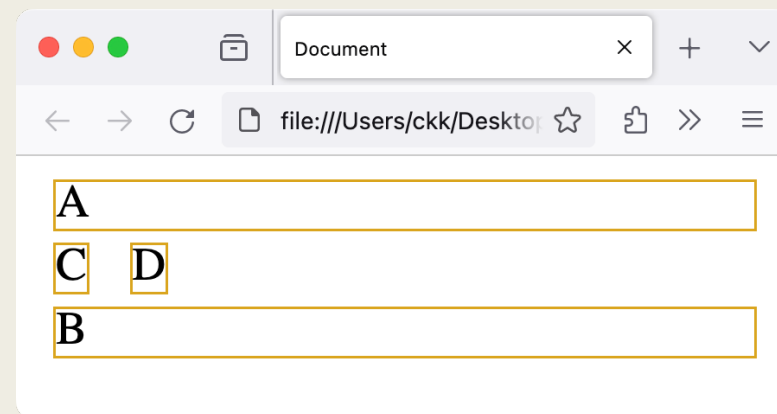
Display

■ 元件的顯示模式有兩種

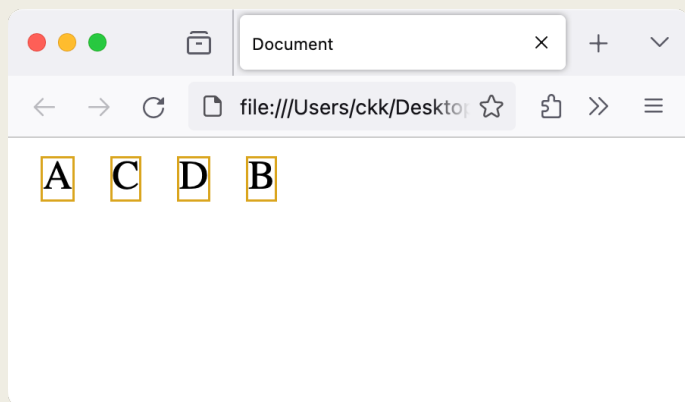
- Block：元件寬度佔滿整個父元件，可用 width 與 height 改大小。例如
`<p><div><hr>`
- Inline：元件寬度依據內容自動調整，無法透過 width 與 height 調整大小。例如
`<label><a>`

```
div, span {  
    margin: 5px;  
    border: 1px goldenrod solid;  
}
```

```
<div>A</div>  
<span>C</span>  
<span>D</span>  
<div>B</div>
```

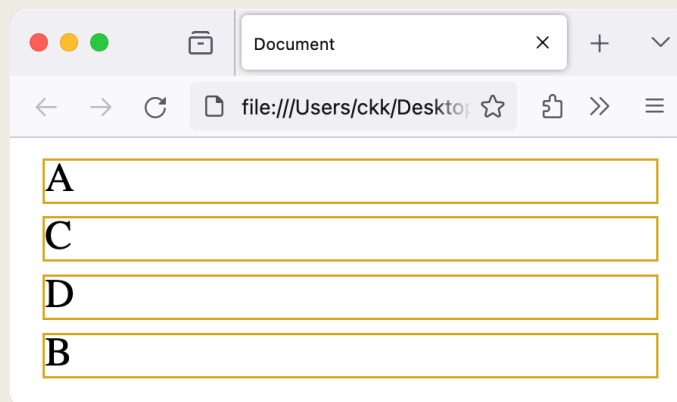


調整 Display



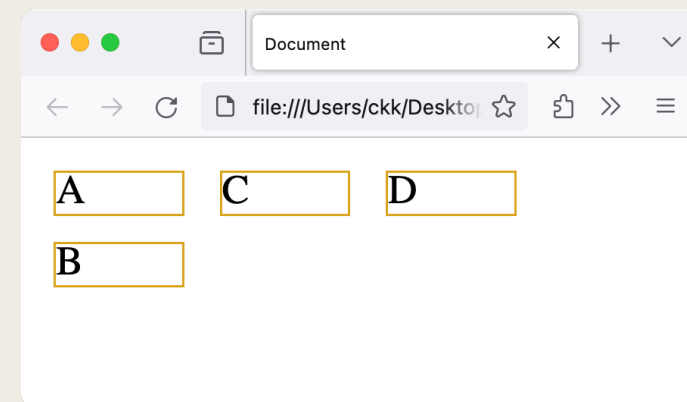
/* 一行放多個元素，不可調大小 */

display: inline;



/* 一行放一個元素，可調大小 */

display: block;



/* 一行放多個元素，可調大小 */

display: inline-block;
width: 50px;

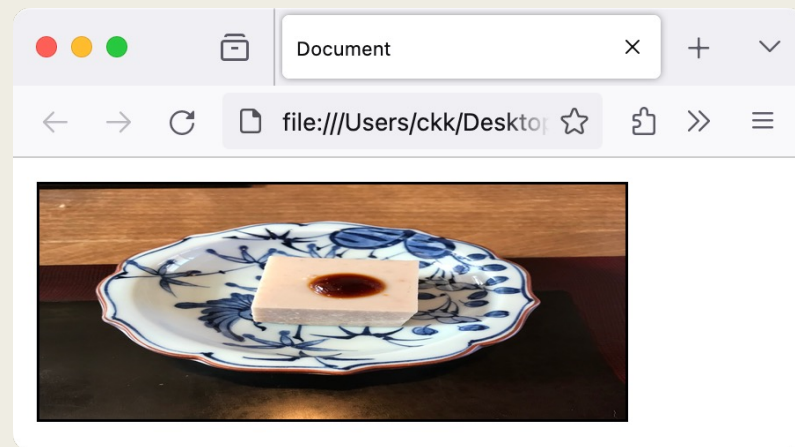
圖片

- 當圖片大小與 大小不一樣時，如何將圖片「塞」到 img 標籤中
- 預設為塞滿，圖片變形

```
img {  
    width: 200px;  
    height: 80px;  
    border: 1px black solid;  
}
```

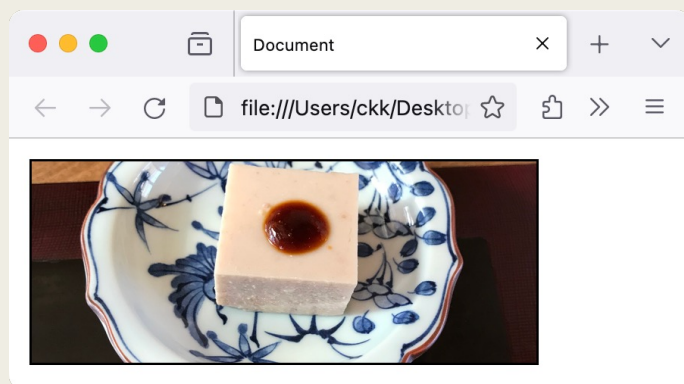
```

```

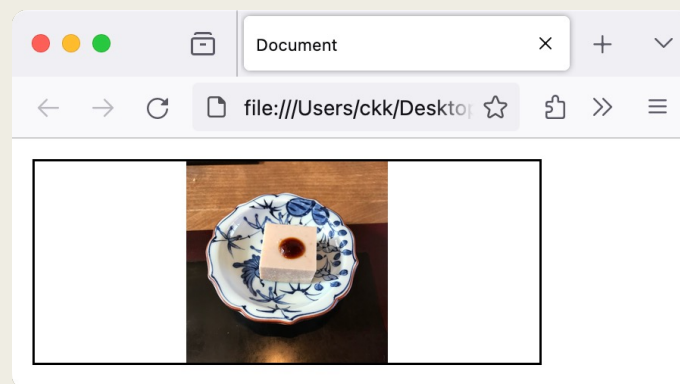


Object-fit：圖片縮放方式

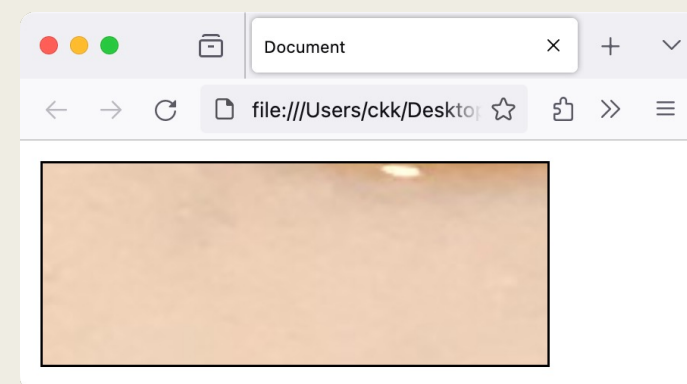
- 預設為 fill，塞滿父元件



`object-fit: cover;`



`object-fit: contain;`



`/* 原尺寸 */
object-fit: none;`

背景圖片

- 整個網頁設定背景圖片，當然也可以在特定的標籤上設定背景圖片

```
body {  
    background-image: url("bg.jpeg");  
    background-size: cover;  
}
```

-image 可以省略

選取器

選取標籤

- 只要是 <p> 標籤就套用字體顏色為橘色

```
p {  
    color: orange;  
}
```

- 只有在 <div> 標籤裡面的 <p> 標籤才會被選取

```
div p {  
    color: orange;  
}
```

```
div > p {  
    color: orange;  
}
```

只有第一層 <p> 才會選取

選取 id

- 例如，只有 id 為 id1 的標籤才會被選取

```
#id1 {  
    color: orange;  
}
```

- 範例

```
<label id="p1">Hello, World!</label>
```

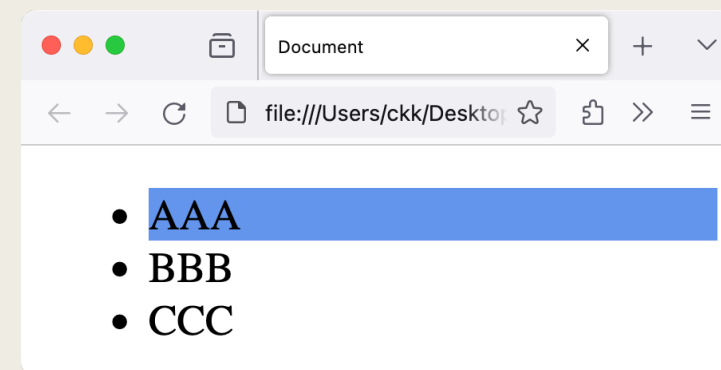
選取 class

■ 選取 class

```
.highlight {  
    background-color: cornflowerblue;  
}
```

■ 範例

```
<ul>  
    <li class="highlight">AAA</li>  
    <li>BBB</li>  
    <li>CCC</li>  
</ul>
```



全域選取與全域變數

- 相當於選取 <html> 標籤，但 :root 還多了設定全域變數功能

```
:root {  
    color: cornflowerblue;  
}
```

- 全域變數。變數一定要兩個「-」開頭

```
:root {  
    --theme-color: lightgrey;  
    --theme-background-color: darkblue;  
}  
  
.highlight {  
    background-color: var(--theme-background-color);  
    color: var(--theme-color);  
}
```

區域變數

- 在父元件中設定變數，使其成為所有子元件的區域變數

```
#content {  
    --content-color: green;  
}  
  
#content * {  
    color: var(--content-color);  
}
```

選取子元素

- 選取全部子元素

```
#parent * {  
    color: green;  
}
```

- 選取第 n 個子元素

```
#parent :nth-child(2) {  
    color: green;  
}
```

```
<div id="parent">  
    <label>A</label>  
    <label>B</label>  
    <label>C</label>  
</div>
```

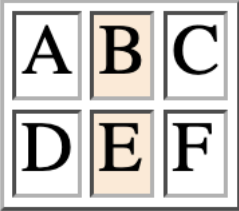


綠色

「:」 虛擬類別 (偽選擇器)

- :nth-child(n) 表示選取第 n 個子元素，也可以 (4n+1)，4為步進1為從第幾個開始
- :first-child 表示選取第一個子元素
- :last-child 表示選取最後一個子元素
- 例如，選取表格中的第二欄

```
table td:nth-child(2) {  
    background-color: antiquewhite;  
}
```



A	B	C
D	E	F

- 試試盤旋 (hover)

```
table td:hover {  
    background-color: antiquewhite;  
}
```
- 多如牛毛：<https://developer.mozilla.org/zh-TW/docs/Web/CSS/Pseudo-classes>

「::」偽元素

- 選取到的前、後加上特定內容，例如

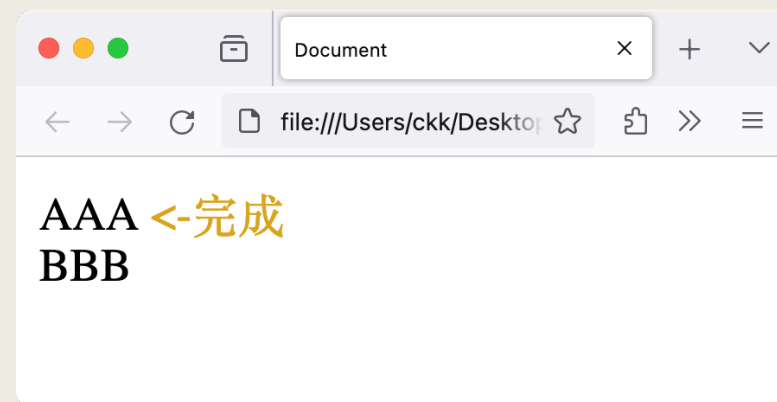
```
#item1::after {  
  content: " <-完成";  
  color: goldenrod;  
}
```

改 before
就是前面

- 範例

```
<div id="item1">AAA</div>  
<div id="item2">BBB</div>
```

- 還有那些：<https://developer.mozilla.org/zh-CN/docs/Web/CSS/Pseudo-elements>

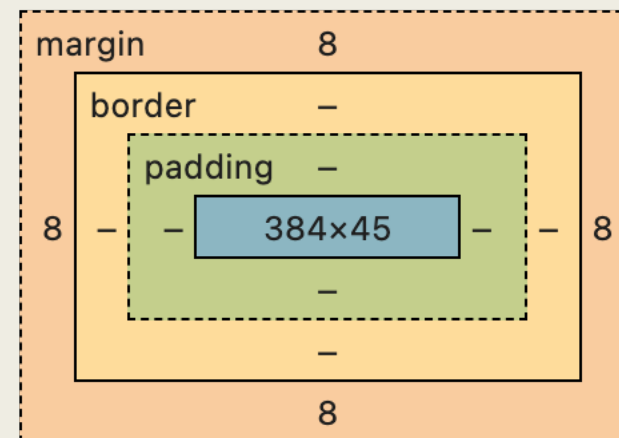


位置

盒子模型

- 每個元素都視為一個 box
- 藍色區域：標籤內容
- Padding：內容與邊線間的距離
- Border：邊線（包含種類、寬度）
- Margin：與旁邊 box 的距離

在瀏覽器的開發人員工具
找到下面這張圖



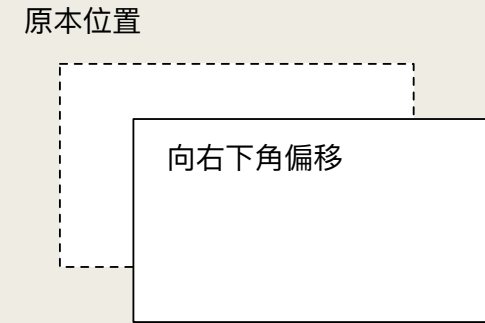
與盒子模型有關的 CSS

- 從盒子模型中找到下列資料，並試著從盒子模型中直接修改這些值

```
#myid {  
    margin: 50px;  
    padding: 5px;  
    border: 10px cyan solid;  
    width: 150px;  
    height: 100px;  
    background-color: burlywood;  
}
```


調整位置

- 透過 position 調整元素所在位置，有五種設定
- static：預設，此時 left、top、right、bottom 失效
- relative：相對於該元件原本的位置
- absolute：以文件或父元件左上角為原點的絕對位置（下頁）
- fixed：該元件獨立於排版系統外，並以頁面左上角座標為原點，且忽略捲軸作用
- sticky：當 scroll bar 捲動且元件捲到最上方時，永遠黏在最上方，意思是 top 值即為最小值



```
position: relative;  
left: 10px;  
top: 10px;
```

```
#parent {  
  position: relative;  
  border: 1px black solid;  
  width: 100px;  
  height: 600px;  
}
```

```
#div1 {  
  position: sticky;  
  left: 0px;  
  top: 0px;  
  background-color: burlywood;  
}
```

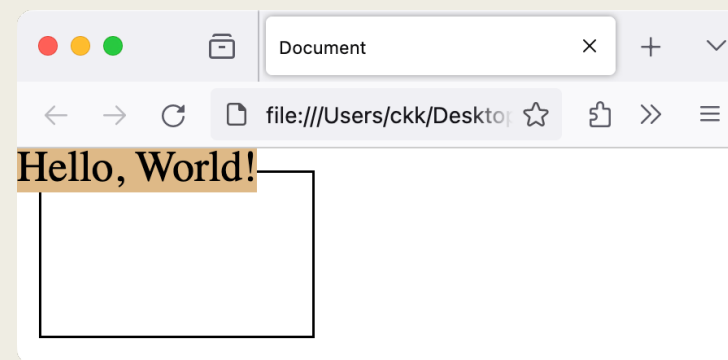
Position: absolute

- 父元件分成兩種狀態：父元件設定為 static

```
#parent {  
    position: static; /* default */  
    border: 1px black solid;  
    width: 100px;  
    height: 60px;  
}
```

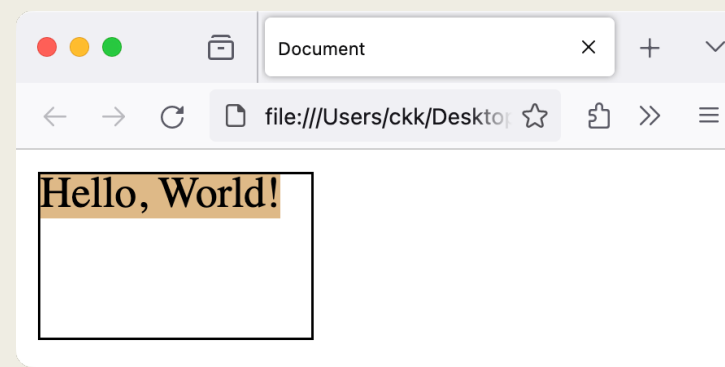
```
#div1 {  
    position: absolute;  
    left: 0px;  
    top: 0px;  
    background-color: burlywood;  
}
```

```
<div id="parent">  
    <div id="div1">Hello, World!</div>  
</div>
```



- 父元件設定為 non-static

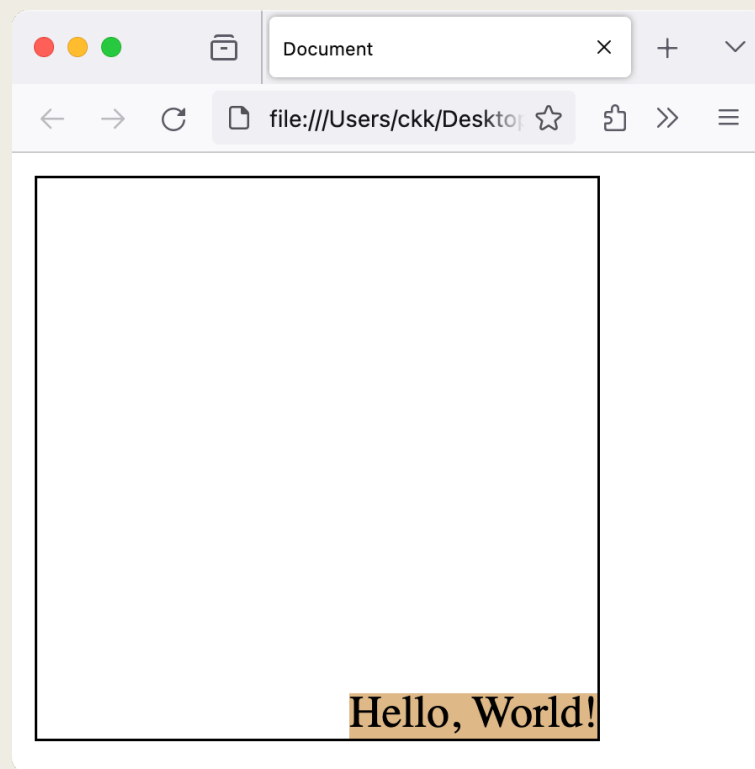
```
position: relative;
```



貼齊

■ 向右下角貼齊

```
.container {  
    position: relative;  
    width: 200px;  
    height: 200px;  
    border: 1px solid black;  
}  
.container label {  
    position: absolute;  
    right: 0px;  
    bottom: 0px;  
    background-color: burlywood;  
}
```

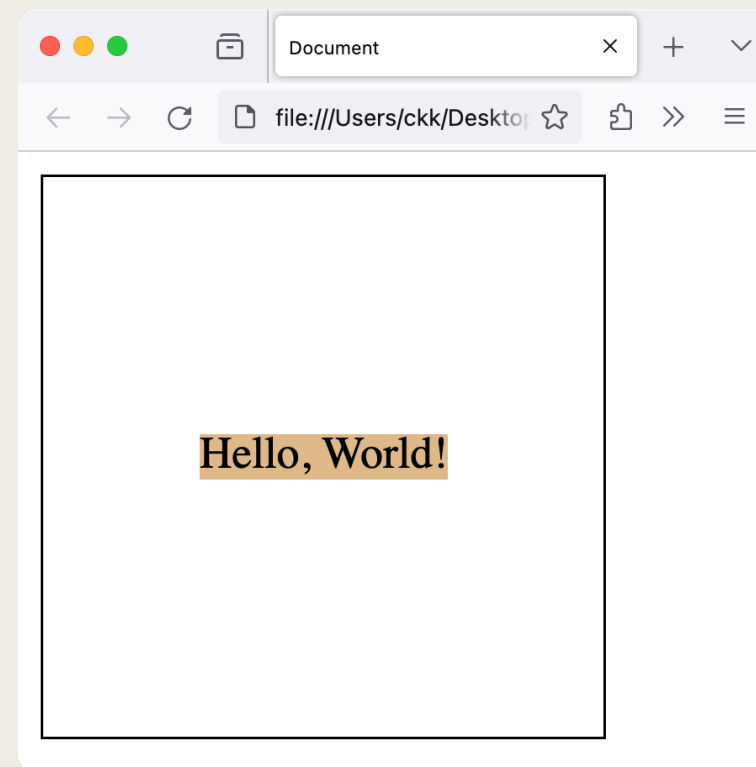


置中對齊

■ 水平置中、垂直置中

```
.container label {  
    position: absolute;  
    left: 50%;  
    top: 50%;  
    transform: translate(-50%, -50%);  
    background-color: burlywood;  
}
```

向左上角移動自身
長寬的一半

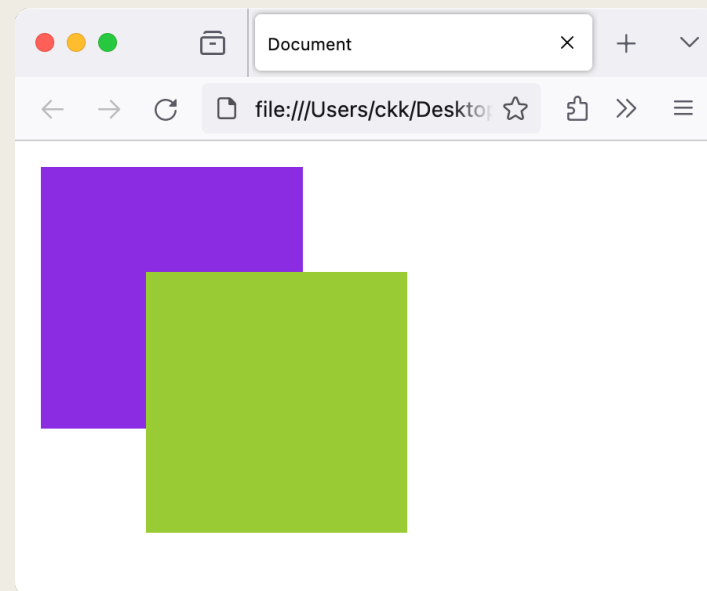
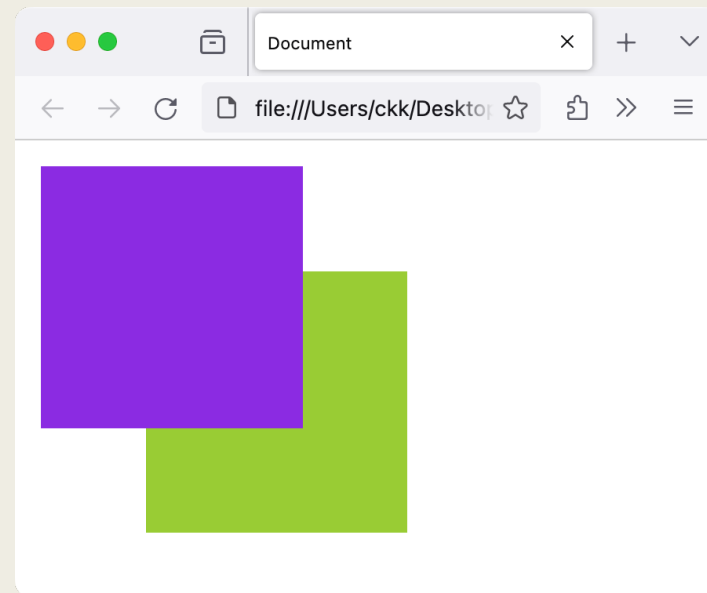


層級：z-index

```
#div1 {  
  position: absolute;  
  z-index: 20;  
  left: 10px;  
  top: 10px;  
  width: 100px;  
  height: 100px;  
  background-color: blueviolet;  
}
```

數字越小層級越低，
最小0，預設auto

```
#div2 {  
  position: absolute;  
  left: 50px;  
  top: 50px;  
  width: 100px;  
  height: 100px;  
  background-color: yellowgreen;  
}
```

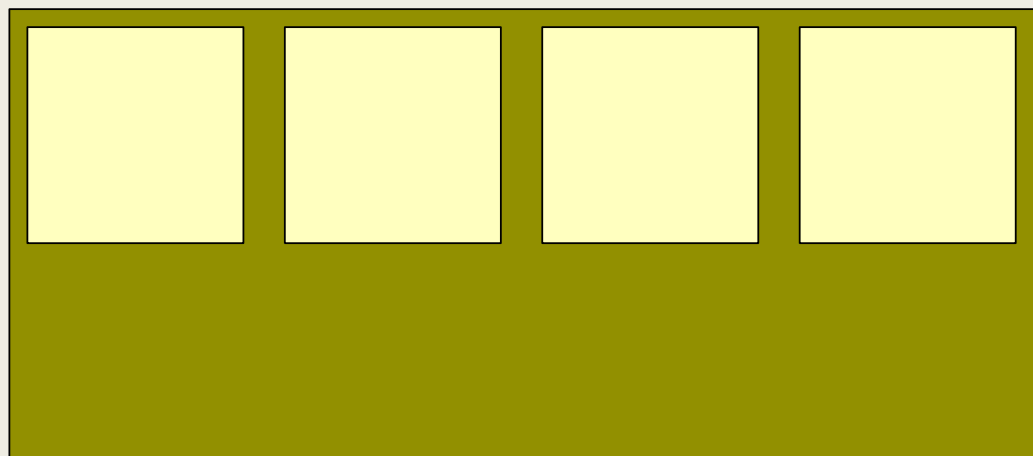


FLEX

說明

- Flex排版模式是一種將<div>當成容器元件，並且透過排版屬性來設定內容元件要怎麼排列的一種排版方式，類似 Stack View。

容器元件



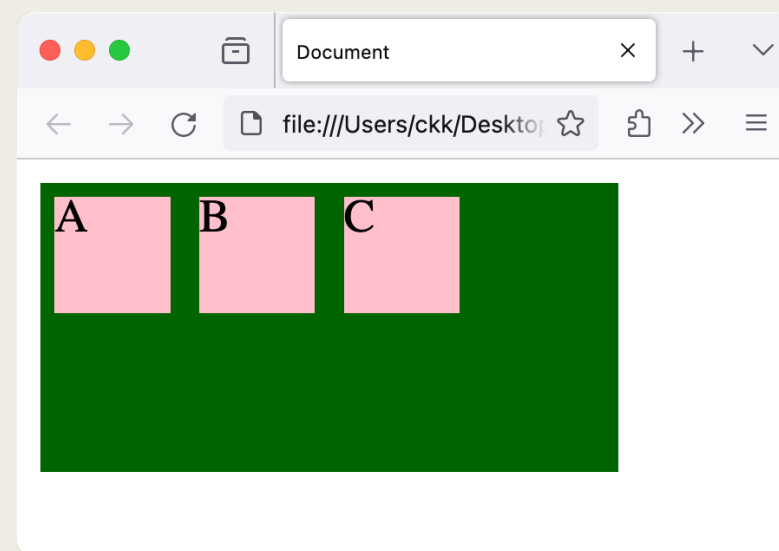
基本用法

```
#container {  
  display: flex;  
  width: 200px;  
  height: 100px;  
  background-color: darkgreen;  
}
```

```
#container div {  
  width: 40px;  
  height: 40px;  
  margin: 5px;  
  background-color: pink;  
}
```

```
<div id="container">  
  <div>A</div>  
  <div>B</div>  
  <div>C</div>  
</div>
```

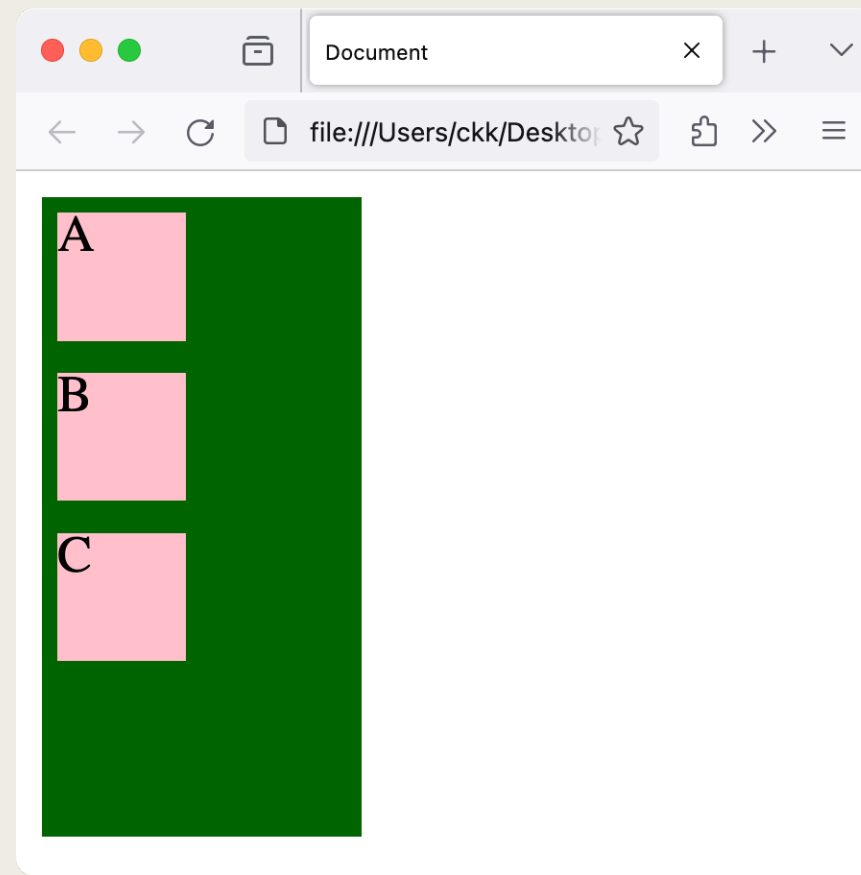
試試加一堆後會
發生什麼事情？



垂直排列

```
#container {  
  display: flex;  
  flex-direction: column;  
  width: 100px;  
  height: 200px;  
  background-color: darkgreen;  
}
```

預設：row

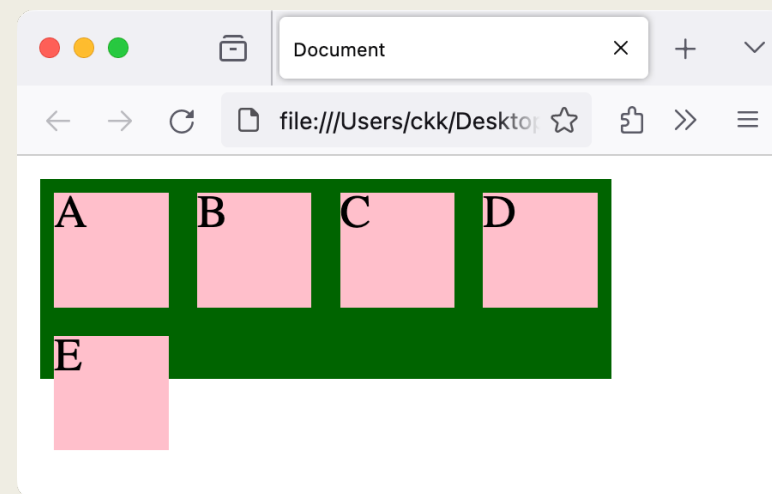


自動換行

- 讓超過容器邊界的元件自動換到下一行或下一欄

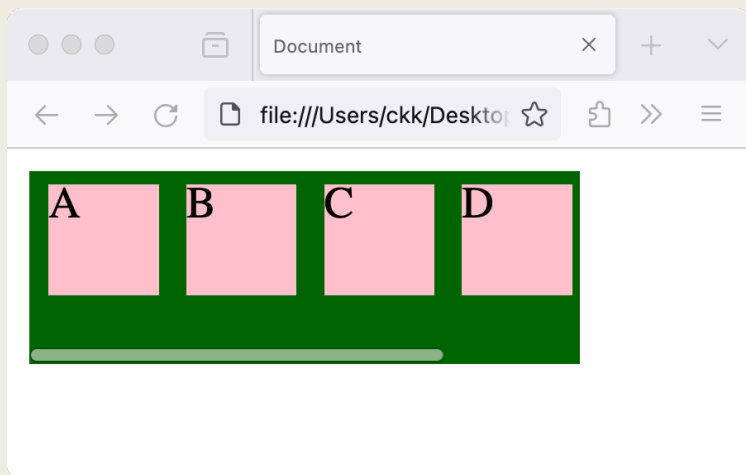
```
#container {  
  display: flex;  
  flex-wrap: wrap;  
  width: 200px;  
  height: 70px;  
  background-color: darkgreen;  
}
```

- 若未設定 flex-wrap，這時子元件寬度會重新調整，盡可能全部擠進 container 中



捲軸

- 超過又不想移到下一行，也不想縮小寬度或高度，可以在 container 中設定 `overflow: auto`，子元素中設定 `flex-shrink: 0`，這樣就會出現捲軸
- `overflow` 會在 x 與 y 都出現捲軸，`overflow-x` 或 `overflow-y` 只讓 x 或 y 軸出現捲軸。



```
#container {  
    display: flex;  
    overflow: auto;  
    width: 200px;  
    height: 70px;  
    background-color: darkgreen;  
}
```

```
#container div {  
    flex-shrink: 0;  
    width: 40px;  
    height: 40px;  
    margin: 5px;  
    background-color: pink;  
}
```

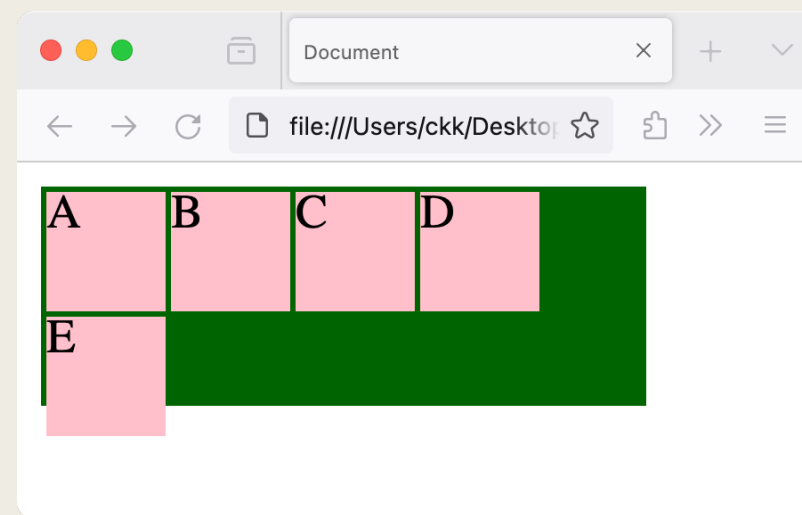
設定元件間距

- 在容器元件中設定內容的間距

```
column-gap: 2px;  
row-gap: 2px;
```

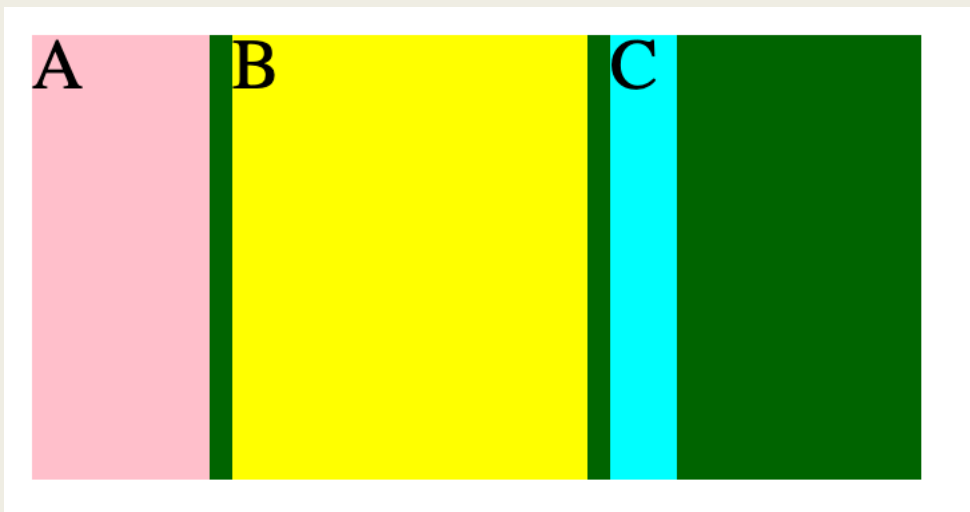
- 範例

```
#container {  
    display: flex;  
    flex-wrap: wrap;  
    padding: 2px;  
    column-gap: 2px;  
    row-gap: 2px;  
    width: 200px;  
    height: 70px;  
    background-color: darkgreen;  
}
```



剩餘空間分配

```
<div id="container">
  <div class="pink">A</div>
  <div class="yellow">B</div>
  <div class="cyan">C</div>
</div>
```



```
#container {
  display: flex;
  column-gap: 5px;
  width: 200px;
  height: 100px;
  background-color: darkgreen;
}
.pink {
  width: 40px;
  background-color: pink;
}
.yellow {
  width: 80px;
  background-color: yellow;
}
.cyan {
  width: 15px;
  background-color: cyan;
}
```

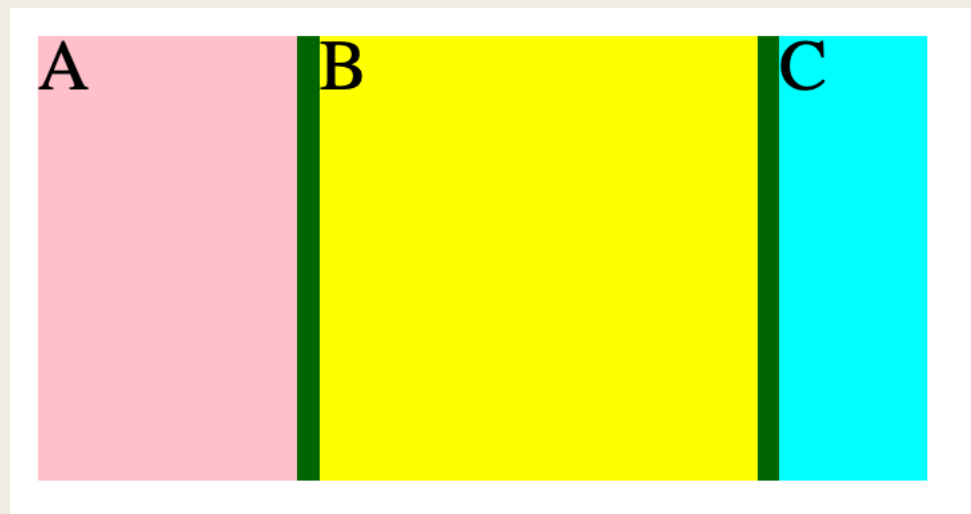
等分擴大

- 在子項目中加上 flex-grow 屬性，填入 1 表示剩餘空間平分給所有子元件

```
#container div {  
    flex-grow: 1;  
}
```

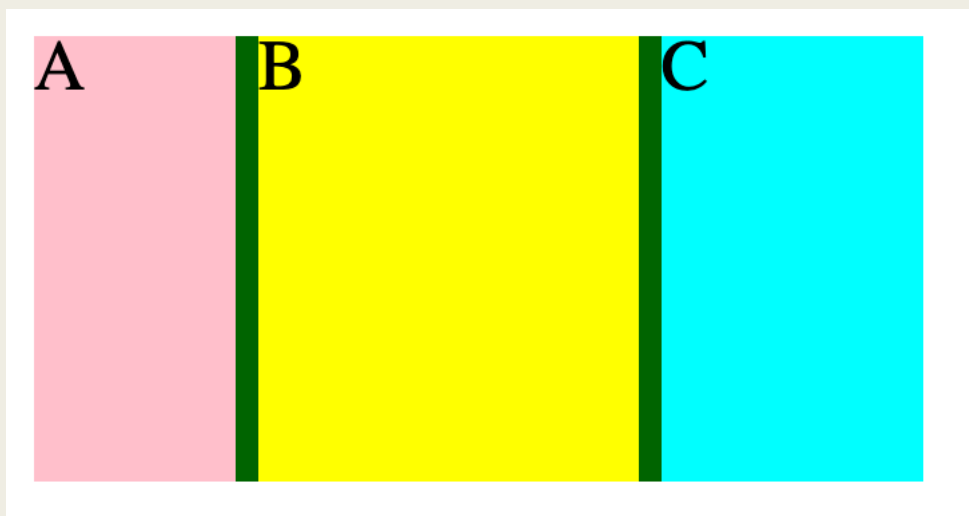
- 將剩餘空間等分成三份，然後給 A、B、C，最終
 - A 寬度 58.33
 - B 寬度 98.33
 - C 寬度 33.33

怎麼來的？



特定比例擴大

- 若希望 A 得到剩餘空間的 10%，B 得到 10%，C 得到 80%

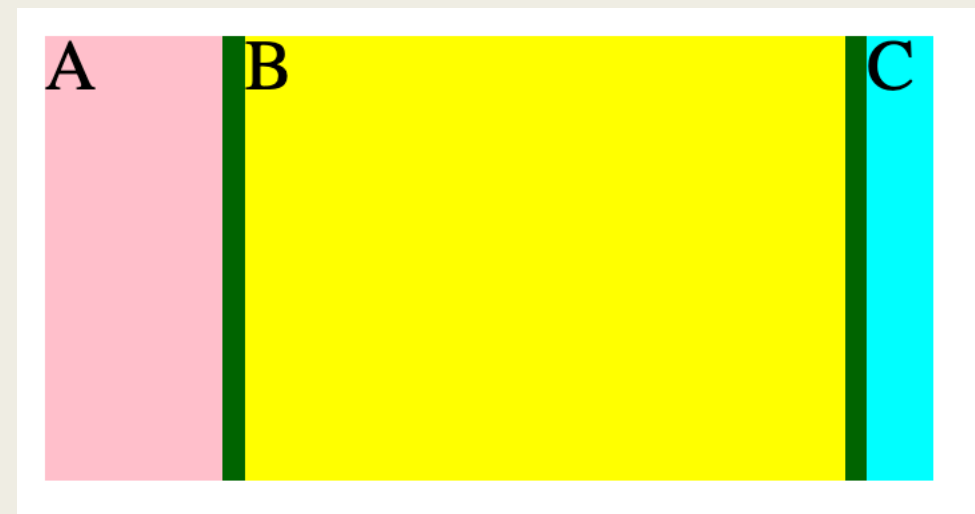


```
.pink {  
    flex-grow: 0.1;  
    width: 40px;  
    background-color: pink;  
}  
.yellow {  
    flex-grow: 0.1;  
    width: 80px;  
    background-color: yellow;  
}  
.cyan {  
    flex-grow: 0.8;  
    width: 15px;  
    background-color: cyan;  
}
```

全部給特定元件

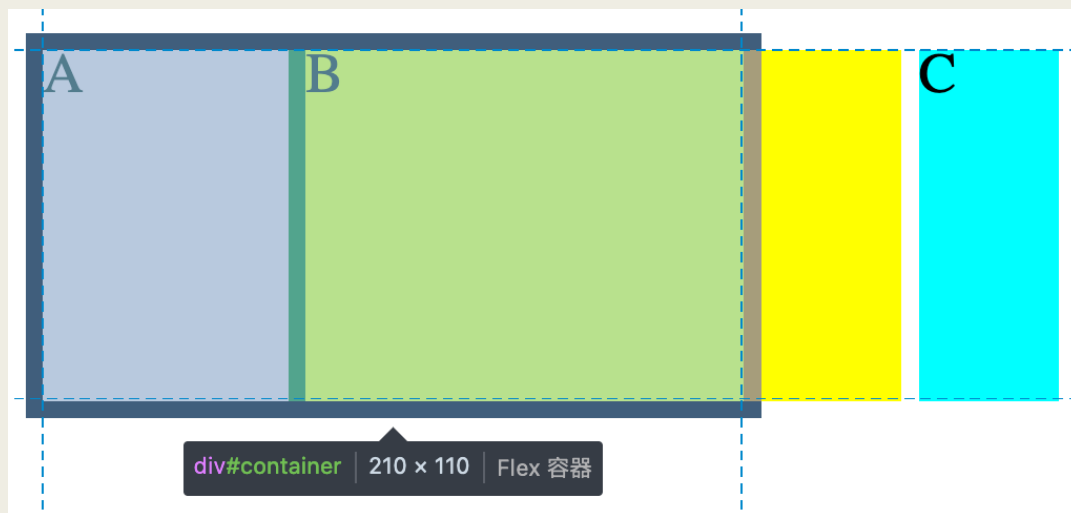
- 若希望 B 得到所有剩餘空間時，將 `flex-grow` 放到 B 即可

```
.yellow {  
    flex-grow: 1;  
    width: 80px;  
    background-color: yellow;  
}
```



超過縮小

- 當在子元件設定 flex-shrink 並填入 0，表示子元件寬度可以超過父元件寬度



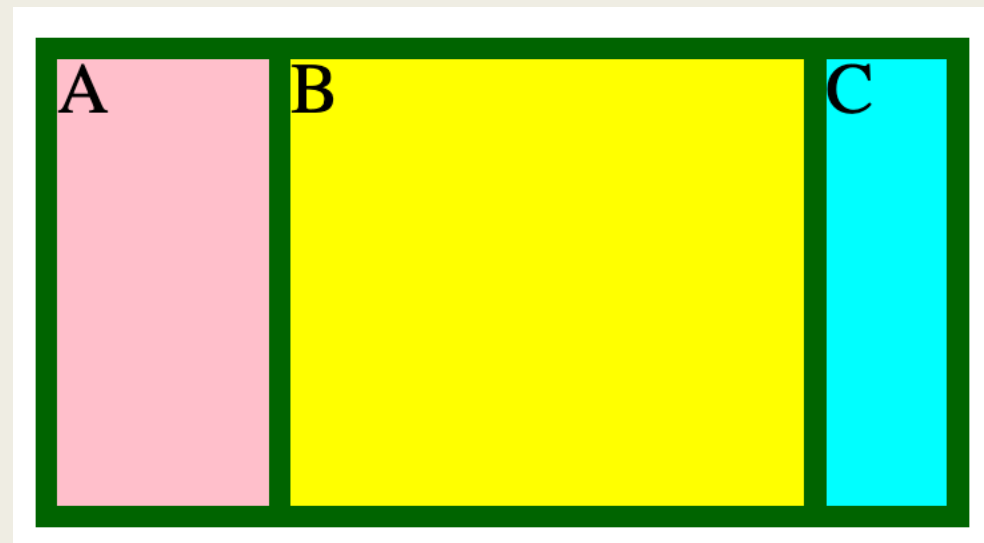
```
#container {  
  display: flex;  
  column-gap: 5px;  
  padding: 5px;  
  width: 200px;  
  height: 100px;  
  background-color: darkgreen;  
}  
  
#container div {  
  flex-shrink: 0  
}  
  
.pink {  
  width: 70px;  
  background-color: pink;  
}  
  
.yellow {  
  width: 170px;  
  background-color: yellow;  
}  
  
.cyan {  
  width: 40px;  
  background-color: cyan;  
}
```

等比例縮小

- 在子項目中加上 flex-shrink 屬性，填入 1 表示超過空間等比例縮小子元件

```
#container div {  
    flex-shrink: 1  
}
```

- 將超過空間等比例縮小
A、B、C，最終
 - A 寬度 47.5
 - B 寬度 115.35
 - C 寬度 27.15

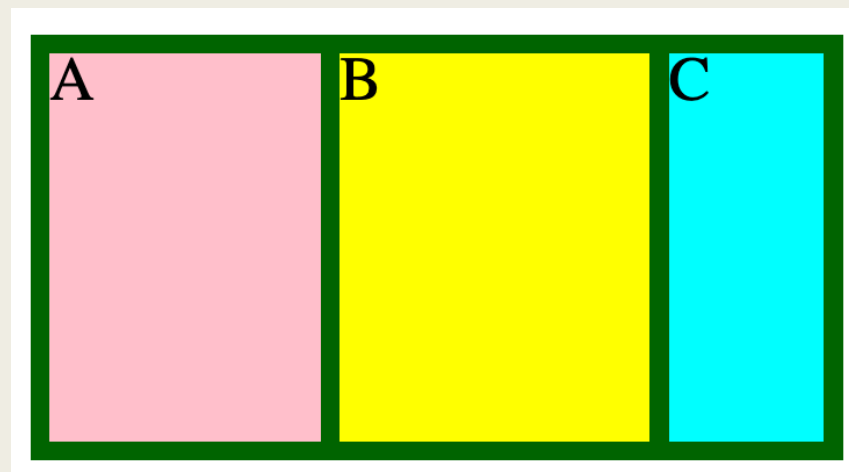


怎麼來的？

部分固定

- 若 A、C 寬度固定，剩餘空間全部給 B

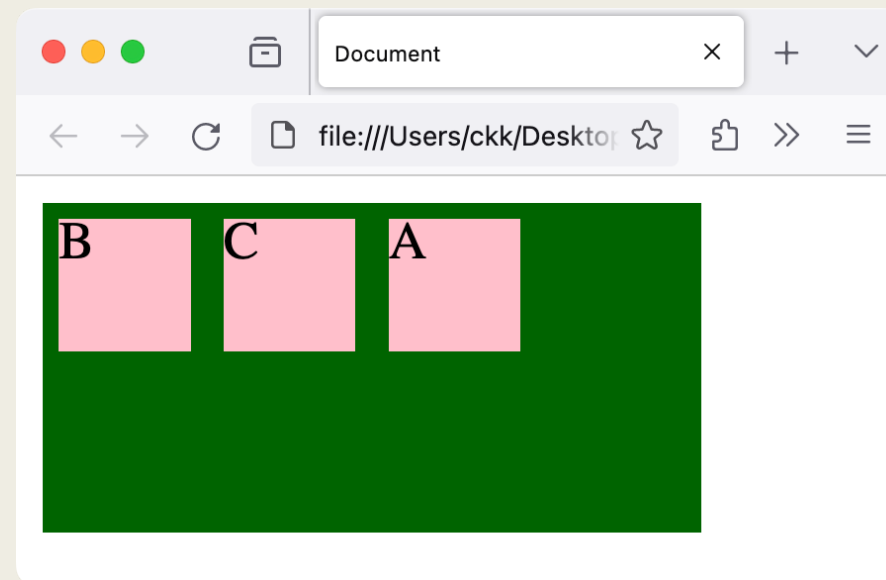
```
.pink {  
    flex-shrink: 0;  
    width: 70px;  
    background-color: pink;  
}  
.yellow {  
    width: 100%;  
    background-color: yellow;  
}  
.cyan {  
    flex-shrink: 0;  
    width: 40px;  
    background-color: cyan;  
}
```



調整順序

- 透過 order 調整排列順序
- Order 數字小的在前面，可以填負數

```
<div id="container">  
  <div style="order: 5">A</div>  
  <div style="order: 0">B</div>  
  <div style="order: 3">C</div>  
</div>
```



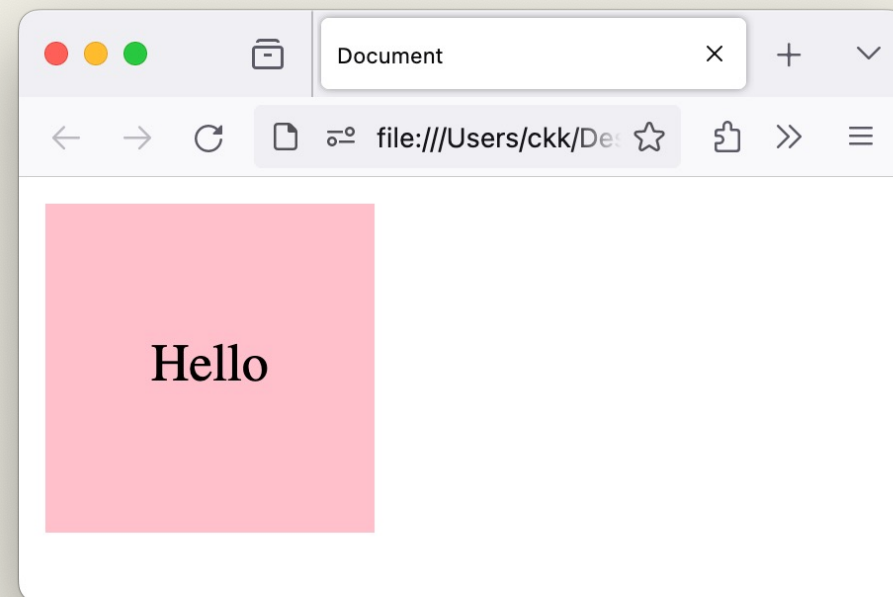
置中

- 使用 inline-flex 做水平與垂直置中
- HTML 標籤為

```
<label>Hello</label>
```

- CSS 為

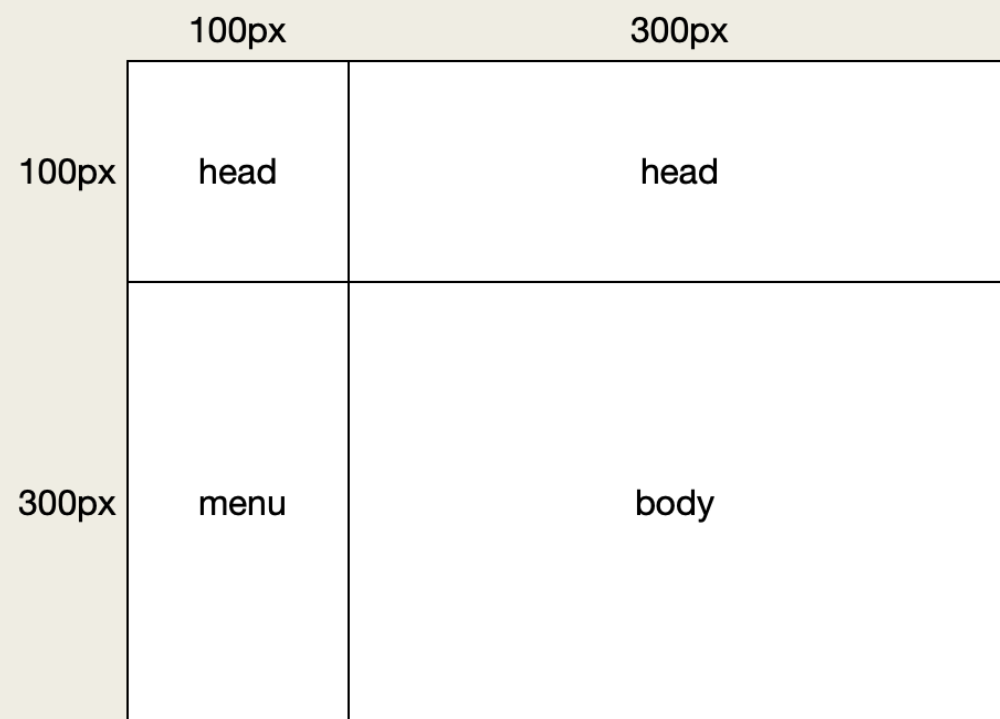
```
label {  
    display: inline-flex;  
    width: 100px;  
    height: 100px;  
    background-color: pink;  
    justify-content: center;  
    align-items: center;  
}
```



GRID

說明

- GRID 就是將畫面切割成像表格一樣的區塊，然後將資料放到各區塊中就可以了，基本概念跟使用表格來排版一樣

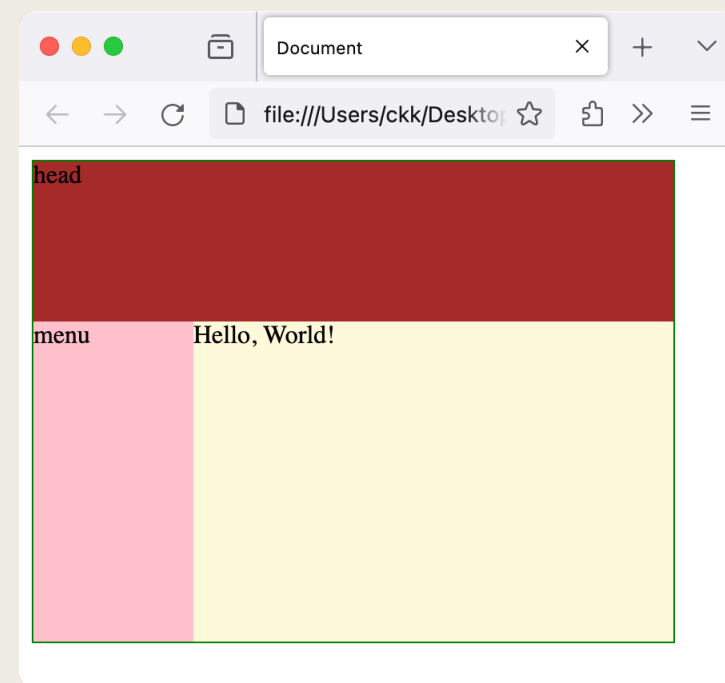


使用 Grid-Area

```
.container {  
  display: grid;  
  width: 400px;  
  height: 300px;  
  border: 1px solid green;  
  grid-template-columns: 100px auto;  
  grid-template-rows: 100px auto;  
  grid-template-areas:  
    "head head"  
    "menu body";  
}
```

```
.head {  
  grid-area: head;  
  background-color: brown;  
}
```

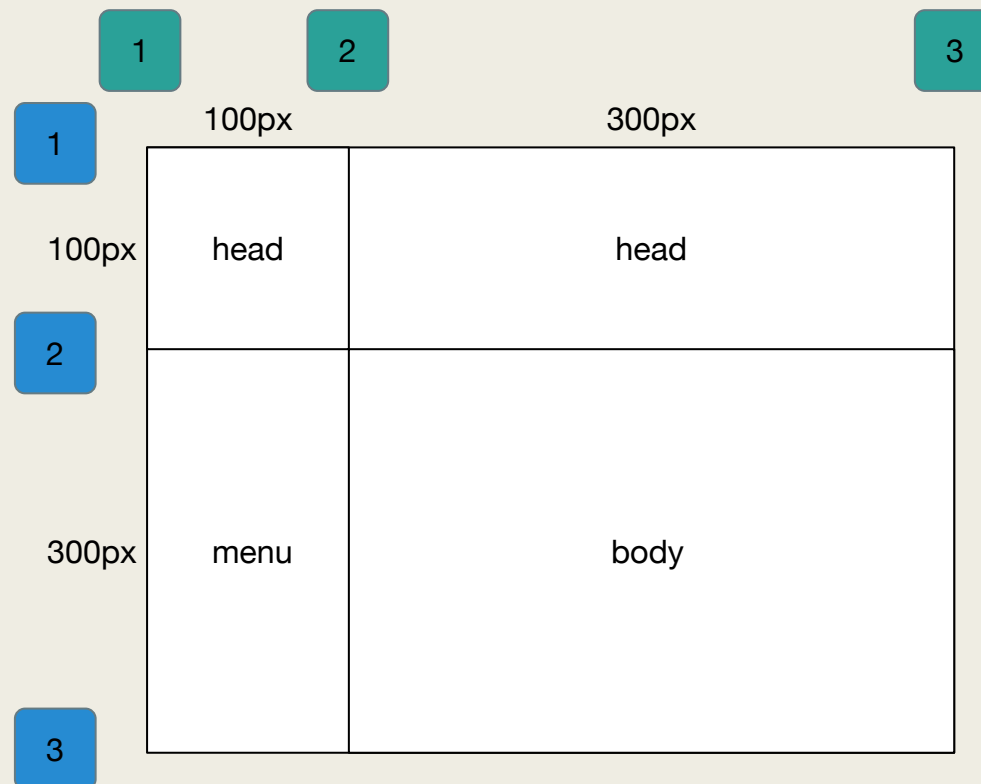
```
.menu {  
  grid-area: menu;  
  background-color: pink;  
}  
.body {  
  grid-area: body;  
  background-color: cornsilk;  
}
```



```
<div class="container">  
  <div class="head">head</div>  
  <div class="menu">menu</div>  
  <div class="body">Hello, World!</div>  
</div>
```


使用線條編號

```
.container {  
  display: grid;  
  width: 400px;  
  height: 300px;  
  border: 1px solid green;  
  grid-template-columns: 100px auto;  
  grid-template-rows: 100px auto;  
}  
.head {  
  grid-column: 1/3;  
  grid-row: 1/2;  
  background-color: brown;  
}  
.menu {  
  grid-column: 1/2;  
  grid-row: 2/3;  
  background-color: pink;  
}
```

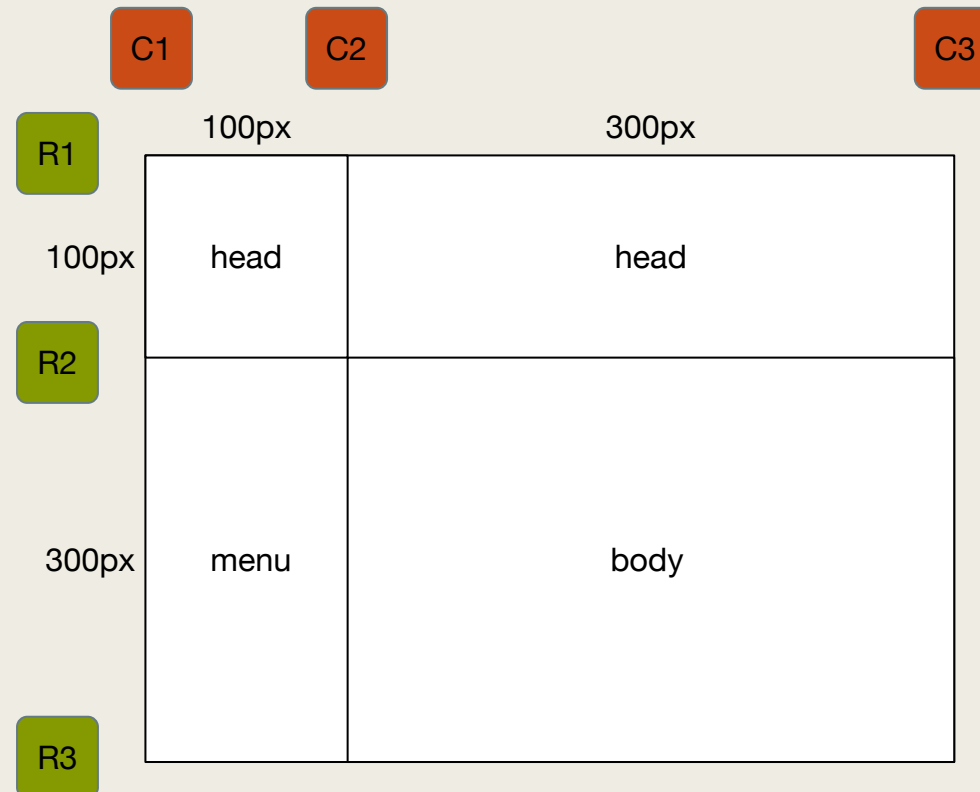


```
.body {  
  grid-column: 2/3;  
  grid-row: 2/3;  
  background-color: cornsilk;  
}
```

使用線條名稱

- 為線條取個名稱的語法為
 - [名稱1] size [名稱2] size [名稱3]

```
.container {  
  display: grid;  
  width: 400px;  
  height: 300px;  
  border: 1px solid green;  
  grid-template-columns: [c1] 100px [c2] auto [c3];  
  grid-template-rows: [r1] 100px [r2] auto [r3];  
}  
.head {  
  grid-column: c1/c3;  
  grid-row: r1/r2;  
  background-color: brown;  
}
```



媒體查詢

@media

- 想要針對不同的裝置來套用不同的 CSS，可以透過 @media 來偵測裝置
- 例如，螢幕還是印表機

```
@media screen {
```

```
}
```

```
@media print {
```

```
}
```

- 請找出，如何偵測目前螢幕是橫向（ Landscape ）還是直向（ Portrait ）？
- 文件：<https://developer.mozilla.org/zh-CN/docs/Web/CSS/@media>

RWD

- 響應式網頁：
根據不同大小
螢幕調整不同
排版樣式

```
@media screen and (max-width: 575px) {  
  h1 {  
    font-size: 20pt;  
    color: green;  
  }  
}
```

width >= 576 and width <= 767

```
@media screen and (min-width: 576px) and (max-width: 767px) {  
  h1 {  
    font-size: 40pt;  
    color: blue;  
  }  
}
```

width >= 768

```
@media screen and (min-width: 768px) {  
  h1 {  
    font-size: 100pt;  
    color: orange;  
  }  
}
```

行動裝置優先

- 使用 min-width
- 順序由小到大

```
@media screen and (min-width: 0px) {  
  h1 {  
    font-size: 20pt;  
    color: green;  
  }  
}  
  
@media screen and (min-width: 576px) {  
  h1 {  
    font-size: 40pt;  
    color: blue;  
  }  
}  
  
@media screen and (min-width: 768px) {  
  h1 {  
    font-size: 100pt;  
    color: orange;  
  }  
}
```

桌面優先

- 使用 max-width
- 順序由大到小

```
@media screen and (max-width: 1200px) {  
  h1 {  
    font-size: 100pt;  
    color: orange;  
  }  
}  
  
@media screen and (max-width: 992px) {  
  h1 {  
    font-size: 40pt;  
    color: blue;  
  }  
}  
  
@media screen and (max-width: 768px) {  
  h1 {  
    font-size: 20pt;  
    color: green;  
  }  
}
```



BOOTSTRAP

載入 Library

- 使用 CDN 載入。建議從官網複製。

```
<link  
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"  
rel="stylesheet" integrity="sha384-  
QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhJY6hW+ALEwIH"  
crossorigin="anonymous">
```

```
<script  
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min  
.js" integrity="sha384-  
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdslK1eN7N6jIeHz"  
crossorigin="anonymous"></script>
```

Breakpoints

Breakpoint	Class infix	Dimensions
Extra small		<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

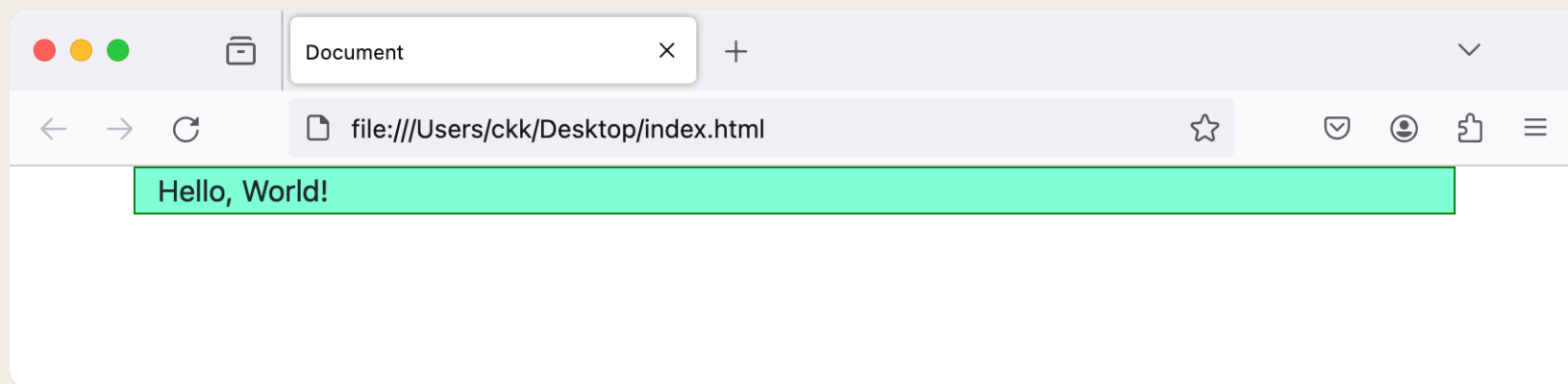
Container

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px	XX-Large ≥1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

Container 範例

- 當使用中綴 md 且螢幕寬度大於 768px 時，div 寬度與螢幕邊界間會出現 margin

```
<div class="container-md">Hello, World!</div>
```

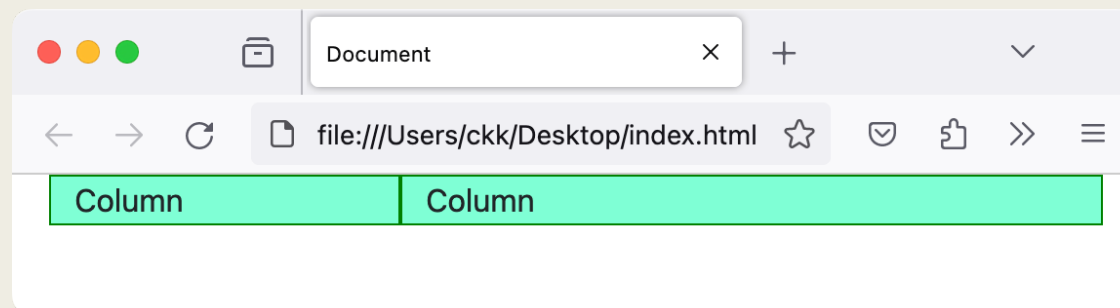


Grid 排版

- Bootstrap 將每列分為 12 格，利用後綴來控制每個儲存格佔幾格，例如：

```
<div class="container">  
  <div class="row">  
    <div class="col-4">Column</div>  
    <div class="col-8">Column</div>  
  </div>  
</div>
```

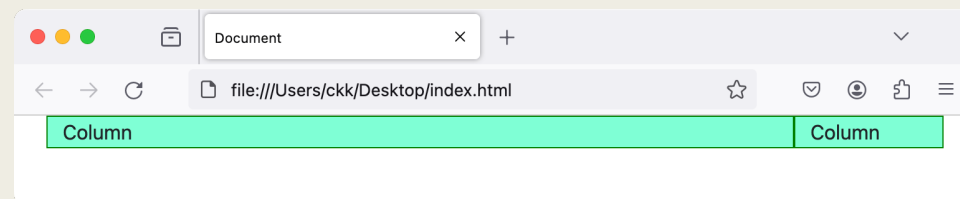
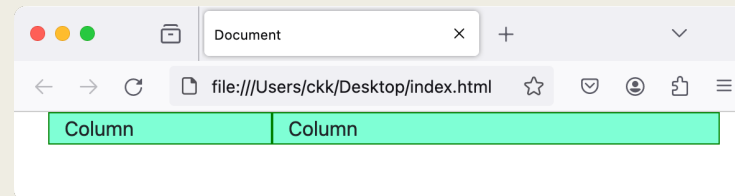
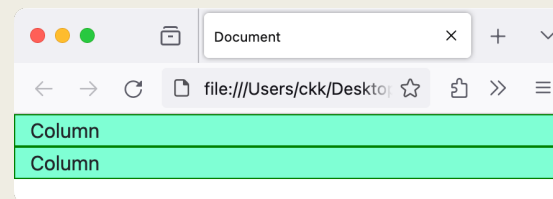
可寫成 col，表示「剩下的」



RWD

- 試試看不同寬度的瀏覽器上會看到什麼效果

```
<div class="container">  
  <div class="row">  
    <div class="col-sm-4 col-md-10">Column</div>  
    <div class="col-sm col-md">Column</div>  
  </div>  
</div>
```



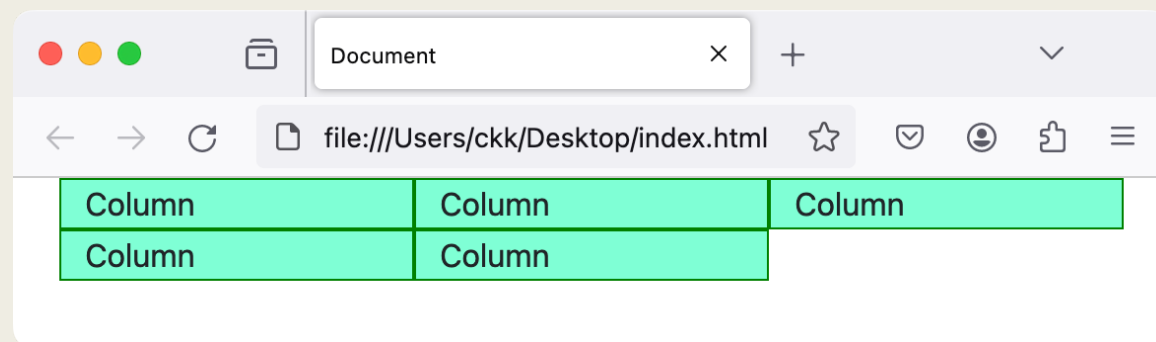
在列上控制一列多少欄

- 一列顯示三欄，超過三欄自動換列

```
<div class="container">  
  <div class="row row-cols-3">  
    <div class="col">Column</div>  
    <div class="col">Column</div>  
    <div class="col">Column</div>  
    <div class="col">Column</div>  
    <div class="col">Column</div>  
  </div>  
</div>
```

- 試試這個效果

```
<div class="row row-cols-1 row-cols-sm-3 row-cols-lg-5">
```



隱藏與顯示元件

■ 隱藏

Screen size	Class
Hidden on all	<code>.d-none</code>
Hidden only on xs	<code>.d-none .d-sm-block</code>
Hidden only on sm	<code>.d-sm-none .d-md-block</code>
Hidden only on md	<code>.d-md-none .d-lg-block</code>
Hidden only on lg	<code>.d-lg-none .d-xl-block</code>
Hidden only on xl	<code>.d-xl-none .d-xxl-block</code>
Hidden only on xxl	<code>.d-xxl-none</code>

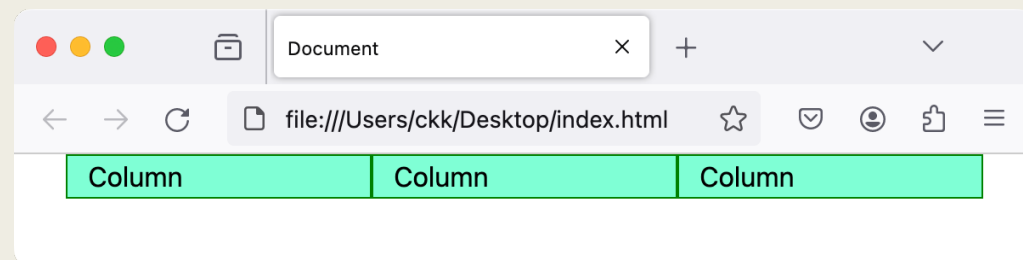
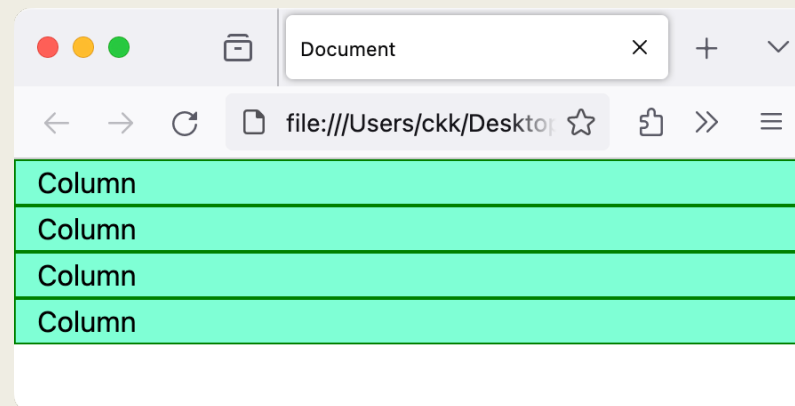
■ 顯示

Screen size	Class
Visible on all	<code>.d-block</code>
Visible only on xs	<code>.d-block .d-sm-none</code>
Visible only on sm	<code>.d-none .d-sm-block .d-md-none</code>
Visible only on md	<code>.d-none .d-md-block .d-lg-none</code>
Visible only on lg	<code>.d-none .d-lg-block .d-xl-none</code>
Visible only on xl	<code>.d-none .d-xl-block .d-xxl-none</code>
Visible only on xxl	<code>.d-none .d-xxl-block</code>

範例

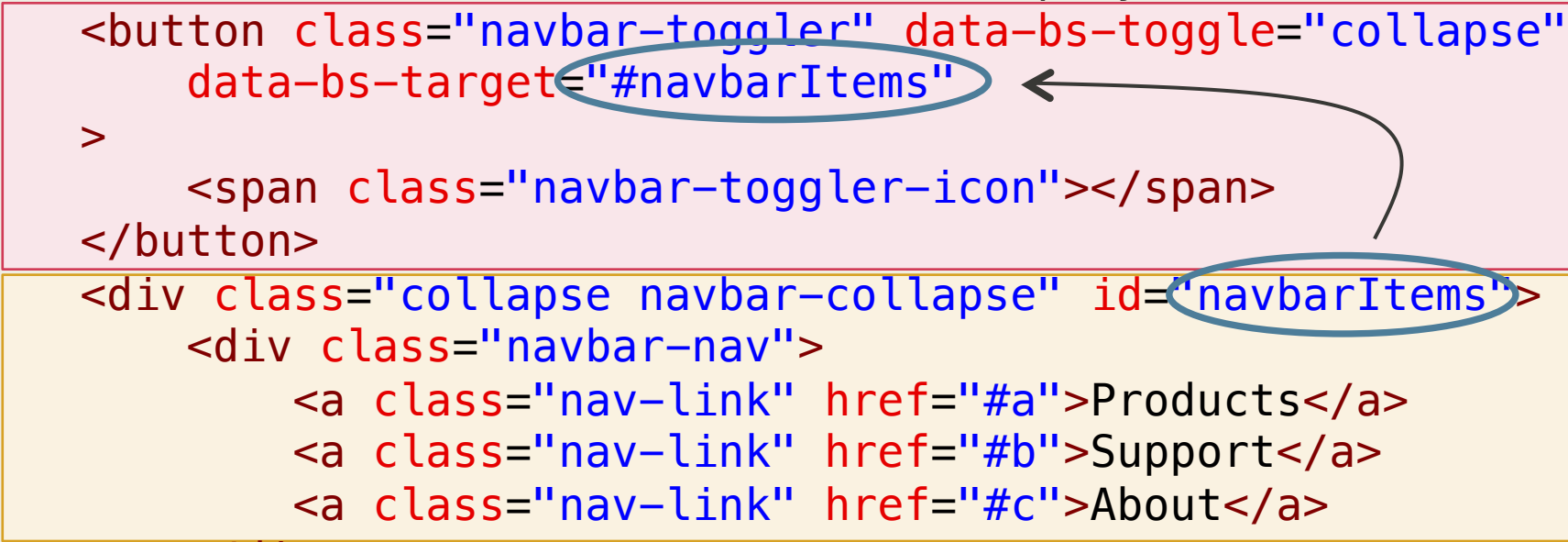
- 設定某格子在 sm 時顯示，其他解析度隱藏

```
<div class="container">
  <div class="row row-cols-1 row-cols-sm-3">
    <div class="col d-block d-sm-none">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```



導覽列與收合選單（漢堡選單）

```
<nav class="navbar navbar-expand-sm navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Company</a>
    <button class="navbar-toggler" data-bs-toggle="collapse"
      data-bs-target="#navbarItems"
    >
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarItems">
      <div class="navbar-nav">
        <a class="nav-link" href="#a">Products</a>
        <a class="nav-link" href="#b">Support</a>
        <a class="nav-link" href="#c">About</a>
      </div>
    </div>
  </div>
</nav>
```



The diagram illustrates the relationship between the `data-bs-target` attribute and the target element. A blue oval highlights the value `"#navbarItems"` in the `data-bs-target` attribute of the `navbar-toggler` button. Another blue oval highlights the `id="navbarItems"` attribute of the `collapse navbar-collapse` div. A black arrow points from the first oval to the second, indicating that the button's target is the collapse div with the matching ID.

得知點選項目

- 取得點選項目的 href 內容

```
window.onload = () => {  
    document.querySelectorAll('.nav-link').forEach((item) => {  
        item.onclick = (event) => {  
            event.preventDefault()  
            console.log(event.target.href)  
        }  
    })  
}
```