CSS

R3 CODE

كتب بواسطة : عبدالرحمن عرفات
Written by : Abdulrahman Arafat

لغة CSS هي اللغة المسؤولة عن برمجة التصميم الإبداعي للموقع فإن لغة HTML هي التي تضع البنية الاساسية للموقع اما التصميم المبهر فهو للغة CSS

إذا افترضنا أن الأمر يشبه البيت

HTML هي الجدران والقواعد

CSS هي الألوان والديكور

طرق استدعاء الـ css في الصفحة

1-External Style
2-Internal Style
3-Inline Style

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>welcome</title>
    <link rel="stylesheet" href="css.css"><!-- External Style -->
    <style>
        .d {
            color: red;
            font-size: 50px;
        }
    </style> <!-- Internal Style -->
</head>

<body>
    <h1 class="title">welcome</h1>
    <p>welcome</p>
    <p class="d">welcome</p>
    <p style="color: blue;">welcome</p><!--In Line Style -->

</body>
</html>
```

استدعاء عناصر الـ HTML في ملف ال CSS

استدعاء كل البراجرافات

```
P{
}
```

استدعاء العنصر باستخدام الـ CLASS

```
.className{
}
```

استدعاء العنصر بإستخدام الـ ID

```
#idName{
}
```

لو عندنا عنصر جوة عنصر جوة عنصر نوصلة ازاي
علي سبيل ا لمثال عندنا براجراف جوة كلاس جوة كلاس نوصل للبراجراف ازاي

```
.outer-class .inner-class p {}
```

و نقدر نمشي على نفس النهج للوصول لأي عناصر متداخلة

يمكن إضافة بعض الخصائص العامة لمجموعة عناصر في نفس السطر وهذا لايعني انه لن يمكنك التعديل علي كل عنصر في سطر آخر بشكل خاص ولكن عند التعديل على عنصر بشكل خاص يجب تجنب الخصائص التي اضفتها مسبقاً ضمن المجموعة

```
.z,
.y,
.x {
    width: 100px;
    height: 100px;
    background: red;
}
```

ويمكن أيضا استدعاء جميع عناصر الصفحة لوضع تنسيقات لها

```
v
```

كتابة التعليقات في لغة CSS

```
/* هذا تعليق */
```

في ال CSS احنا بنستدعي كل عنصر علشان نطبق علية اكواد التنسيق بتاعتنا

Background

هي مجموعة خصائص بتتحكم في خلفية أي عنصر في الصفحة،
سواء كانت لون، صورة، تدرج لوني، أو حتى فيديو (بطرق متقدمة)

اهم عناصر الBackground

1-Background-color

لازم نحدد حجم الخلفية ثم نقوم بتحديد لون الخلفية

القيم الممكنة

1-Color Name => Red

```
div {
  background-color: blue;
}
```

2-RGB => Red Green Blue

```
.z{
    width: 4000px;
    height: 3000px;
    background-color: rgb(255, 0, 0);
}
```

3-Hex => FF0000

4-Transparent => تعني ان الخلفية ستكون شفافة

الخاصية	الوظيفة	ا
background-image	يضيف صورة أو تدرج كخلفية	url("img.jpg"), linear-gradient(...)
background-repeat	يمنع أو يسمح بتكرار الصورة	Repeat Repeat-x Repeat-y no-repeat
background-position	تحديد موضع الصورة	Center Top Left 50% 50% 10px 20px
background-size	تحديد حجم الصورة	Auto Cover Contain 100px 200px
background-attachment	تثبيت الخلفية أثناء التمرير	Scroll Fixed local
background-origin	نقطة بداية الخلفية	Padding-box Border-box content-box
background-clip	يحدد أين تُعرض الخلفية داخل	Border-box

Padding-box content-box	العنصر	
يجمع كل ما سبق في سطر واحد	اختصار لكل الخصائص	background

مثال تطبيقي شامل

```
.card {
  width: 400px;
  height: 250px;
  background-color: #1a1a1a;
  background-image: url("code-bg.png"), linear-gradient(to bottom right, #0ff,
#00f);
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  background-attachment: fixed;
  background-origin: padding-box;
  background-clip: border-box;
}
```

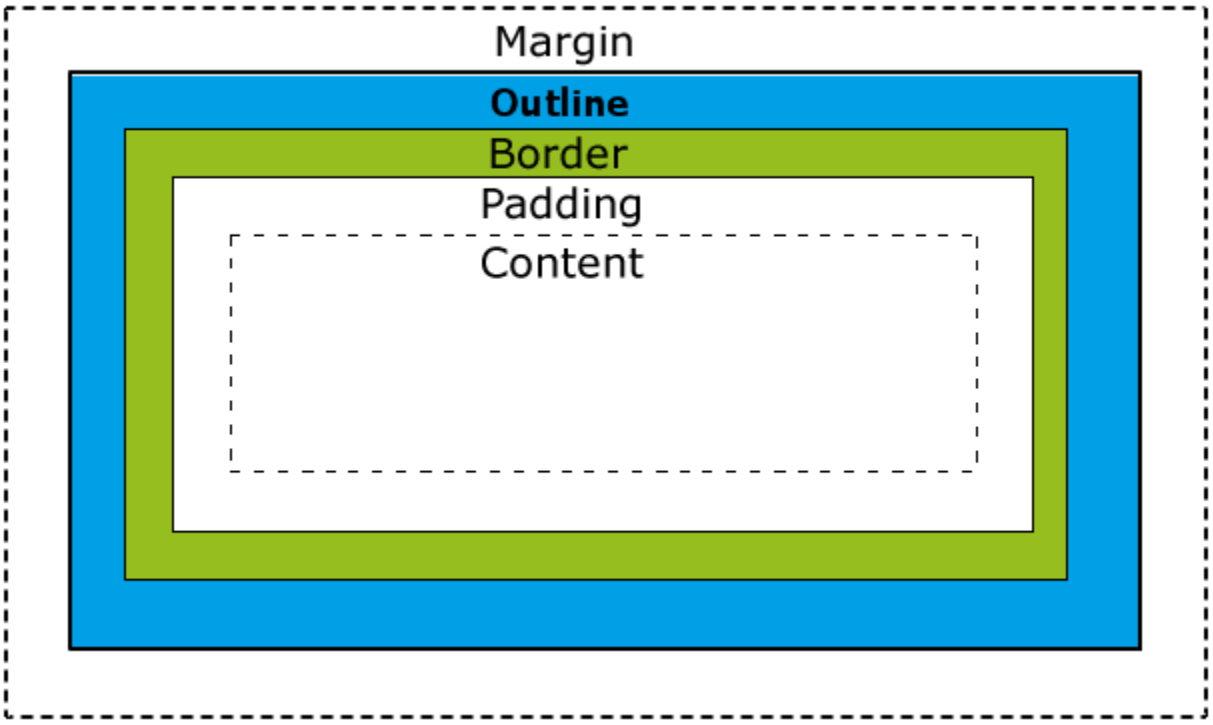
التعامل مع النص الزائد عن أبعاد الخلفية

```
p {
    color: rgb(255, 255, 0);
    background-color: black;
    width: 400px;
    /*حددنا العرض 400*/
    height: 300px;
   /*حددنا الارتفاع300 */
   overflow: visible;/*يظهر النص الزيادة خارج الإطار المحدد ودة التلقائي اصلا*/
   overflow: hidden;/*يخفي النص الزيادة خارج الإطار المحدد */
   overflow: scroll;/*يظهر شريط تمرير النص الزيادة خارج الإطار المحدد */
   overflow: auto;/*يظهر شريط تمرير فقط عند الحاجة */
   overflow-x: hidden;
      /*
      يخفي النص الزيادة خارج الإطار المحدد افقيا
و يظهر شريط تمرير عمودي فقط عند الحاجة
      */
```

```
   overflow-y: hidden;
/*يخفي النص الزيادة خارج الإطار المحدد عموديا
و يظهر شريط تمرير أفقي فقط عند الحاجة     */}
```

Box model

الهوامش



1-Padding

هي الي بتعمل مسافة بين العنصر وحواف العنصر في الطبيعي العنصر بيكون بائ من الحواف ودة مش بيكون لطيف لذلك ال padding بتخلي فية زي هامش الـ padding بياخد مني القيم بالبكسل ال padding خمس انواع ابسط من بعض

```
.z{
padding-top: 20px; /*make padding from top 20px*/
padding-left: 30px; /*make padding from left 30px*/
padding-right: 40px; /*make padding from right 40px*/
padding-bottom: 50px; /*make padding from bottom 50px*/
padding: 20px; /*make padding from all directions 20px*/
padding: 20px 30px;
/*Make Padding from top and bottom 20px, left and right 30px*/
padding: 20px 30px 40px 50px;
/*Make Padding from top 20px, right 30px, bottom 40px, left 50px*/
}
```

2-margin

هو الذي يحدد كم يبعد الصندوق عن الصناديق الأخرى أو العناصر الأخرى من حولة

```
.z{
margin-top: 20px; /*make margin from top 20px*/
margin-left: 30px; /*make margin from left 30px*/
margin-right: 40px; /*make margin from right 40px*/
margin-bottom: 50px; /*make margin from bottom 50px*/
margin: 20px; /*make margin from all directions 20px*/
margin: 20px 30px; /*Make Margin from top and bottom 20px, left and right
30px*/
margin: 20px 30px 40px 50px; /*Make Margin from top 20px, right 30px, bottom
```

```
40px, left 50px*/
}
```

3-Border

هو الي بيعمل إطار للصندوق
بياخد مني 3 قيم

```
.z{
border-width: 2px;
border-style: solid;
border-color: black;
border: 2px solid black;
/* border shorthand property sets all border properties in one declaration */
}
```

4-Outline

يشبه الـ border فهو يقوم بعمل اطار مثله لكن الـ border يقوم بعمل اطار داخل الصندوق اما ال Outline يقوم بعملة بالخارج ولا يتمتع بحرية التخصيص مثل الـ border فلا يستعمل بكثرة

Box sizing

بتاخد مني قيمة من اتنين

1-content-box

دي هي القيمة الافتراضية الي الموقع بيكون بيتعامل بيها اصلاً

2-border-box

دي بتخلي الموقع يحسب ال padding وال border ضمن المساحة الي انت بتديها للعنصر فيحافظ على تنسيق ال box

```
box-sizing: border-box;
```

TEXT

Paragraph

يتم استدعاء جميع ال p الموجودين في الصفحة للتعديل عليهم عن طريق الـ p المثال التالي يوضح

```
p {
    font-size: 20px;
    color: rgb(255, 255, 0);
    background-color: black;
    width: fit-content;
    /*حددنا العرض ليتناسب تلقائيا مع المحتوي*/
    height: fit-content;
   /*حددنا الارتفاع ليتناسب تلقائيا مع المحتوي */
    text-shadow: 3px 3px 0px  red;
}
/* تأخذ الخاصية 4 قيم
القيمة الأولى تحدد ما إذا كان بالاتجاة الايمن اذا كانت قيمة موجبة
أو بالاتجاه الايسر اذا كانت قيمة سالبة
القيمة الثانية تحدد مقدار الإزاحة العمودية للأعلى إذا كانت قيمة موجبة
او لاسفل اذا كانت قيمة سالبة
القيمة الثالثة تحدد مقدار الضبابية
القيمة الرابعة تحدد لون الظل */
```

```
p {
    font-size: 20px;
    color: rgb(255, 255, 255);
    background-color: black;
    text-align: center;
    /*تحديد موقع النص داخل العنصر*/
    direction: ltr;
    /*تحدد اتجاه النص داخل العنصر*/
```

```
    /*ltr: من اليسار إلى اليمين
    rtl: من اليمين إلى اليسار*/
}
```

```
p {
    font-size: 20px;
    color: rgb(255, 255, 255);
    background-color: black;
    text-align: center;
    text-decoration: line-through;
/*تحدد نمط النص
ونقدر نعمل بيه حاجات كتير ومتنوعة ده مجرد مثال بسيط
لعمل خط على النص  */
    text-indent: 2px;/*تحدد المسافة قبل النص*/
    line-height: 1.5;/*تحدد المسافة بين الأسطر*/
    word-spacing: 20px;/*تحدد المسافة بين الكلمات*/
    white-space: nowrap;/* بتخلي النص يكمل على سطر واحد */
    overflow: hidden; /* بتخلي النص اللي يخرج من الصندوق يختفي */
    text-overflow: ellipsis; /* بتضيف نقاط (...) للنص اللي يخرج من الصندوق */
   font-weight: bold;
/*BOLD بتخلي الخط
وممكن نعطيها قيم تانية غيرها بس هي الاكثر شيوعاً    */
}
```

Word breck

تستخدم لتقسيم العبارات داخل العناصر لما ميكونش فيه مساحة كافية لعرضها تستخدم في اكثر الاوقات مع اللغات التي لا تحتوي على مسافات زي الصينية على سبيل المثال

```
word-break: normal | break-all | keep-all | break-word;
```

القيمة	الوصف
`normal`	**القيمة الافتراضية. الكلمات بتتكسر بس عند النقاط الطبيعية زي المسافات أو الشرطات.**
`break-all`	**يكسر الكلمة في أي مكان لو مفيش مساحة، حتى لو في وسط الكلمة نفسها. مفيد للغات بدون مسافات.**
`keep-all`	**يمنع كسر الكلمات تمامًا. بيستخدم غالبًا مع اللغات الآسيوية.**
`break-word`	**بيكسر الكلمة في أي مكان لو مفيش مساحة، لكن بيحاول يحافظ على الكلمة قدر الإمكان. (مش رسمي في CSS3 لكنه مدعوم في معظم المتصفحات).**

Font Family

يمكننا استعمال ال font family من خلال طريقتين

-الطريقة الآمنة وهي استخدام خط من الخطوط المدمجة في اللغة

```
p{
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif
}
```

- او من خلال خطوط جوجل

1- الدخول لموقع google font

2-اختيار الخط المناسب

3- الضغط على get font

4-نقوم بنسخ Embed code in the <head> of your html ووضعه ضمن الروابط في الـ head

5-وبعدين بناخد ال css class الي عايزينها كوبي ونحطها في كود العنصر الي هنطبق عليه الخط ده في صفحة css

Calc

هي خاصية اقدر اضيف بيها اي عملية حسابية للكود علي سبيل المثال

```
width:calc(100% - 20px);
```

Position

هي خاصية بتحدد طريقة تموضع العنصر داخل الصفحة بالنسبة للعناصر الأخرى أو بالنسبة للصفحة نفسها.

أنواع القيم ل Position

القيمة	الوصف
statice	القيمة الافتراضية. العنصر يتموضع حسب تدفق الصفحة الطبيعي.
relative	العنصر يتموضع بالنسبة لمكانه الأصلي، ويمكن تحريكه باستخدام top, left, إلخ.
absolute	العنصر يتموضع بالنسبة لأقرب عنصر أبوه يكون position غير static.
fixed	العنصر يتموضع بالنسبة للنافذة (viewport)، ويظل ثابتًا عند التمرير.
sticky	مزيج بين relative و fixed. يبقى في مكانه حتى يتم التمرير لمسافة معينة.

مثال:

```
position: fixed ;
top: 300px;
right: 100px;
bottom: 300px;
left: 100px;
```

Z-index

يحدد ترتيب الطبقات فوق بعض
بياخد قيمة سالبة وموجبة كلما قلت قيمة الرقم أصبحت الطبقة في الخلف كلما زادت قيمة الرقم أصبحت الطبقة في الامام

```css
.box1 {
  position: absolute;
  z-index: 1;
  background-color: lightblue;
}

.box2 {
  position: absolute;
  z-index: 2;
  background-color: lightgreen;
}
```

Cursor

هي الخاصية المسئولة عن شكل مؤشر الماوس عند وضعه فوق عنصر معين

```
cursor: نوع المؤشر;
```

أنواع المؤشرات

القيمة	شكل المؤشر	الاستخدام
default	السهم العادي	الشكل الافتراضي
pointer	يد صغيرة	لما يكون العنصر قابل للنقر
text	مؤشر الكتابة	ما يكون العنصر فيه نص قابل للتحديد
move	اسهم تحريك	لما يكون العنصر قابل للسحب أو التحريك
not-allowed	دائرة فيها خط	لما يكون التفاعل غير مسموح
wait	ساعة رملية	لما يكون فيه تحميل او انتظار
crosshair	علمة +	يستخدم أحيانًا في أدوات الرسم
grab / grabbing	يد تمسك	لما يكون العنصر قابل للسحب

visibility

هي تعديل ظهور العناصر كإخفاء العناصر وغيرها من الأمور

```css
.z{
    width: 100px;
    height: 100px;
    background-color: red;
    display: none; /* تقوم بإخفاء العنصر ولكن لا يحتفظ بمكانه في التخطيط */
    visibility: hidden; /* تقوم بإخفاء العنصر ولكن يحتفظ بمكانه في التخطيط */
}
```

Display

Opacity

هي خاصية اقدر اغير بيها الشفافية

```
opacity: 0.1;
```

Variables

```
:root {
    --main-color: #3498db;
    --secondary-color: #2ecc71;
    --tertiary-color: #e74c3c;
    --quaternary-color: #f1c40f;
}
/* عملنا متغيرات*/
.z,
.y,
.x {
    width: 100px;
    height: 100px;
    background: var(--main-color);/*اعطينا العنصر قيمة أحد المتغيرات*/
}
```

layouts

flex box

Flex box : هو نظام يستخدم لتنظيم العناصر داخل الحاوية او داخل الصفحة بشكل مرن

لنبدأ في الشرح

هذا هو كود html الذي سنطبق عليه الشرح ب css

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <div class="container">
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
    </div>

</body>
</html>
```

ازاي بنفعل ال flex علي الحاوية

display

```
.container{
    display: flex;
}
```

Flex بتاخد عرض الصفحة تلقائياً باعتبارها block element
ولحل المشكلة بنستخدم inline-flex ودي بتخليه ياخد العرض المناسب للمحتوى فقط بدلاً من عرض الصفحة

الترتيب الطبيعي للصفحه بيكون column (رأسي) بمجرد ما بنفعل ال flex العناصر بتتحول من column (عمودي) الي row (افقي)

Main Axis → أفقي

Cross Axis → رأسي

```
---------------------->Main Axis
|
|
|
▼
Cross Axis
```

طيب على سبيل المثال أنا فعلت ال flex بس انا عايز العناصر تكون column اعمل اي؟

هنا بيجي دور

Flex-direction

ودي خاصية بتغير الاتجاه للعناصر جوه ال flex container
وبتاخد 4 قيم

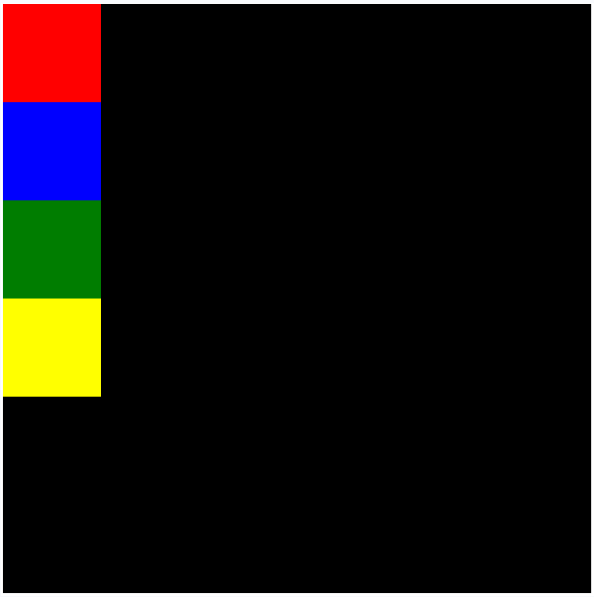
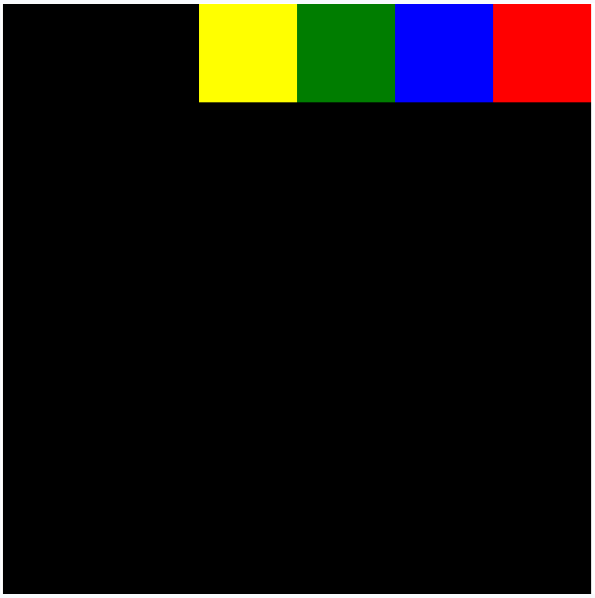
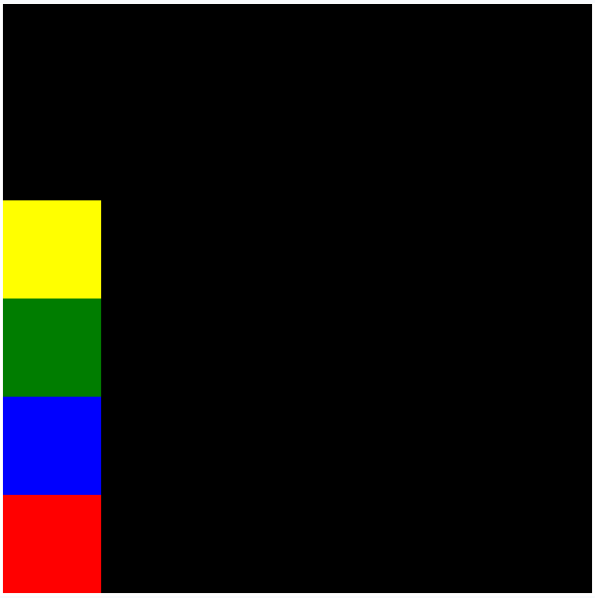
1-row
2-column
3-row-reverse
4-column-reverse

Row-reverse لو العناصر جاية من الشمال لليمين هي تخليهم من اليمين للشمال يعني بتعكس row
column-reverse لو العناصر جاية من فوق لتحت هي تخليهم من تحت لفوق يعني بتعكس column

دة ال flex العادي

```css
.container{
    width: 600px;
    height:600px;
    display: flex;
    background-color: black;

}
.red{
    background-color: red;
}
.green{
    background-color: green;
}
.blue{
    background-color: blue;
}
.yellow{
    background-color: yellow;
}
.item{
    width: 100px;
    height: 100px;
}
```

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

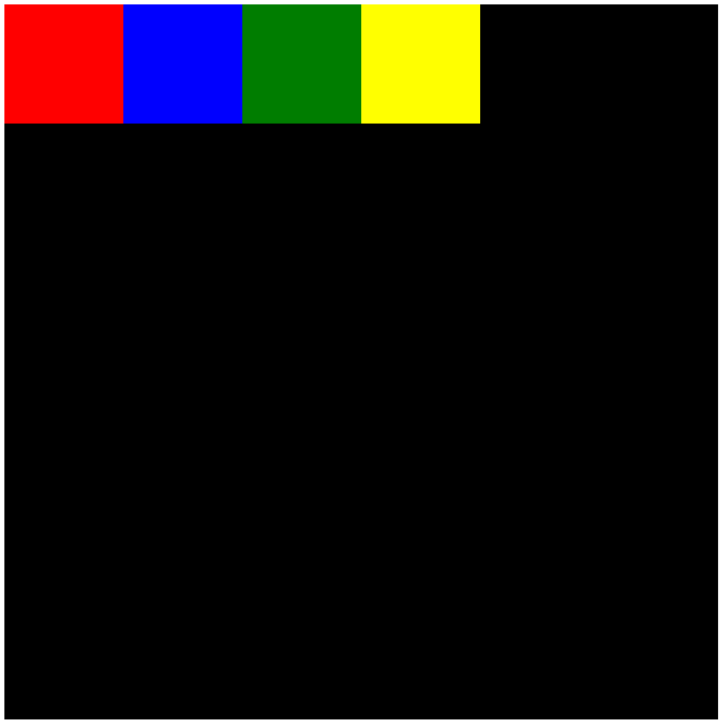
طيب لو دلوقتي عايز اتحكم في المحاذاة العمودية للعناصر هنا بيجي دور

Align-items

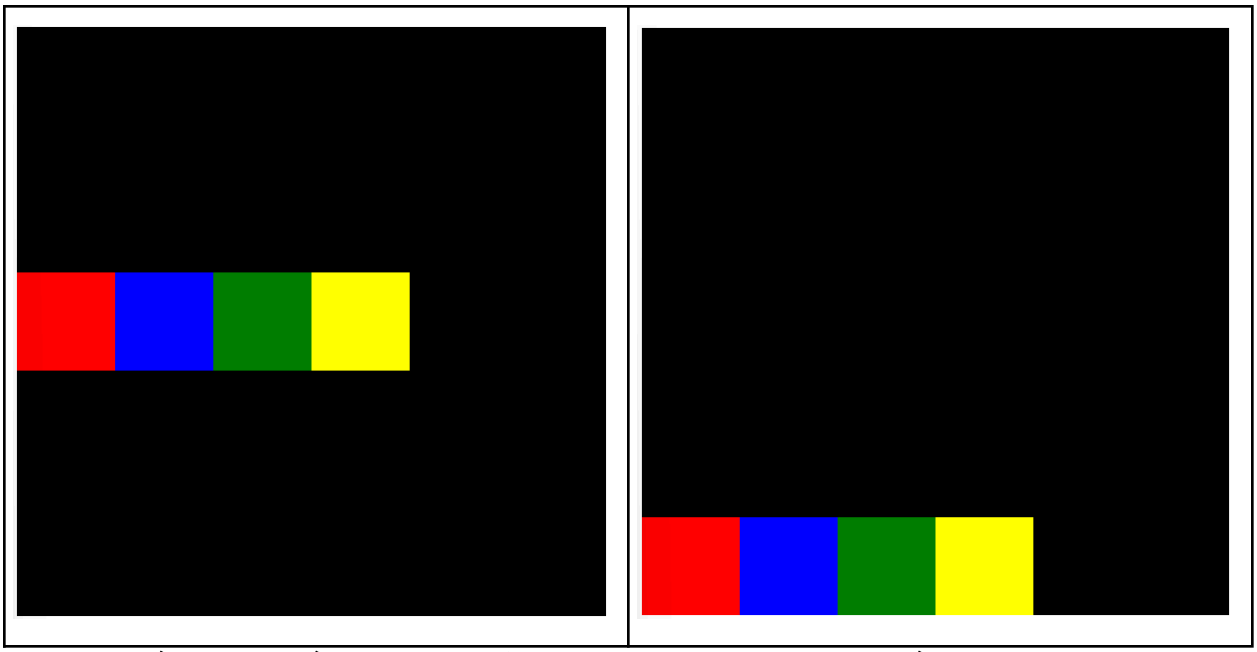
هي تتحكم في المحور العرضي Cross Axis

```
.container{
    width: 600px;
    height:600px;
    display: flex;
    background-color: black;
    align-items: flex-start;
}
.red{
    background-color: red;
}
.green{
    background-color: green;
}
.blue{
    background-color: blue;
}
.yellow{
    background-color: yellow;

}
.item{
    width: 100px;
    height: 100px;
}
```



`align-items:center;`	`align-items:flex-end;`



Stretch هي قيمة بتخلي العناصر تمتد في المساحة الفاضية ولكن يشترط عدم وجود *height* للعناصر

```
align-items:stretch;
```

فية عندنا بردو حاجة اسمها ***align-self*** ودي بتتحكم في المحور العمودي لعنصر انا بحدده يعني بتستخدم على عنصر واحد او مجموعة عناصر ليهم نفس ال *class* مش علي ال *container* الي

فيه العناصر زي *align-items* وبتاخد نفس القيم

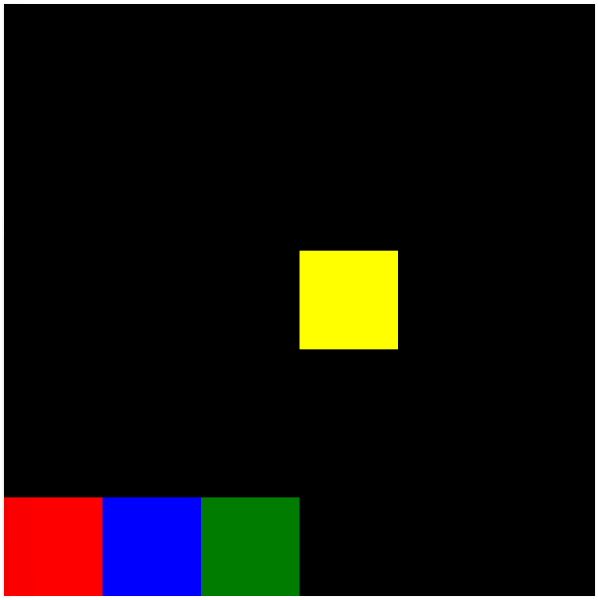
1-Flex-right

2-flex-left

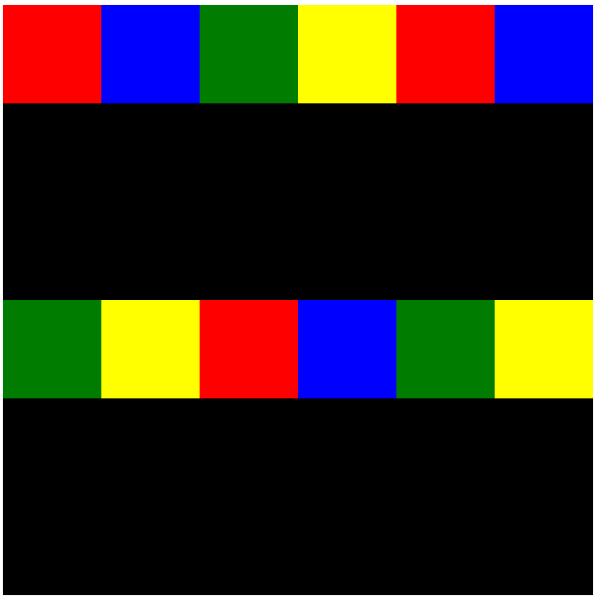
3-center

4-stretch

```
.container{
    width: 600px;
    height:600px;
    display: flex;
    background-color: black;
    align-items: flex-end;
}
.red{
    background-color: red;
}
.green{
    background-color: green;
}
.blue{
    background-color: blue;
}
.yellow{
    background-color: yellow;
    align-self: center;
}
.item{
    width: 100px;
    height: 100px;
}
```



طيب لو عندنا عناصر كل عنصر
عرضة 100 بكسل والحاوية
عرضها 600 بكسل وعندنا 9
عناصر هيكون الشكل عامل كدة
العناصر هتتكمش علشان
متطلعش بره الحاوية
فالقائمين على لغة css
عملوا flex-wrap علشان كدة



```css
.container{
    width: 600px;
    height:600px;
    display: flex;
    background-color: black;
    flex-wrap: wrap;
    align-content: space-between;
}
.red{
    background-color: red;
}
.green{
    background-color: green;
}
.blue{
    background-color: blue;
}
.yellow{
    background-color: yellow;
}
.item{
    width: 100px;
    height: 100px;
}
```

flex-wrap: nowrap (default)

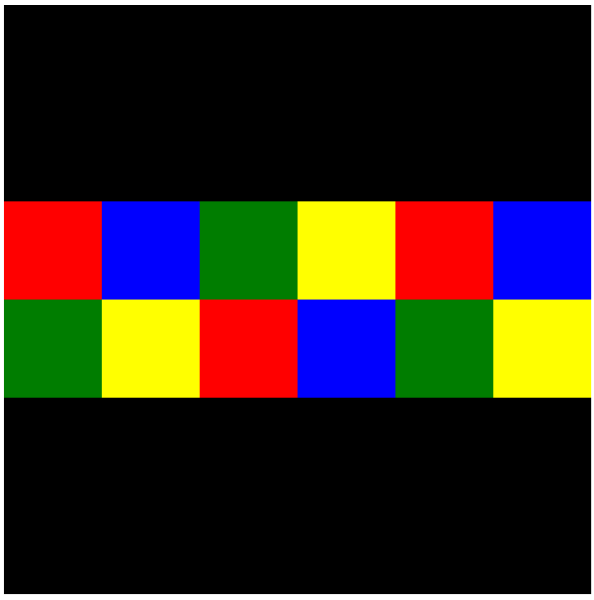
flex-wrap: wrap
flex-wrap: wrap-reverse

وبالرغم انه تم حل المشكله الا نا التنسيق مكانش منظم فعملوا

Align-content

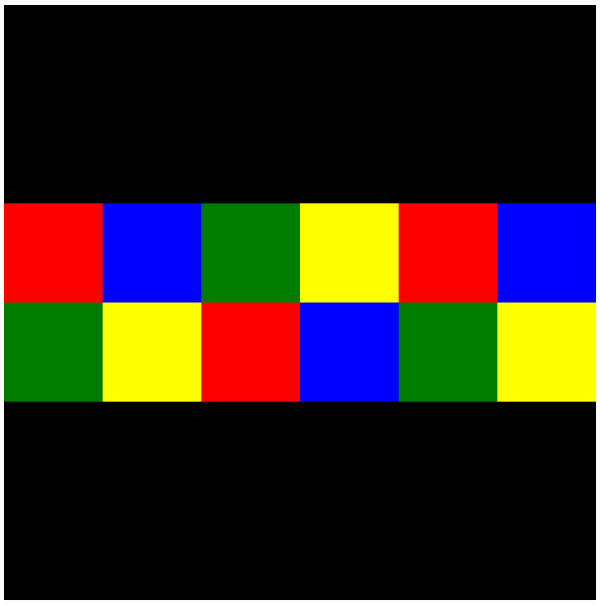
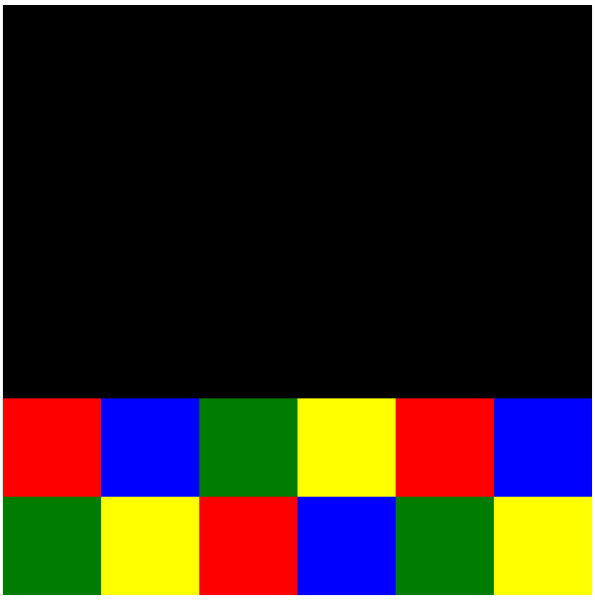
علشان تظبط المسافات بين السطور

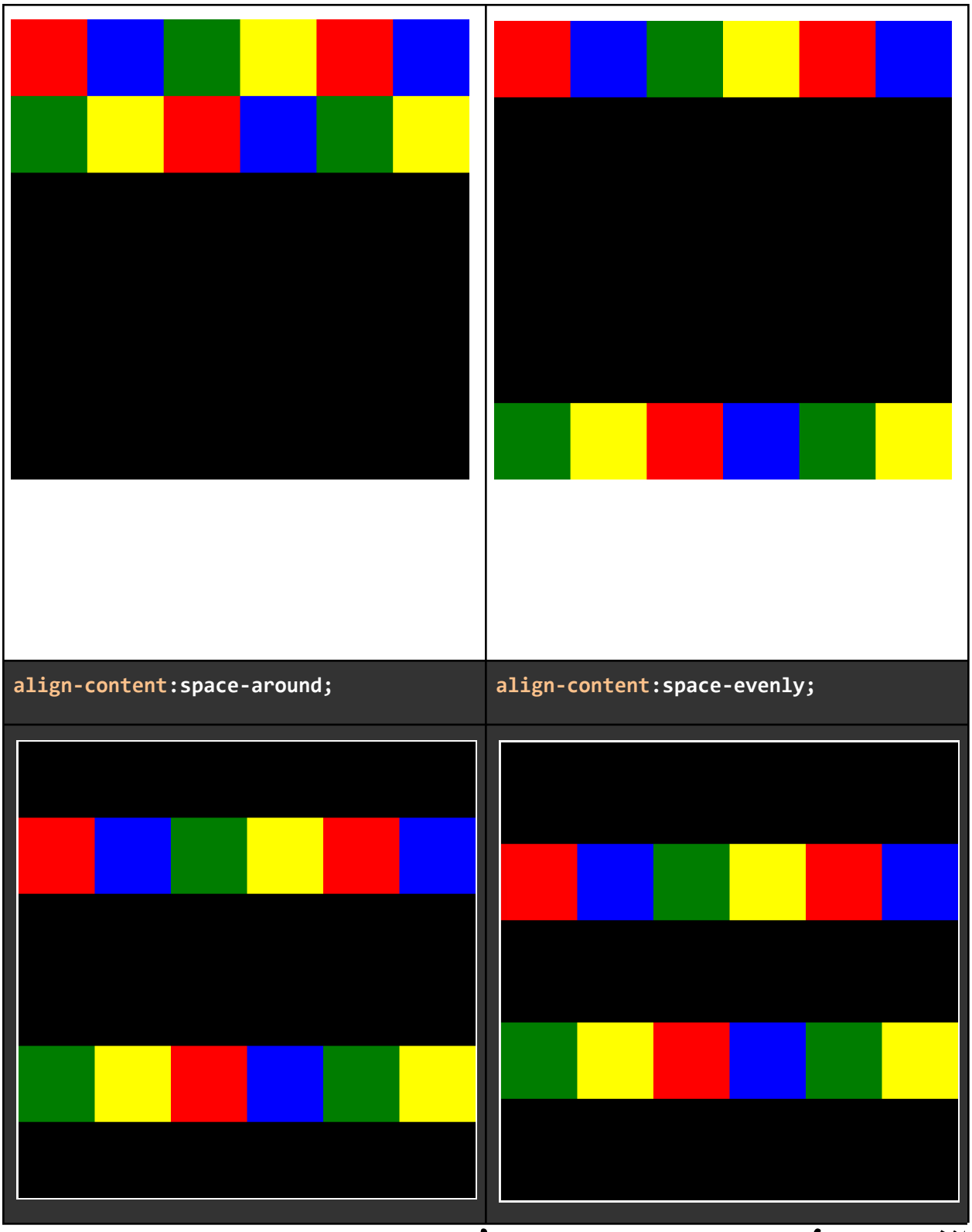
```
.container{
    width: 600px;
    height:600px;
    display: flex;
    background-color: black;
    flex-wrap: wrap;
    align-content:center;
}
.red{
    background-color: red;
}
.green{
    background-color: green;
}
.blue{
    background-color: blue;
```



```
}
.yellow{
    background-color: yellow;
}
.item{
    width: 100px;
    height: 100px;
}
```

القيم التي تقبلها *align-content*

align-content:center;	align-content:flex-end;
	
align-content:flex-start;	**align-content:space-between;**



الفرق بين space-around و space-evenly

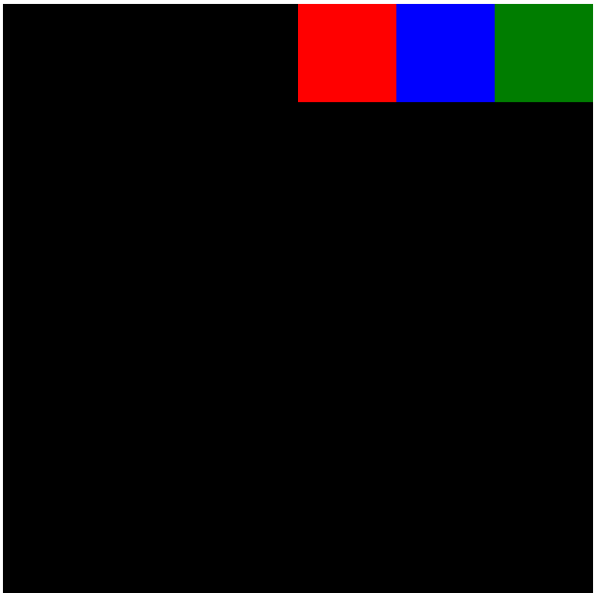
Space-around بتخلي في مساحة حول الأسطر بس المساحة الي

برا بتكون اكبر اما *space-evenly* تكون المساحة بين السطور نفس برا السطور

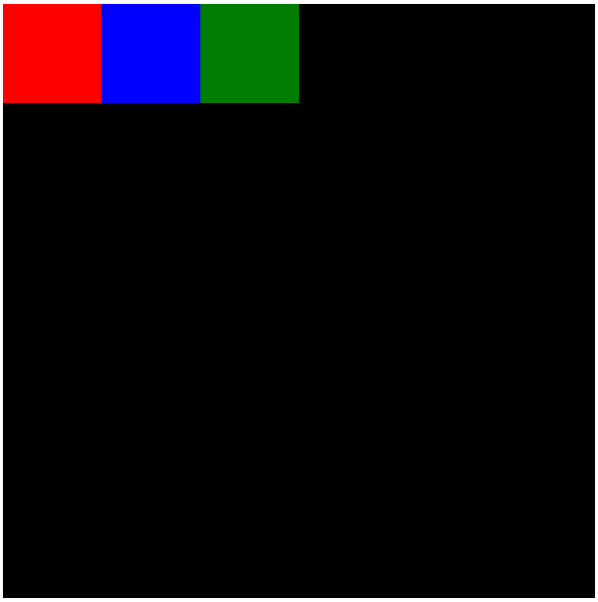
Justify-content

هي خاصية تستخدم لمحاذاة وتوزيع العناصر على المحور الرئيسي إذا كان كان *flex-direction : row;* فهي هتشتغل علي المحور *row* و نفس الكلام لو *column* تستخدم الخاصية علي الحاوية وليس على عنصر محدد

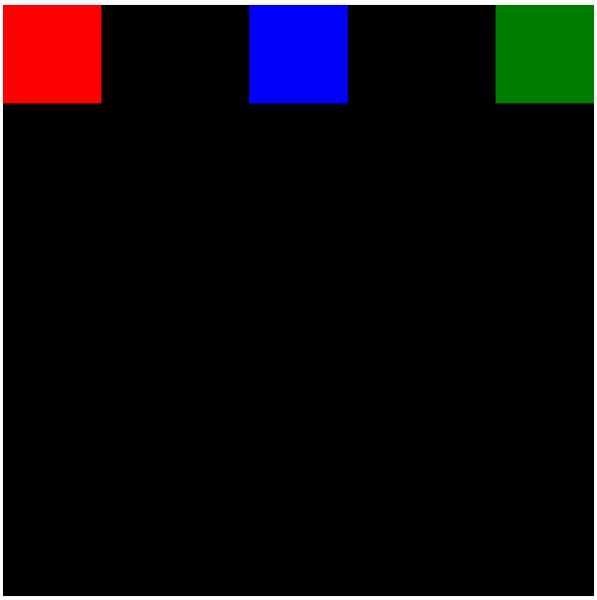
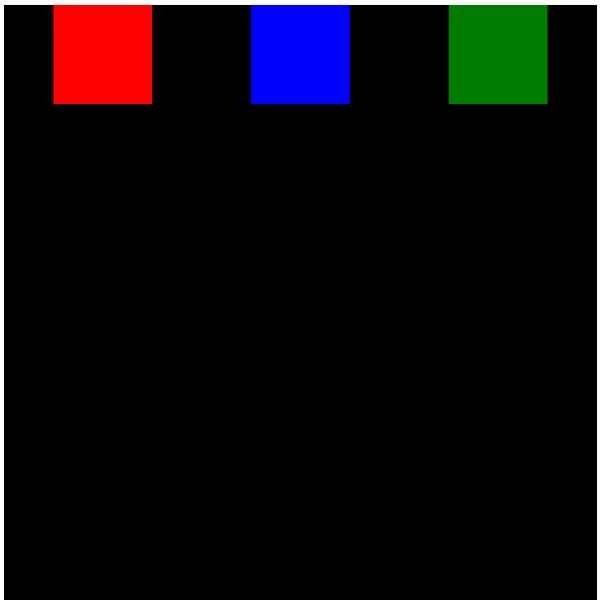
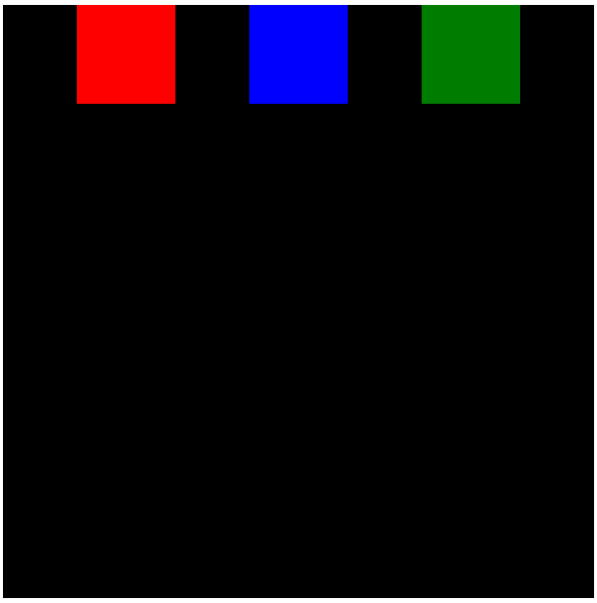
`justify-content:flex-start;`	`justify-content:flex-end;`



`justify-content:center;`



`justify-content:space-between;`

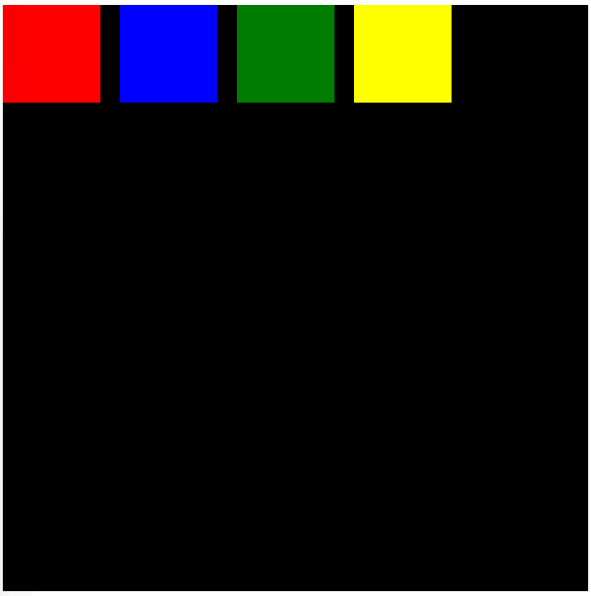
	
`justify-content:space-around;`	`justify-content:space-evenly`
	

Gap

gap هي خاصية تستخدم على *flex-container* لتحديد مسافة بين عناصره

```css
.container{
    width: 600px;
    height:600px;
    display: flex;
    align-items: flex-start;
    justify-content: flex-start;
    background-color: black;
    gap:20px;
}

.item{
    width:100px;
    height:100px;
}
.red{
    background-color: red;
}
.green{
    background-color: green;
}
.blue{
    background-color: blue;
}
.yellow{
    background-color: yellow;
}
```



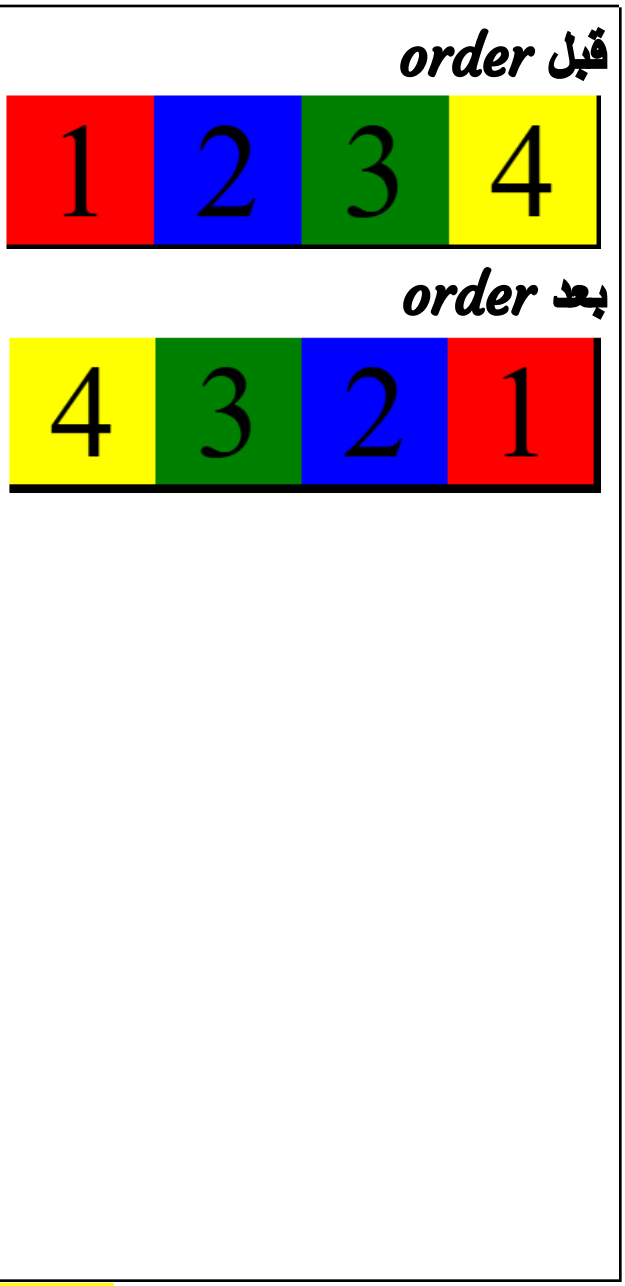
وكمان تقدر تستخدم

row-gap || column-gap

Order

هي الخاصية المسئولة عن ترتيب العناصر
العنصر صاحب القيمة الأقل هو الأول
القيمة الافتراضية *0* ويمكن اعطاء قيم موجبة او سالبة

```css
.container{
    width: 600px;
    height:600px;
    display: flex;
    align-items: flex-start;
    justify-content: flex-start;
    background-color: black;
    row-gap: 20px;
}
.item{
    width:100px;
    height:100px;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 90px;
}
.red{
    background-color: red;
    order: 3;
}
.green{
    background-color: green;
    order:1;
}
.blue{
    background-color: blue;
    order: 2;
}
.yellow{
    background-color: yellow;
    order: -1;
}
```



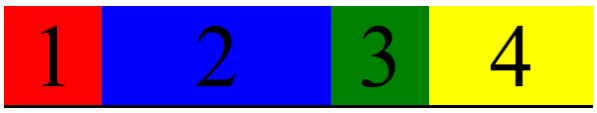
flex-grow

دي بتخلي العنصر لو فيه مساحة فاضية يمتد ويستغلها ولو فية 100px فاضيين العنصر يمتد ويستغلهم

Flex-grow

تقبل قيم زي 0 دي القيمة الافتراضية 1 العنصر يمتد 2 العنصر يمتد ضعف امتداد العنصر الي قيمتة 1 وهكذا كل ما العدد يزيد

```
.blue{
    background-color: blue;
    flex-grow: 2;
}
.yellow{
    background-color: yellow;
    flex-grow: 1;
}
```



flex-shrink

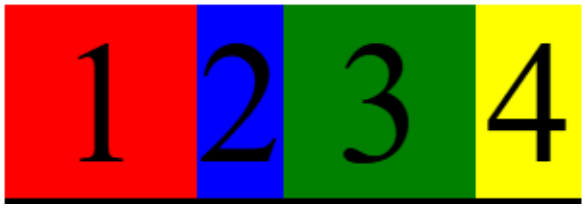
دي بتخلي العنصر لو المساحة مش كافية يصغر ولو فية *50px* ناقصين عن المساحة اللي محتاجاها العناصر العنصر يصغر ويسيب مساحة للعناصر التانية

Flex-shrink

تقبل قيم زي *0* دي القيمة الافتراضية *1* العنصر يصغر *2* العنصر يصغر ضعف تقلص العنصر الي قيمتة *1* وهكذا كل ما العدد يزيد

يرجي العلم ان عند تقليل المساحة فجميع العناصر تتقلص الا العناصر التي تملك min-width & min-height

```
.red{
    background-color: red;
    min-width: 100px;
}
.green{
    background-color: green;
        min-width: 100px;
}
.blue{
    background-color: blue;
    flex-shrink: 2;
}
.yellow{
    background-color: yellow;
    flex-shrink: 1;
}
```



Flex-basis

بنستخدمها عشان نعطي الحجم المبدئي للعناصر في flexbox
وهي نفس width ولو استخدمنا width هتشتغل ولكن
flex-basis هي الأساس في ال flexbox

```
Flex-basis:100px;
```

Flex short hand

1-flex-flow: direction wrap;

direction - wrap **ودي بتختصر معايا حاجتين**

2-flex: grow shrink basis;

grow - shrink - basis **ودي بتختصر 3 حاجات**

3-gap: row-gap column-gap;