CSS : Cascading Style Sheets

Plan

Plan

Cascading Style Sheets

Définition et caractéristiques

- un langage qui permet de bien gérer la mise en page de documents HTML
- une définition homogène des styles dans un site Web
- une combinaison de styles pour une meilleure réutilisation / personnalisation
- trois façons pour définir ces styles :
  - en utilisant l'attribut `style` dans des balises HTML classiques
  - dans la page html au sein de la section `<head>` par une balise `<style>`
  - dans un fichier d'extension `.css` appelé dans une page d'extension `.html` par la balise `<link>`

Cascading Style Sheets

CSS : évolution

- CSS1 : introduit en 1996. Début très difficile à cause de la guerre entre les navigateurs (IE, Netscape...)
- CSS2 : apparue en 1998, environ 70 nouvelles propriétés par rapport à CSS1
- **CSS3** : version officiellement non finalisée mais plusieurs nouvelles propriétés prises en charge par les navigateurs
- En parallèle de CSS3, on parle de CSS4 qui a débuté en 2010

HTML en 91 et CSS en 96 ?

Comment fait-on avant le CSS ?

- des balises html comme `<b>...</b>, <u>...</u>,...`
- avec la balise `<font color="#000000">`

Trois façons de le faire

Première méthode :

```
<tag style="property: value;">
```

Exemple :

```
<p style="color:red;">
```

Le texte de ce paragraphe aura un arrière plan rouge

Trois façons de le faire

Deuxième méthode :

```
<style type="text/css">
    selector
    {
        property: value;
    }
```

Exemple :

```
<style type="text/css">
    p
    {
        color: red;
    }
```

Tous les paragraphes de mon document seront affichés en rouge.

Trois façons de le faire

Avec HTML 5

```
<style>
    p
    {
        color: red;
    }
```

Plus besoin de préciser le type pour la balise `style`.

Trois façons de le faire

Troisième méthode :
Dans le fichier CSS (file.css) :

```
p
{
    color: red;
}
```

Dans le fichier HTML :

```
<head>
    <link rel="stylesheet" type="text/css" href="
     file.css">
</head>
```

Tous les paragraphes de mon document seront affichés en rouge.

Exemple

```
<!doctype HTML>
<html>
    <head>
        <title>Premiere page web </title>
        <meta charset='utf-8'/>
        <style type="text/css">
            p{
                color: blue;
            }
        </style>
    </head>
    <body>
        <p> je m'affiche en bleu </p>
        <p style="color:red;"> je m'affiche en rouge </p>
        <p> je m'affiche en bleu </p
    </body>
</html>
```

je m'affiche en bleu
je m'affiche en rouge
je m'affiche en bleu

Sélecteur

C'est quoi ?

- Une balise HTML
- Une classe
- Un identifiant
- Une combinaison de sélecteurs

Sélecteur

Une classe (`class`)

regroupe plusieurs éléments dans une même famille

```
<!DOCTYPE html>
    <body>
        <h1 class='bleu'> titre </p>
        <p class='rouge'> bonjour </p>
        <p class='bleu'> bonsoir </p>
    </body>
```

Dans le fichier CSS :

```
.bleu
{
    color: blue;
}
.rouge
{
    color: red;
}
```

Sélecteur

Un identifiant (`id`)

concerne un seul élément de la page

```
<!DOCTYPE html>
    <body>
        <p id='gras'> bonsoir </p>
    </body>
```

Dans le fichier CSS :

```
#gras
{
    font-weight: bold;
}
```

Autres sélecteurs

Les sélecteurs

`*`	tous les éléments	
`h1, p`	les h1 et p	`<h1>...<p>...<p>...<h1>`
`div p`	les p situés dans h1	`<div>oui<p>...<span>non<p>`
`p#gras`	les p portant l'id gras	`<p id='gras'>oui<p>non`
`#gras`	tout élément avec id gras	`<p id='gras'>oui<p>non`
`p.bleu`	les p de la classe bleu	`<p class='bleu'>oui<p>non`
`h1 + p`	les p directement après h1	`<h1>...<p>oui<p>non`
`div > p`	les p enfant direct de div	`<div>...<p>oui</p>` `<div>...<span>...<p>non`
`div ~ p`	les p précédés par div avec p et div ont le même parent	`<ul>...</ul><p>non</p>` `<div>...</div><p>oui</p>`

Autres sélecteurs

Exercice 1 : sans utiliser de classes, ni d'identifiants, écrire le code CSS qui permet de colorier la liste suivante ainsi :

- Langages de programmation
  - Java
  - C++
  - PHP
- Éditeurs de texte
  - Sublime text
  - Atom
  - Notepad++

Autres sélecteurs

Exercice 2 : en utilisant une seule classe (aucun identifiant), écrire le code CSS qui permet de colorier la liste suivante ainsi :

- France
  - Bleu
  - Blanc
  - Rouge
- Tunisie
  - Blanc
  - Rouge
- Uruguay
  - Blanc
  - Bleu

Autres sélecteurs

Les sélecteurs

`input[type=text]`	les inputs de type text	`<input type='text'/>`
`img[title~=red]`	les images avec un titre contenant le mot red	`<img title='red bull'/>oui` `<img title='redbull'/>non`
`img[title*=red]`	les images avec un titre contenant le mot red	`<img title='red bull'/>oui` `<img title='redbull'/>oui`
`[lang|=en]`	les éléments avec un attribut lang commençant par en	`<div lang="en-us">oui` `<div lang="fr">non`
`img[src$=.jpg]`	les images ayant un nom se terminant par .jpg	`<img src='1.jpg'/>oui` `<img src='jpg.png'/>non`

Autres sélecteurs

Les pseudo-sélecteurs

`p:first-letter`	la première lettre de chaque élément p
`p:first-line`	la première ligne de chaque élément p
`p:before`	le contenu avant chaque élément p
`p:after`	le contenu après chaque élément p
`p:lang(en)`	les p avec un attribut `lang` dont la valeur commence par 'en'
`td[colspan='3']`	les cases d'un tableau qui sont sur 3 colonnes
`li:not(:first-child)`	tous les éléments `<li>` qui ne sont pas les premiers
`p:nth-child(2)`	les p qui sont le second enfant de leur parent
`p:nth-last-of-type(2)`	les p qui sont le second enfant de leur parent en commençant par le dernier
`div ul.menu li.entree`	les li de classe 'entree' qui sont dans 'ul' de classe 'menu', qui sont dans des div
`a:hover`	les liens survolés
`a:visited`	les liens visités
`input:optional`	les inputs sans la propriété `required`
`input:required`	les inputs avec la propriété `required`
`input:read-only`	les inputs avec la propriété `readonly`
`input:read-write`	les inputs sans la propriété `readonly`

Exemple

```
<body>
    <p msg="ligne␣ecrite␣le␣16␣octobre␣2017">
     formation PHP </p>
</body>
```

```
p:before {
    content: "bonjour␣";
}
p:after {
    content: "␣c'est␣ma␣page␣("attr(msg)")␣Aurevoir"
     ;
}
```

Affiche :

bonjour formation PHP c'est ma page (ligne ecrite le 16 octobre 2017) Aurevoir

Structure d'un fichier CSS

```
Selecteur(s)[:pseudoSelecteur]
{
    property: value;
}
```

Exemple

Pour bien maîtriser les sélecteurs

- `https://flukeout.github.io/`

Propriétés du texte

Les valeurs de la propriété `font-style`

- `italic`
- `normal`
- `oblique`

Les valeurs de la propriété `font-weight`

- `normal`
- `bold`

Les valeurs de la propriété `text-decoration`

- `underline`
- `overline`
- `line-through`
- `none`

Les valeurs de la propriété `font-size`

- `nombre px`
- `small, large, medium, x-large, xx-large,...`
- `decimal em` : 1em la taille normale

Les valeurs de la propriété `font-family`

- `"Times New Roman"`
- `Arial`
- ...

La propriété `@font-face`

pour définir sa propre police

Les valeurs de la propriété `text-align`

- `center`
- `left` (par défaut)
- `right`
- `justify`

Autres propriétés sur le texte

Autres propriétés

- `color` : couleur du texte
- `background-color` : couleur de fond
- `background-image` : url(image de fond)
- `opacity` : transparence (valeur de 0 à 1)

Autres propriétés sur le texte

Exercice : étant donnée une balise `<div class=container>` contenant plusieurs balises paragraphes, écrire le code CSS qui permet d'afficher en gras le paragraphe survolé, en italic les paragraphes situés après le paragraphe survolé et en souligné les paragraphes situés avant le paragraphe survolé

Exemple :

Je fais parti des paragraphes situés avant le paragraphe survolé.

Je fais parti des paragraphes situés avant le paragraphe survolé.

Je suis le paragraphe survolé.

Je fais parti des paragraphes situés après le paragraphe survolé.

Je fais parti des paragraphes situés après le paragraphe survolé.

Autres propriétés sur le texte

Correction : HTML

```
<div class="container">
  <p>paragraph 1</p>
  <p>paragraph 2</p>
  <p>paragraph 3</p>
  <p>paragraph 4</p>
  <p>paragraph 5</p>
</div>
```

Correction : CSS

```
.container:hover p {
  text-decoration: underline;
}
.container p:hover{
  text-decoration: none;
  font-weight: bold;
}
.container p:hover ~ p {
  text-decoration: none;
  font-style: italic;
}
```

Propriété multivaluée

Les valeurs de la propriété `border`

- `border-width` : épaisseur
- `border-color` : couleur
- `border-style` : plusieurs valeurs possibles : `none`, `solid`, `dashed`, `dotted`...

```
Selecteur
{
    border: 1px red dotted;
}
```

Encore les bordures

Autres propriétés de bordure

- `border-top` : haut
- `border-bottom` : bas
- `border-left` : gauche
- `border-right` : droite
- `border-radius` : pour arrondir une bordure, peut prendre 4 valeurs différentes pour en haut à gauche, en haut à droite, en bas à droite, en bas à gauche.
- `box-shadow` : pour l'ombre. prend quatre valeurs, le décalage horizontal de l'ombre, le décalage vertical de l'ombre, l'adoucissement du dégradé et la couleur de l'ombre.
- `text-shadow` : pour l'ombre d'un texte.

Exemple

```
p
{
    border-bottom: 1px red dotted;
    border-top: 1px red dashed;
    border-right: 1px blue solid;
    border-left: 1px blue double;
    box-shadow: 3px 3px 1px gray;
}
```

Et pour les listes ?

Comment changer les puces ou les numéros ?

- La propriété : `list-style-type`
- Les valeurs possibles pour les listes non-ordonnées : `circle`, `square`, `none`...
- Les valeurs possibles pour les listes ordonnées : `upper-roman`, `lower-alpha`, `none`...

On peut aussi utiliser une image comme puce pour les `<ul>`

- `list-style-image: url('image.gif');`

Et pour les listes ?

Supprimer les valeurs par défaut ? (pour un menu par exemple)

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}
```

Les puces à l'intérieur ou à l'extérieur de la liste

- `list-style-position`
- Les valeurs possibles : `inside` ou `outside` (par défaut)

Et pour les listes ?

Exemple avec `outside`

- Tu as regardé l'égaliseur de Denzel Washington ? ou John Wick de Keanu Charles Reeves ?
- Tu aimes bien le Brésil ? Tu te souviens de Romario, Bebeto, Ronaldo et Dunga ?

Et pour les listes ?

Exemple avec `outside`

- Tu as regardé l'égaliseur de Denzel Washington ? ou John Wick de Keanu Charles Reeves ?
- Tu aimes bien le Brésil ? Tu te souviens de Romario, Bebeto, Ronaldo et Dunga ?

Exemple avec `inside`

- Tu as regardé l'égaliseur de Denzel Washington ? ou John Wick de Keanu Charles Reeves ?
- Tu aimes bien le Brésil ? Tu te souviens de Romario, Bebeto, Ronaldo et Dunga ?

Et pour les listes ?

Exemple avec `outside`

- Tu as regardé l'égaliseur de Denzel Washington ? ou John Wick de Keanu Charles Reeves ?
- Tu aimes bien le Brésil ? Tu te souviens de Romario, Bebeto, Ronaldo et Dunga ?

Exemple avec `inside`

- Tu as regardé l'égaliseur de Denzel Washington ? ou John Wick de Keanu Charles Reeves ?
- Tu aimes bien le Brésil ? Tu te souviens de Romario, Bebeto, Ronaldo et Dunga ?

Autre exemple sur `https://www.w3schools.com/css/tryit.asp?filename=trycss_list-style-position`

Et pour les listes ?

Un raccourci ?

```
ul {
    list-style: square inside url("image.gif");
}
```

Explication

Les valeurs, dans l'ordre, correspondent à :

- `list-style-type` : si un style-image est spécifié, ceci sera affiché quand l'image ne peut être affiché
- `list-style-position` : pour indiquer la position des puces (à l'intérieur ou à l'extérieur)
- `list-style-image` : pour spécifier l'image à utiliser comme puce

Les positions d'un élément

`position : static`

- position, par défaut, naturelle sur la page.

Les positions d'un élément

`position : static`

- position, par défaut, naturelle sur la page.

`position : relative`

- position qui peut changer par rapport à une position de référence (un `left: 30px;` par exemple).

Les positions d'un élément

Pour tester la différence : CSS

```
div.static {
    left: 30px;
    position: static;
    border: 3px solid #73AD21;
}
div.relative {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21;
}
```

HTML

```
<div class ="static">
  Bonjour tout le monde
</div>

<div class ="relative">
  Bonjour tout le monde
</div>
```

Les positions d'un élément

Autres positions

- `position : absolute` : position absolue définie par rapport au block englobant (un élément positionné en absolu ne se réfère pas à son élément parent direct, mais au premier ancêtre positionné qu'il rencontre).
- `position : fixed` : position qui reste fixe, l'élément est toujours positionné par rapport à la fenêtre du navigateur même si on descend dans la page (avec un scroll par exemple).
- `position : sticky` (collant) : position dépendante de défilement de l'utilisateur

Des exemples simples à voir sur le lien suivant :

https://www.w3schools.com/css/css_positioning.asp

Les positions d'un élément

La propriété `z-index`

- `z-index` prend une valeur numérique. L'élément avec un plus grand `z-index` sera placé au dessus des autres.
- il ne fonctionne que sur des éléments ayant une position `absolute`, `relative` ou `fixed`.

Dimension d'une boite

Propriété et valeur

- `width` (largeur) : valeur en pixel, pourcentage relatif
- `height` (hauteur) : valeur en pixel, pourcentage relatif
- `min-width` (largeur minimale) : valeur en pixel, pourcentage relatif
- `min-height` (hauteur minimale) : valeur en pixel, pourcentage relatif
- `max-width` (largeur minimale) : valeur en pixel, pourcentage relatif
- `max-height` (hauteur minimale) : valeur en pixel, pourcentage relatif

Les marges

Deux types de marges

- `margin` (marge extérieure) : valeur en pixel
- `padding` (marge intérieure) : valeur en pixel

marge extérieure

espace entre les frontières de la boite et les boites situées à l'extérieur

- `margin-top` (marge extérieure en haut) : valeur en pixel
- `margin-bottom` (marge extérieure en bas) : valeur en pixel
- `margin-right` (marge extérieure à droite) : valeur en pixel
- `margin-left` (marge extérieure à gauche) : valeur en pixel

Les marges

marge intérieure

espace entre le contenu et les frontières de la boite

- `padding-top` (marge intérieure en haut) : valeur en pixel
- `padding-bottom` (marge intérieure en bas) : valeur en pixel
- `padding-right` (marge intérieure à droite) : valeur en pixel
- `padding-left` (marge intérieure à gauche) : valeur en pixel

Les marges

La propriété `overflow`

Et si notre texte ne rentre pas dans la boite, on peut utiliser la propriété `overflow` pour indiquer ce qu'il faut faire

- `hidden` : cacher le texte qui dépasse
- `visible` : le texte reste visible à l'extérieur de la boite
- `scroll` : ajouter une barre de défilement pour parcourir le texte qui dépasse
- `auto` : le navigateur décide de rajoute une barre de défilement si nécessaire

La propriété `float`

La propriété `float`

permet de déclarer et choisir l'emplacement d'un objet flottant.

La propriété `float`

La propriété `float`

permet de déclarer et choisir l'emplacement d'un objet flottant.

Les valeurs de la propriété `float`

- `left`
- `right`

Les valeurs de la propriété `clear` : pour annuler `float`

- `left`
- `right`
- `both`

Exemple avec les propriétés float et clear

Considérons l'exemple suivant :

Le fichier index.html

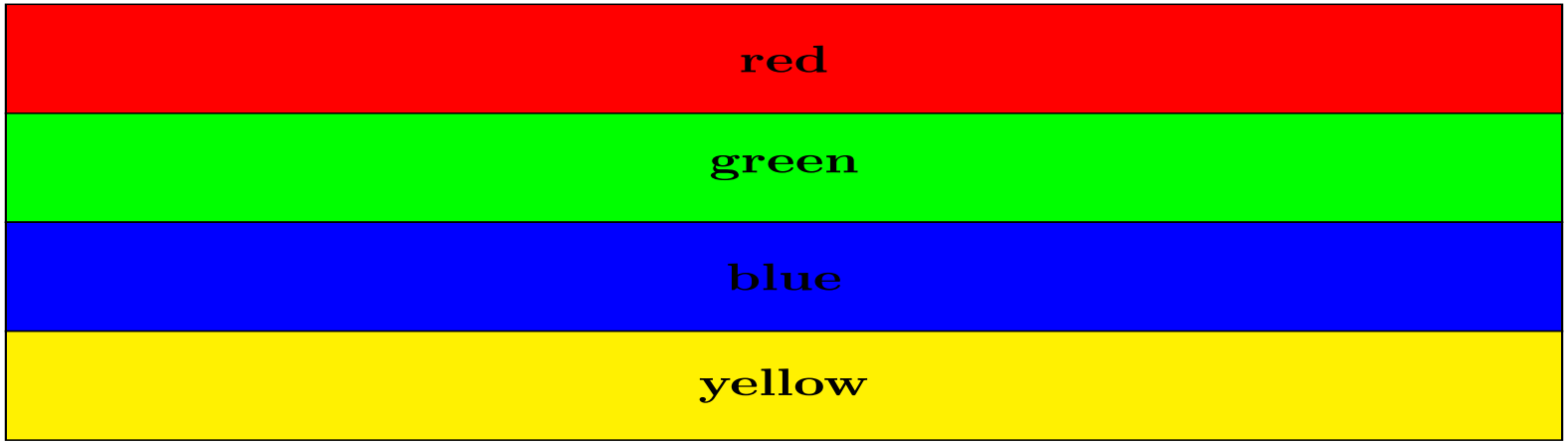
```
<!DOCTYPE html>
<html>
  <head>
    <title> float et clear </title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div class="container" >
      <div class= component1 >red</div>
      <div class= component2 >green</div>
      <div class= component3 >blue</div>
      <div class= component4 >yellow</div>
    </div>
  </body>
</html>
```

Le fichier style.css

```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container > div{
  text-align: center;
}
```

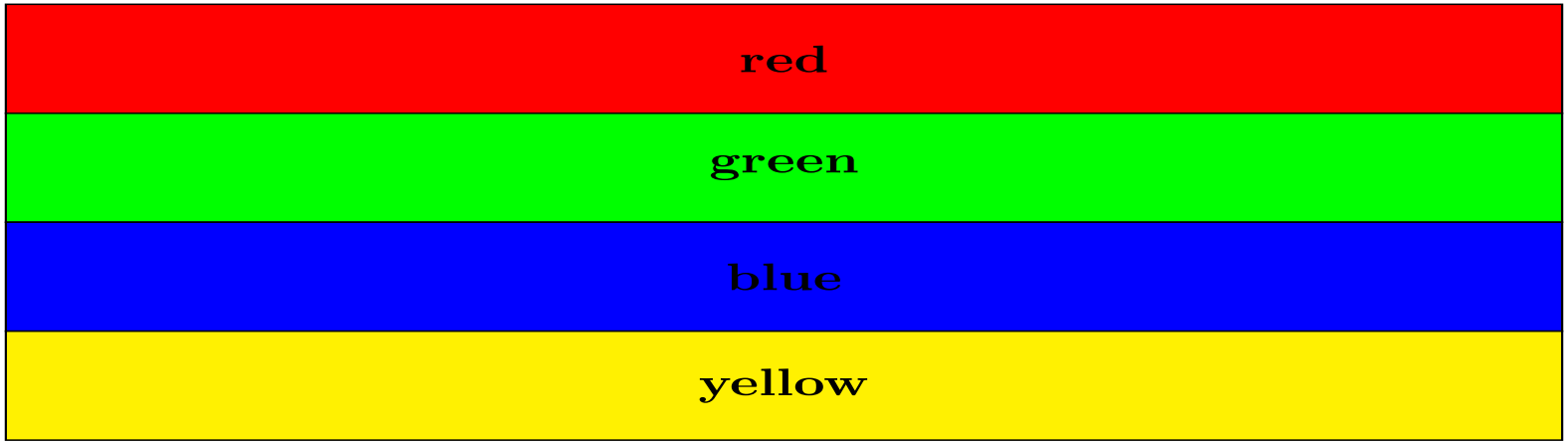
Exemple avec les propriétés `float` et `clear`

Le résultat du code précédent :



Exemple avec les propriétés `float` et `clear`

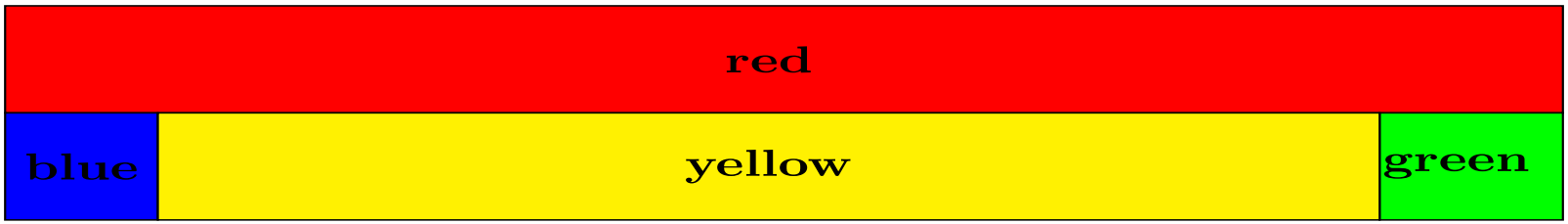
Le résultat du code précédent :



Et si on veut placer le bleu à gauche du vert ?

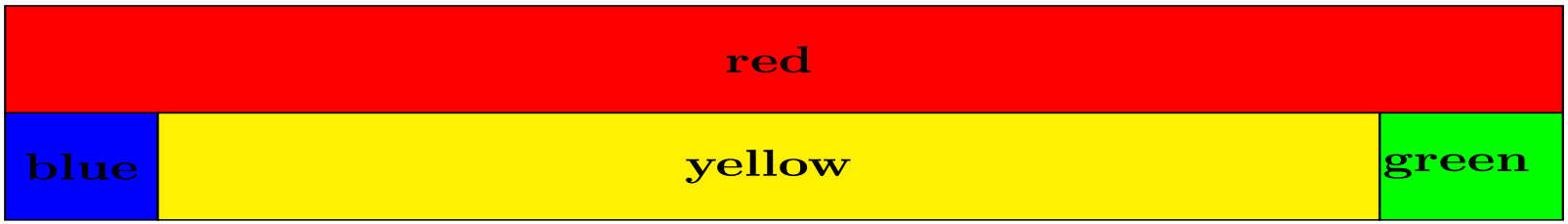
Exemple avec les propriétés `float` et `clear`

```
.component2 {
  float: right;
  background-color: green;
}
.component3{
  float: left;
  background-color: blue;
}
```



Exemple avec les propriétés float et clear

```
.component2 {
  float: right;
  background-color: green;
}
.component3{
  float: left;
  background-color: blue;
}
```

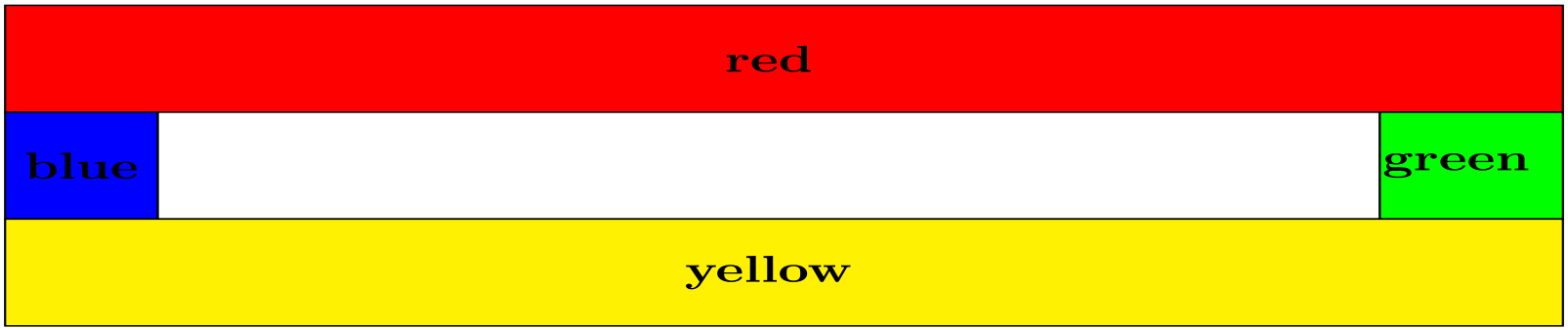


Oui, mais le jaune, on voulait qu'il reste à la ligne

Exemple avec les propriétés `float` et `clear`

Dans ce cas, il faut ajouter

```
.component4{
  clear: both;
  background-color: yellow;
}
```



La propriété visibility

La propriété visibility

Plusieurs valeurs possibles pour visibility :

- hidden : pour rendre un élément invisible (sa place est conservée)
- visible : pour rendre un élément visible
- collapse : pour rendre un élément du tableau invisible, mais la place qu'il occupe est perdue (propriété compatible seulement avec IE et Firefox)
- inherit : pour avoir la même visibilité de l'élément parent

La propriété visibility

collapse vs hidden : CSS

```
#hiddenRow{
        visibility: hidden;
}
#collapseRow{
        visibility: collapse;
}
```

collapse vs hidden : HTML

```
<table>
        <tr ><td > 1 </td><td > 2 </td></tr>
        <tr id=hiddenRow><td> 3 </td><td > 4 </td></tr>
        <tr ><td > 5 </td><td > 6 </td></tr>
        <tr id=collapseRow><td> 7 </td><td > 8 </td></tr>
        <tr ><td > 9 </td><td > 10 </td></tr>
</table>
```

La propriété visibility

collapse vs hidden : CSS

```
#hiddenRow{
        visibility: hidden;
}
#collapseRow{
        visibility: collapse;
}
```

collapse vs hidden : HTML

```
<table>
        <tr ><td > 1 </td><td > 2 </td></tr>
        <tr id=hiddenRow><td> 3 </td><td > 4 </td></tr>
        <tr ><td > 5 </td><td > 6 </td></tr>
        <tr id=collapseRow><td> 7 </td><td > 8 </td></tr>
        <tr ><td > 9 </td><td > 10 </td></tr>
</table>
```

Résultat :

```
1       2

5       6
9       10
```

Display

La propriété `display`

Chaque élément HTML a une propriété `display` qui est par défaut soit `inline` soit `block`.

Display

La propriété `display`

Chaque élément HTML a une propriété `display` qui est par défaut soit `inline` soit `block`.

Autres valeurs possibles ?

Oui : `none`, `inline-block`, `flex`, `grid`...

Display

La propriété `display`

Chaque élément HTML a une propriété `display` qui est par défaut soit `inline` soit `block`.

Autres valeurs possibles ?

Oui : `none`, `inline-block`, `flex`, `grid`...

La valeur `none` pour la propriété `display`

Un élément HTML ayant la valeur `none` pour la propriété `display` ne sera pas affiché, sans avoir à le supprimer réellement de la page (DOM).

visibility:hidden vs display:none

À ne pas confondre display : none et visibility : hidden

- display: none masque totalement l'élément et annule des propriétés telles que margin, padding, width...
- visibility: hidden masque seulement l'élément, ce qui peut laisser des espaces vides.

visibility:hidden VS display:none

Pour tester : CSS

```
.propDisplay{
  display: none;
}
.propHidden{
  visibility: hidden;
}
```

HTML

```
Bonjour. Apres les deux points, il y a un mot dont  display = none : <
 span class ="propDisplay"> ICI </span>. Entre parentheses, il y a un
 mot dont visibility = hidden (<span class ="propHidden"> ICI </span>).
```

visibility:hidden VS display:none

Pour tester : CSS

```
.propDisplay{
  display: none;
}
.propHidden{
  visibility: hidden;
}
```

HTML

```
Bonjour. Apres les deux points, il y a un mot dont  display = none : <
 span class ="propDisplay"> ICI </span>. Entre parentheses, il y a un
 mot dont visibility = hidden (<span class ="propHidden"> ICI </span>).
```

Le résultat

```
Bonjour. Apres les deux points, il y a un mot dont display = none : .
 Entre parentheses, il y a un mot dont visibility = hidden (     ).
```

visibility:hidden VS display:none

Pour tester : CSS

```
.propDisplay{
  display: none;
}
.propHidden{
  visibility: hidden;
}
```

HTML

```
Bonjour. Apres les deux points, il y a un mot dont  display = none : <
 span class ="propDisplay"> ICI </span>. Entre parentheses, il y a un
 mot dont visibility = hidden (<span class ="propHidden"> ICI </span>).
```

Le résultat

```
Bonjour. Apres les deux points, il y a un mot dont display = none : .
 Entre parentheses, il y a un mot dont visibility = hidden (     ).
```

Aller vérifier le DOM dans le navigateur

Display

La valeur `inline` pour la propriété `display`

- `inline-block` : permet de placer des éléments `inline` tout en préservant leurs capacités d'éléments `block`, tels que la possibilité de définir une largeur et une hauteur, des marges et padding top et bottom,...

inline-block VS inline VS block

Exemple avec block : CSS

```
.propBlock{
        display: block;
        height: 100px;
        width: 100px;
}
```

Exemple avec block : HTML

```
<p> Bonjour, ceci est une balise <span class= propBlock> block</span>,
 aurevoir</p>
<p> CSS3 </p>
```

inline-block VS inline VS block

Exemple avec block : CSS

```
.propBlock{
        display: block;
        height: 100px;
        width: 100px;
}
```

Exemple avec block : HTML

```
<p> Bonjour, ceci est une balise <span class= propBlock> block</span>,
 aurevoir</p>
<p> CSS3 </p>
```

Le résultat

Bonjour, ceci est une balise
block

, aurevoir

inline-block VS inline VS block

Exemple avec inline : CSS

```
.propInline{
        display: inline;
        height: 100px;
        width: 100px;
}
```

Exemple avec inline : HTML

```
<p> Bonjour, ceci est une balise <span class= propInline> inline</span>
 , aurevoir</p>
<p> CSS3 </p>
```

inline-block VS inline VS block

Exemple avec inline : CSS

```
.propInline{
        display: inline;
        height: 100px;
        width: 100px;
}
```

Exemple avec inline : HTML

```
<p> Bonjour, ceci est une balise <span class= propInline> inline</span>
 , aurevoir</p>
<p> CSS3 </p>
```

Le résultat

```
Bonjour, ceci est une balise inline, aurevoir

CSS3
```

inline-block VS inline VS block

Exemple avec inline-block : CSS

```
.propInlineBlock{
        display: inline-block;
        height: 100px;
        width: 100px;
}
```

Exemple avec inline : HTML

```
<p> Bonjour, ceci est une balise <span class= propInlineBlock> inline-
 block</span>, aurevoir</p>
<p> CSS3 </p>
```

inline-block VS inline VS block

Exemple avec inline-block : CSS

```
.propInlineBlock{
        display: inline-block;
        height: 100px;
        width: 100px;
}
```

Exemple avec inline : HTML

```
<p> Bonjour, ceci est une balise <span class= propInlineBlock> inline-
 block</span>, aurevoir</p>
<p> CSS3 </p>
```

Le résultat

```
Bonjour, ceci est une balise inline-block      , aurevoir




CSS3
```

Display

Autres valeurs de `display`

- `list-item` : pour afficher sous forme de liste, avec retour à la ligne après chaque élément.
- `inherit` : pour avoir les propriétés du block conteneur.
- `flex`
- `grid`
- `table`
- ...

Les menus

Exercice

Avant de continuer avec **Flexbox** et **Grid**, faisons un exercice sur les menus.

Les menus

Exercice

Avant de continuer avec **Flexbox** et **Grid**, faisons un exercice sur les menus.

Comment créer un menu ?

- Éléments `<ul>` et `<li>` + `<a>` pour les actions (liens)
- Styles CSS pour l'affichage des différents menus et sous-menus
- Exploitation des pseudo-classes (ex `:hover`) pour faire apparaître/disparaître les sous-menus

Exemple

```
<ul id="menu">
    <li><a href="page1.html" title="1">menu1</a>
    <ul class="sousmenu">
        <li><a href="page11.html" title="11">menu11</a></
         li>
        <li><a href="page12.html" title="12">menu12</a></
         li>
    </ul></li>
    <li><a href="page2.html" title="2">menu2</a></li>
    <li><a href="page3.html" title="2">menu3</a></li>
    <li><a href="page4.html" title="4">menu4</a>
    <ul class="sousmenu">
        <li><a href="page41.html" title="41">menu41</a></
         li>
        <li><a href="page42.html" title="42">menu42</a></
         li>
    </ul></li>
</ul>
```

Les menus

Exercice, en se basant sur le code HTML permettant de créer la liste (et sous-liste) précédente

- écrire un premier code CSS permettant de créer un menu vertical
- écrire un deuxième code CSS permettant de créer un menu horizontal

Solution possible pour un menu vertical

```
#menu{
    list-style-type: none;
}

.sousmenu{
    list-style-type: none;
    display: none;
}

li:hover > ul {
    display: list-item;
}
```

Emplacement de boites avec Flexbox

Considérons l'exemple suivant :

Le fichier `flex.html`

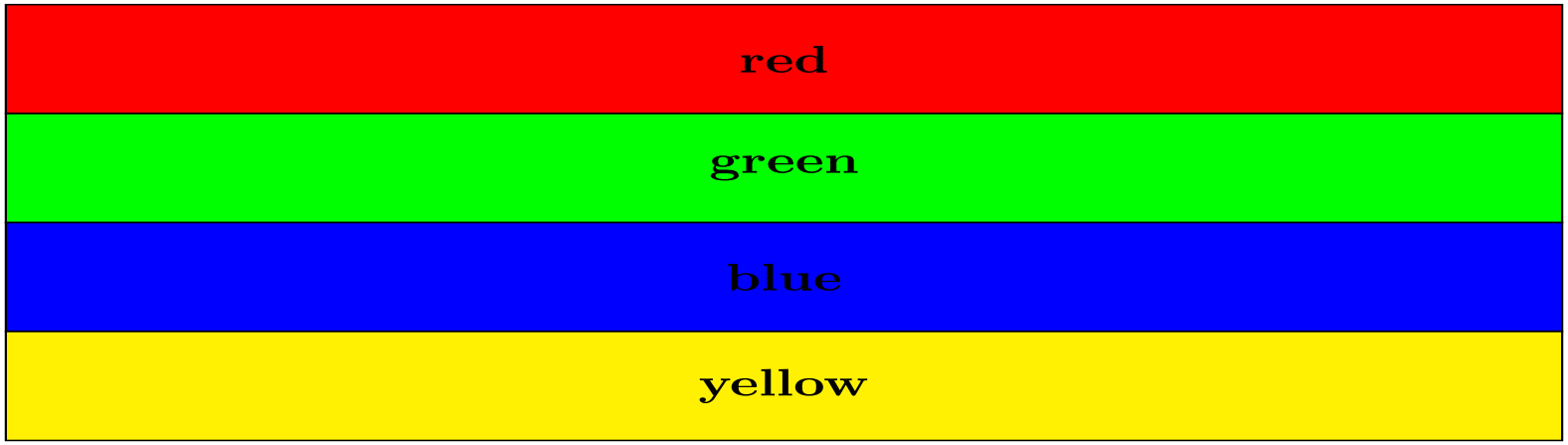
```
<!DOCTYPE html>
<html>
  <head>
    <title> float et clear </title>
    <link rel="stylesheet" href="flex.css">
  </head>
  <body>
    <div class="container" >
      <div class= component1 >red</div>
      <div class= component2 >green</div>
      <div class= component3 >blue</div>
      <div class= component4 >yellow</div>
    </div>
  </body>
</html>
```

Le fichier `flex.css`

```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
}
```

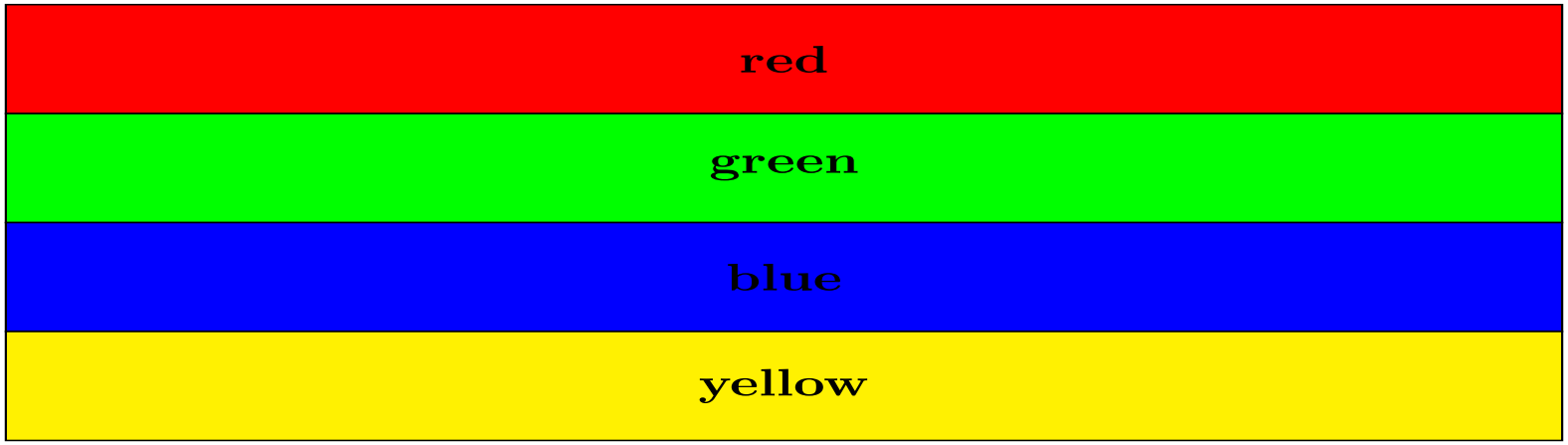
Emplacement de boites avec Flexbox

Le résultat du code précédent :



Emplacement de boites avec Flexbox

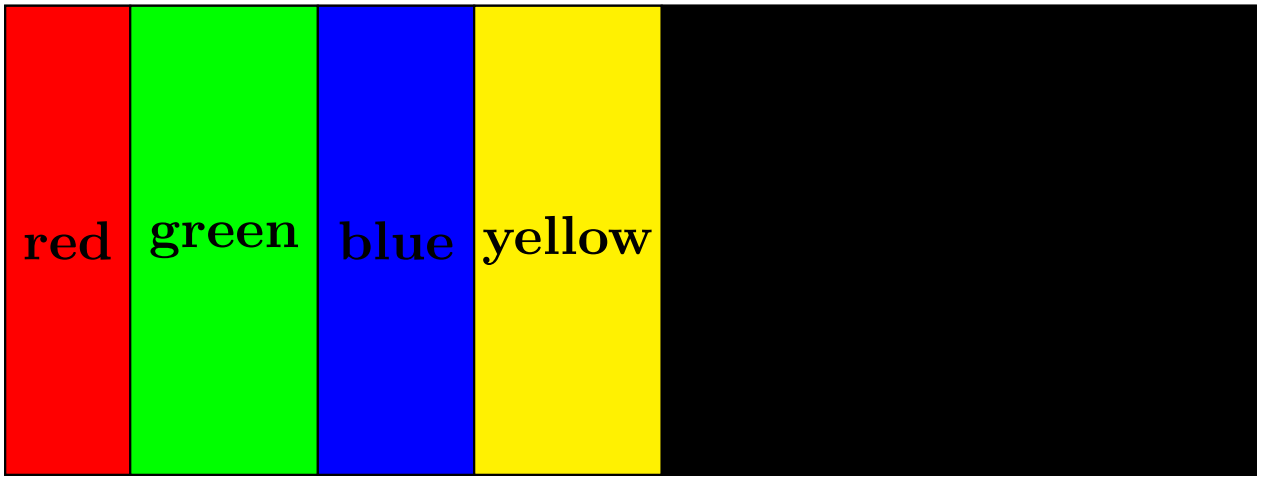
Le résultat du code précédent :



Et si on veut placer les balises `div` les unes à coté des autres ?

Emplacement de boites avec Flexbox

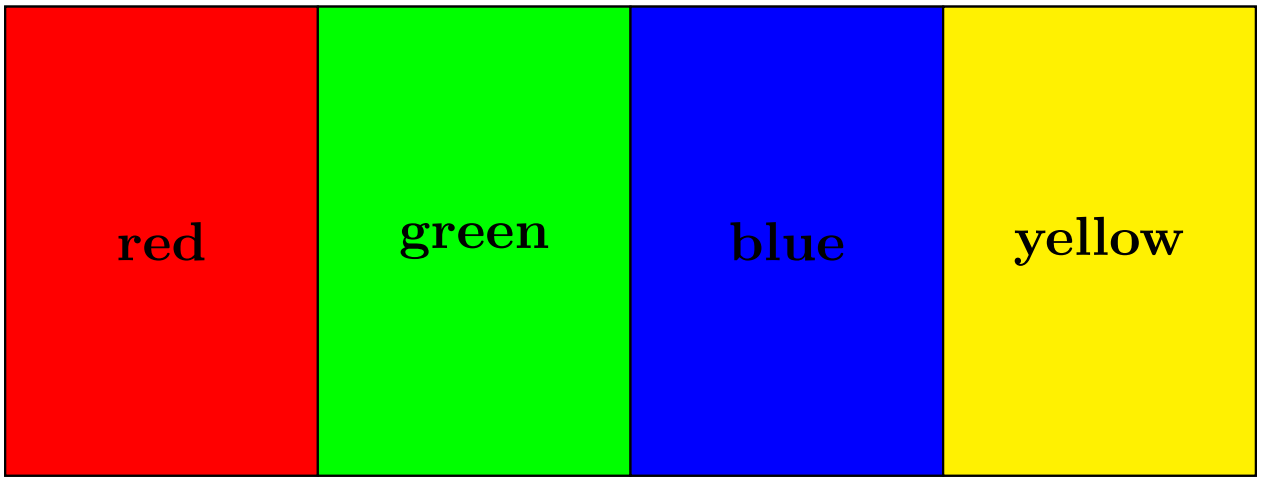
```
.container {
  display: flex;
}
```



Et si on veut donner la même largeur à toutes les `div` ?

Emplacement de boites avec Flexbox

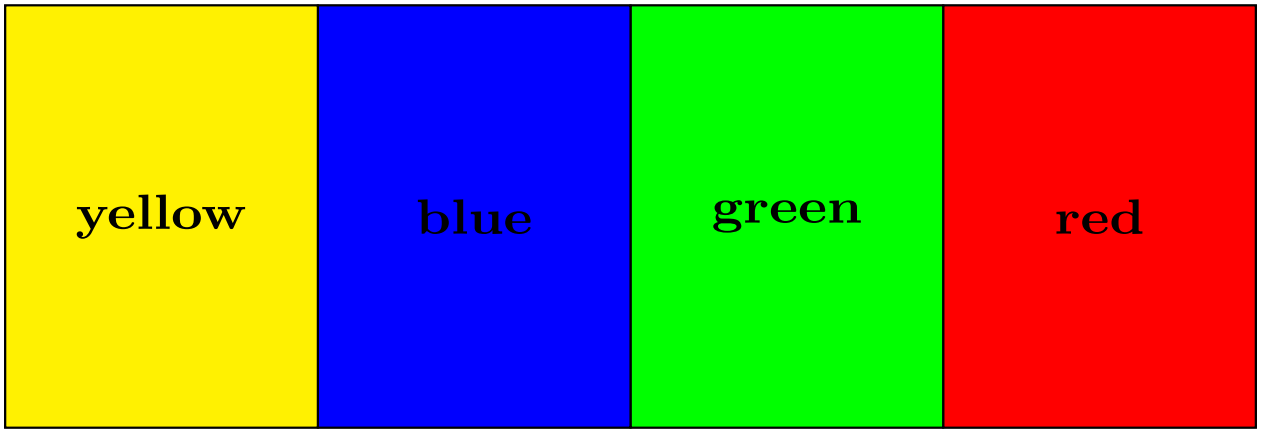
```
.container {
  display: flex;
}
.container > div{
  width: 30%;
}
```



Et si on veut afficher les `div` dans le sens opposé ?

Emplacement de boites avec Flexbox

```
.container {
  background-color: black;
  display: flex;
  flex-direction: row-reverse;
}
```



Emplacement de boites avec Flexbox

Autres valeurs de `flex-direction`

- `row` (par défaut) : organiser les boites sur une ligne.
- `row-reverse` : organiser les boites sur une ligne dans le sens inverse.
- `column` : organiser les boites sur une colonne.
- `column-reverse` : organiser les boites sur une colonne dans le sens inverse.

Emplacement de boites avec Flexbox

Autres valeurs de `flex-direction`

- `row` (par défaut) : organiser les boites sur une ligne.
- `row-reverse` : organiser les boites sur une ligne dans le sens inverse.
- `column` : organiser les boites sur une colonne.
- `column-reverse` : organiser les boites sur une colonne dans le sens inverse.

Remarque

- Les 4 `div` de l'exemple précédent avait chacune une largeur de 30%. Mais **Flexbox** les place, par défaut, sur une seule ligne.
- Et si on veut le forcer à retourner à la ligne si espace insuffisant ? $\Rightarrow$ il faut utiliser la propriété `flex-wrap`.

Emplacement de boites avec Flexbox

La propriété `flex-wrap`

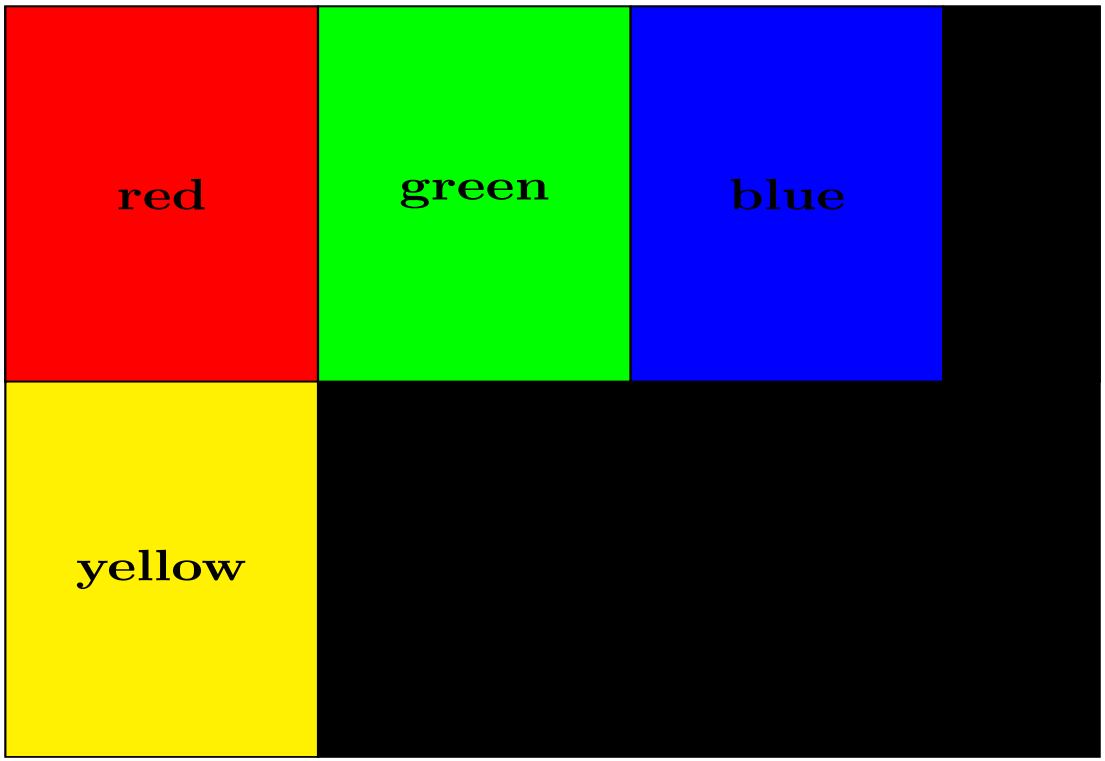
Pour indiquer s'il faut retourner à la ligne si la boite container ne suffit pas

- `nowrap` (par défaut) : organiser les boites sans retour à la ligne
- `wrap` : retourner à la ligne si place insuffisante
- `wrap-reverse` : retourner à la ligne si place insuffisante dans le sens inverse

Emplacement de boites avec Flexbox

Revenons à l'exemple précédent

```
.container {
  background-color: black;
  display: flex;
  flex-wrap: wrap;
}
```



Flexbox : autres propriétés

`flex-flow`

- raccourci de `flex-direction` et `flex-wrap`
- exemple : `flex-flow : row wrap;`

Flexbox : autres propriétés

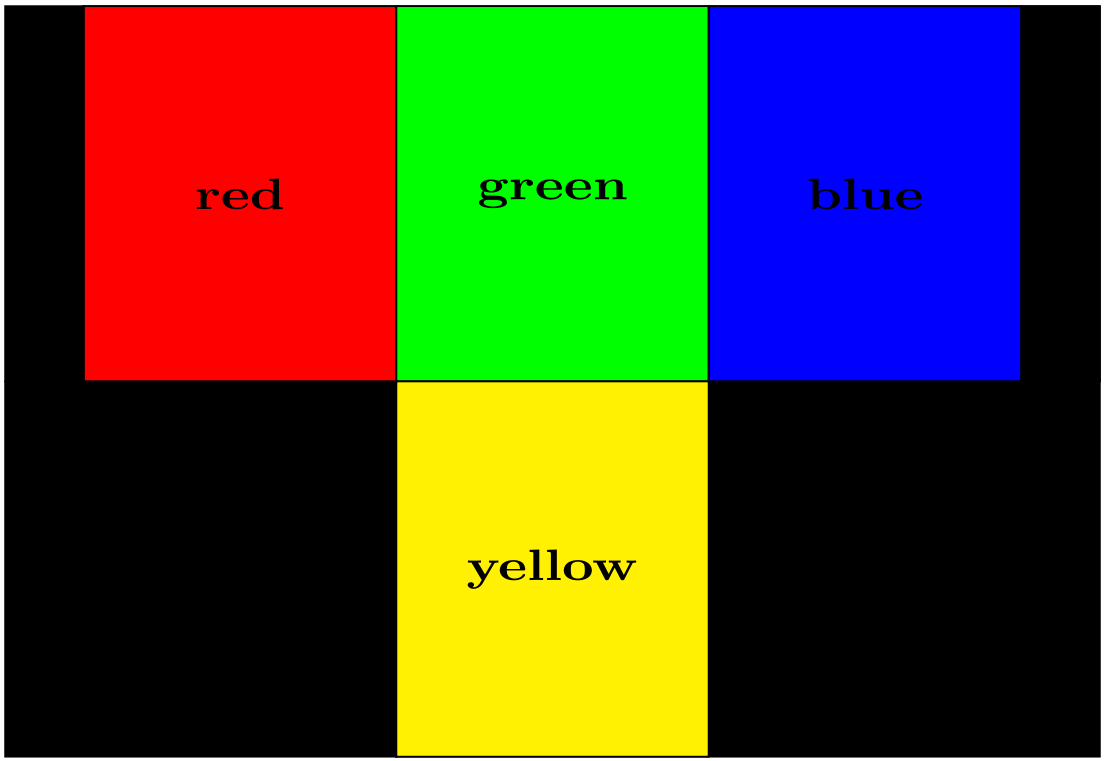
`justify-content` : pour l'alignement des boites (horizontal)

- `center` : pour centrer les éléments dans le conteneur
- `flex-start` : pour aligner les éléments au début du conteneur
- `flex-end` : pour aligner les éléments à l'extrémité gauche du conteneur
- `space-around` : pour ajouter de l'espace autour de chaque élément
- `space-between` : pour ajouter de l'espace entre les éléments

Emplacement de boites avec Flexbox

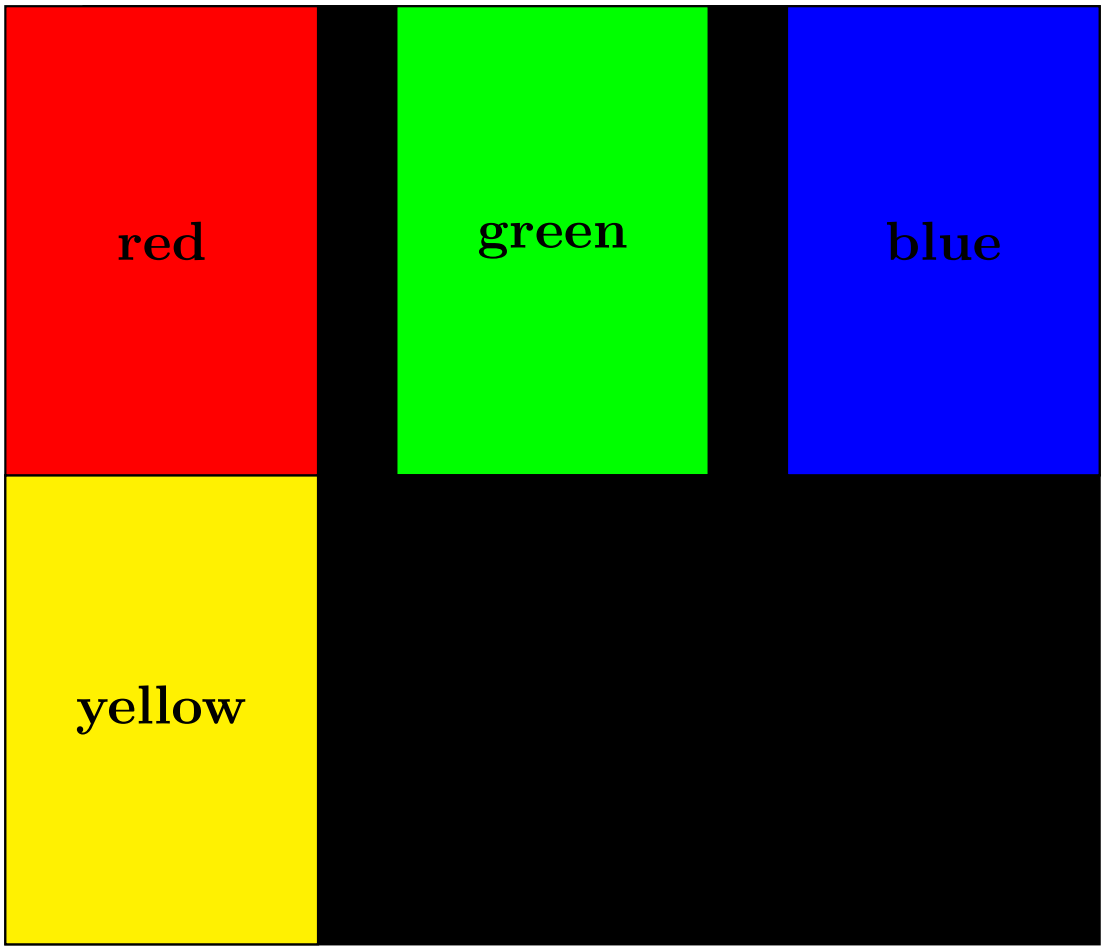
Revenons à l'exemple précédent

```
.container {
  background-color: black;
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
}
```



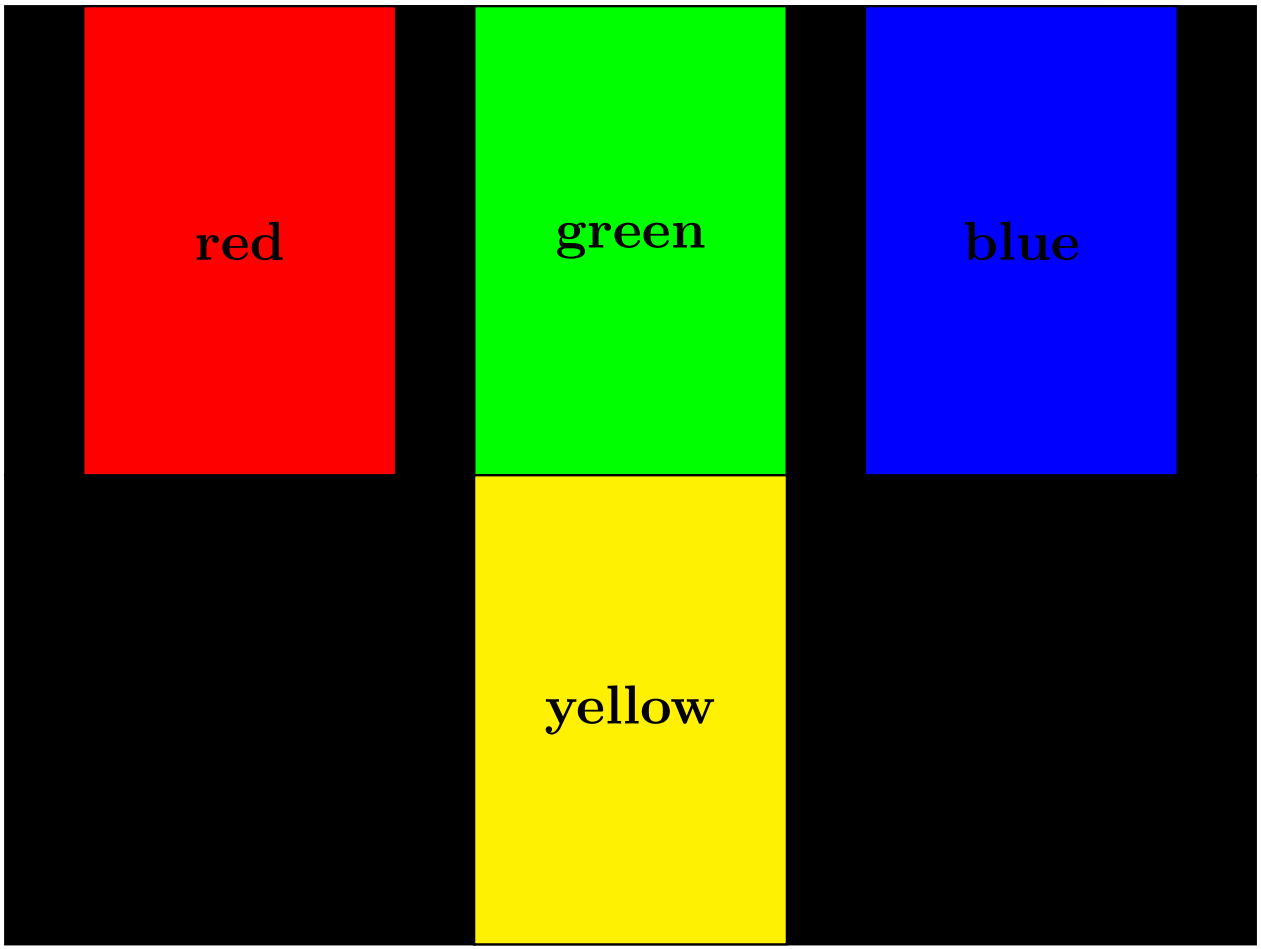
Emplacement de boites avec Flexbox

Exemple avec `justify-content: space-between;`



Emplacement de boites avec Flexbox

Exemple avec `justify-content: space-around;`



Flexbox : autres propriétés

`align-items` : pour l'alignement (vertical)

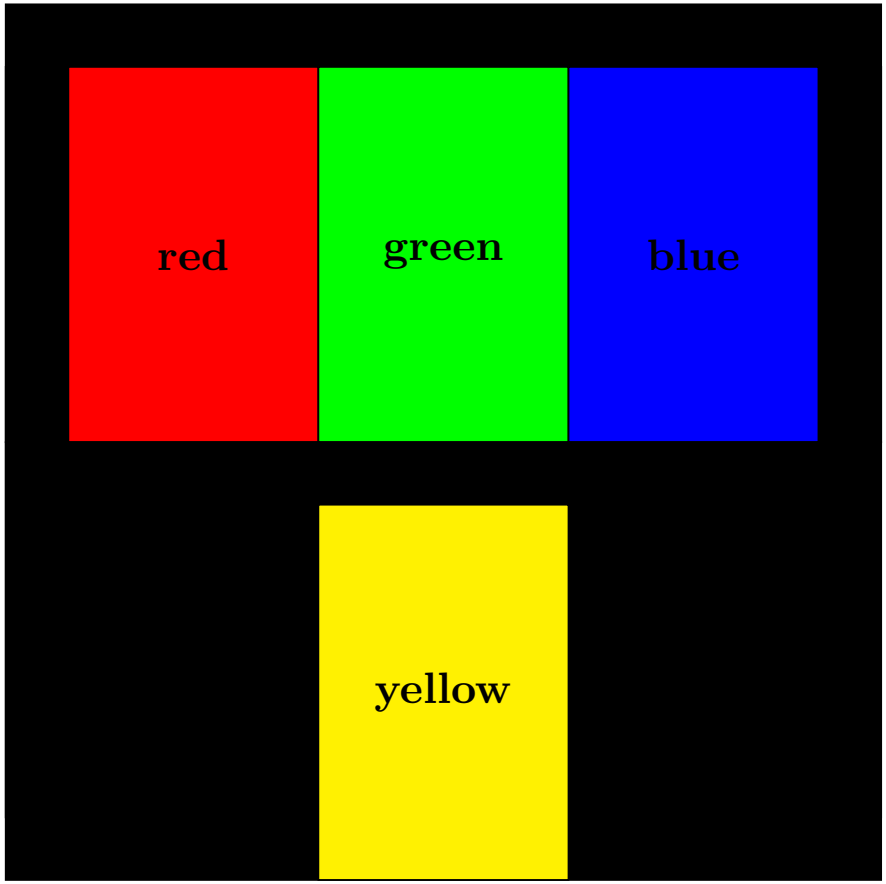
- peut prendre les valeurs `center`, `flex-start` et `flex-end` tout comme `justify-content`
- ou aussi `stretch` (valeur par défaut, rempli le conteneur en hauteur avec les éléments) ou `baseline` (qui aligne les éléments flexibles d'une façon que leurs lignes de base soient alignées)

Exemple avec `align-items: flex-end;`

Le fichier `flex.css`

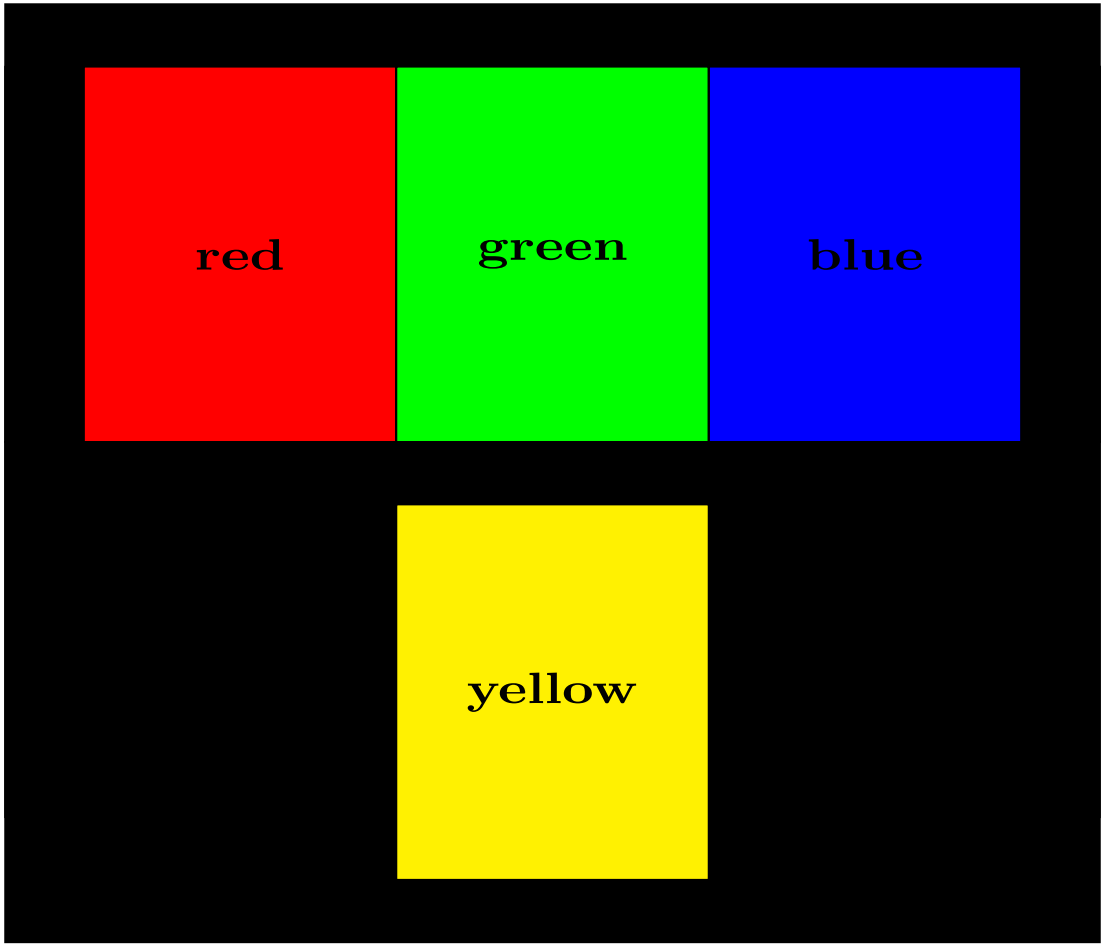
```
.container {
  background-color: black;
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  align-items: flex-end;
  height: 150px;

}
.container > div{
  width : 30%;
  height: 50px;
}
```

Le résultat



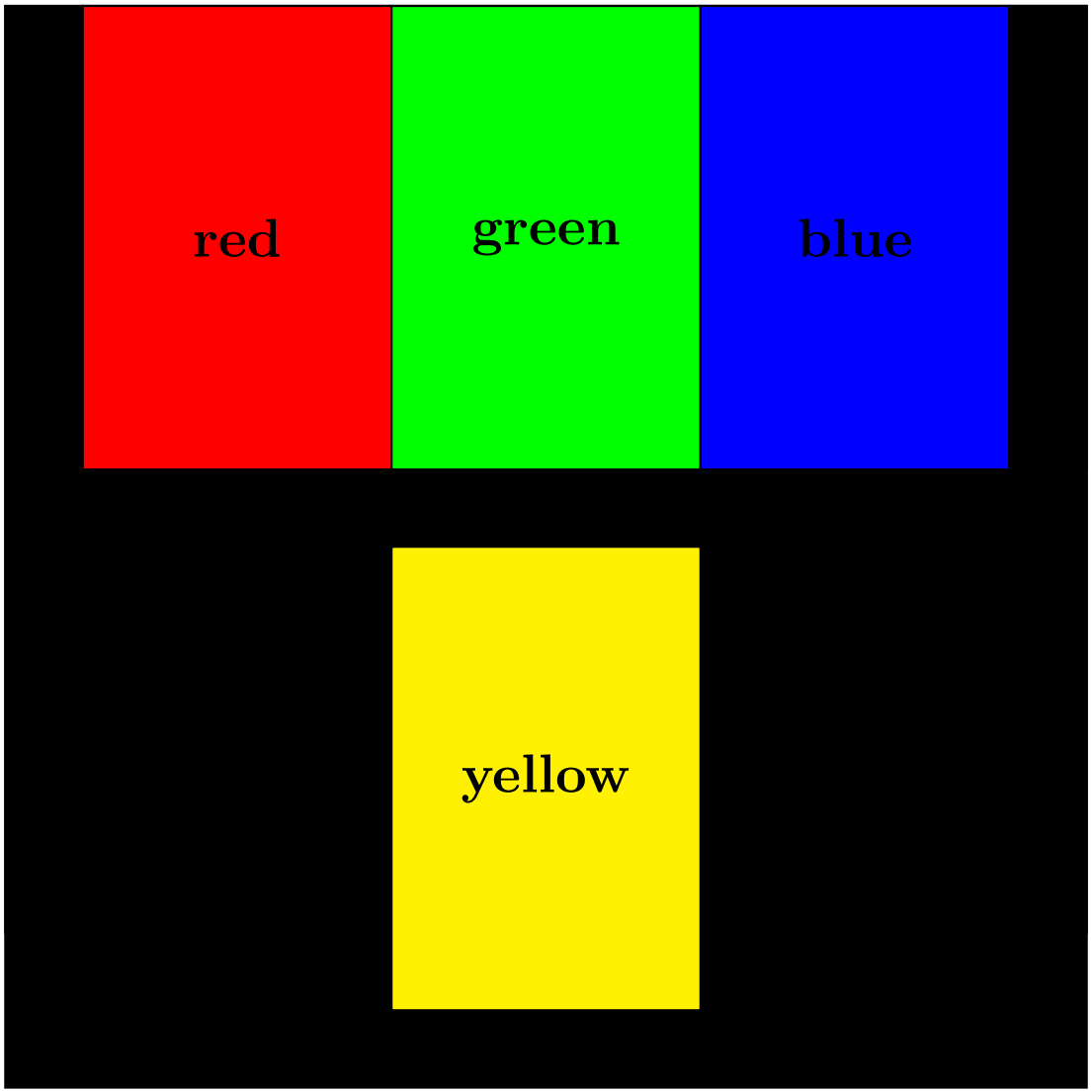
Emplacement de boites avec Flexbox

Exemple avec `align-items: center;`



Emplacement de boites avec Flexbox

Exemple avec `align-items: baseline;`



Flexbox : autres propriétés

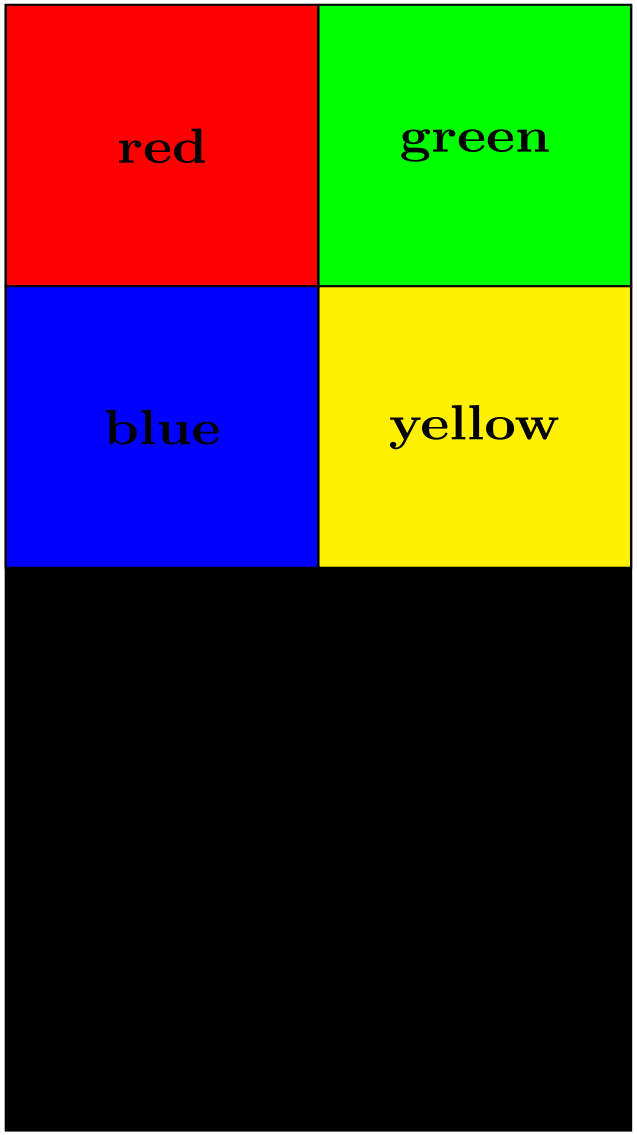
`align-content`

- aligne verticalement les lignes de boites d'un conteneur.
- prend comme valeur `flex-start`, `flex-end`, `center`, `space-between`, `space-around`, `stretch`...
- n'a aucun effet lorsque tous les éléments sont sur une seule ligne

Le fichier `flex.css`

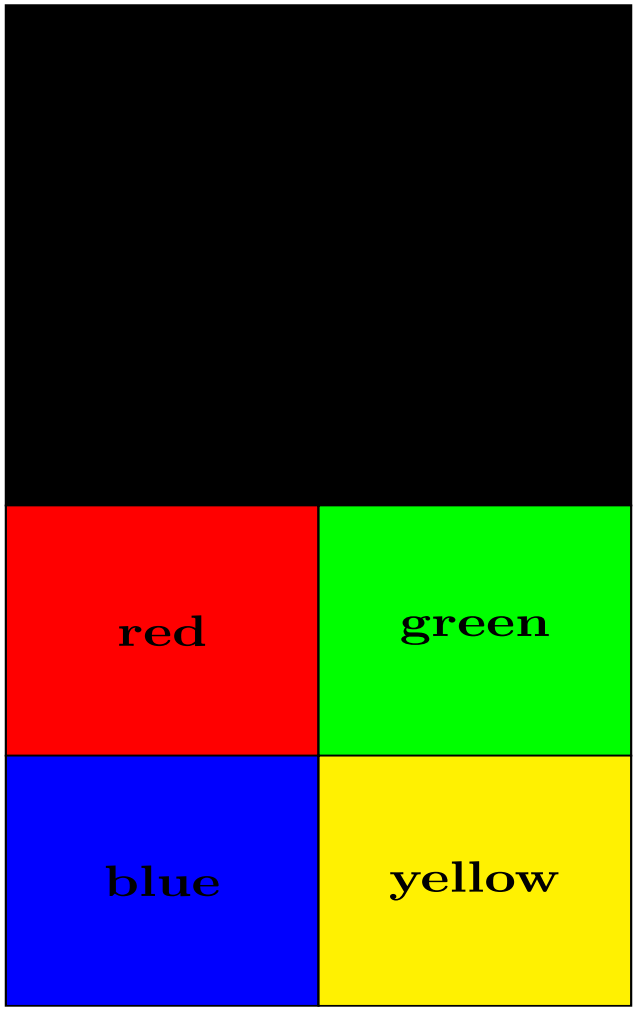
```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: flex;
  flex-wrap: wrap;
  align-content: flex-start;
  height: 150px;
}
.container > div{
  width : 50%;
}
```

Le résultat



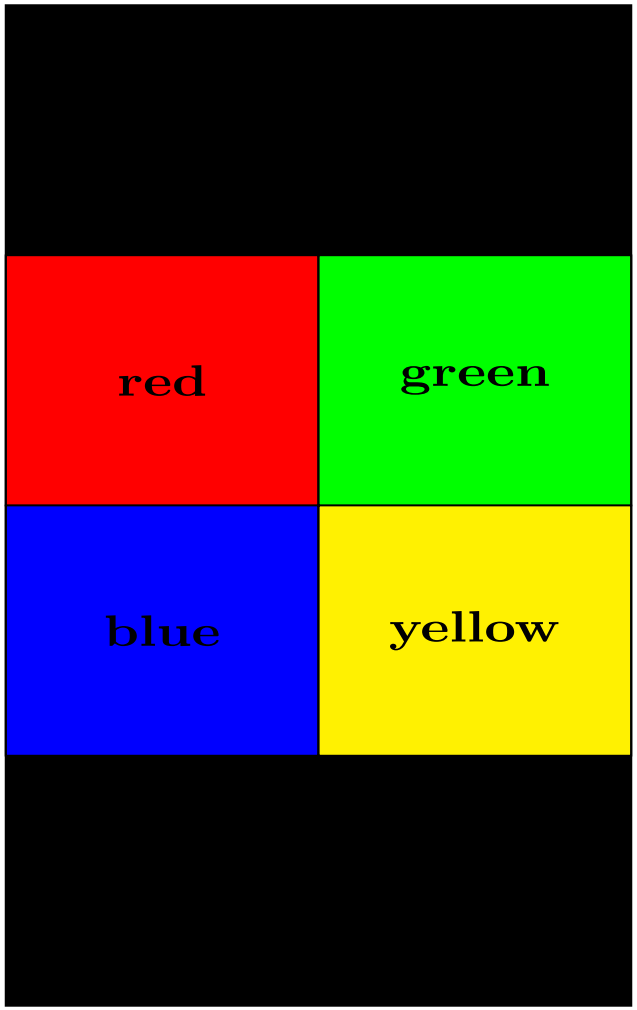
Emplacement de boites avec Flexbox

Exemple avec `align-content: flex-end;`



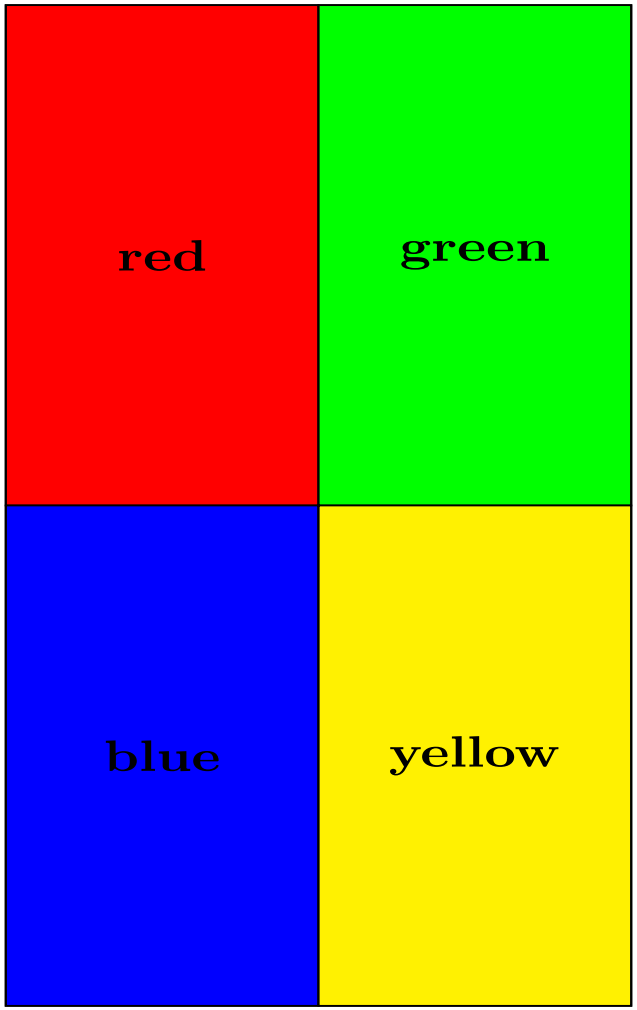
Emplacement de boites avec Flexbox

Exemple avec `align-content: center;`



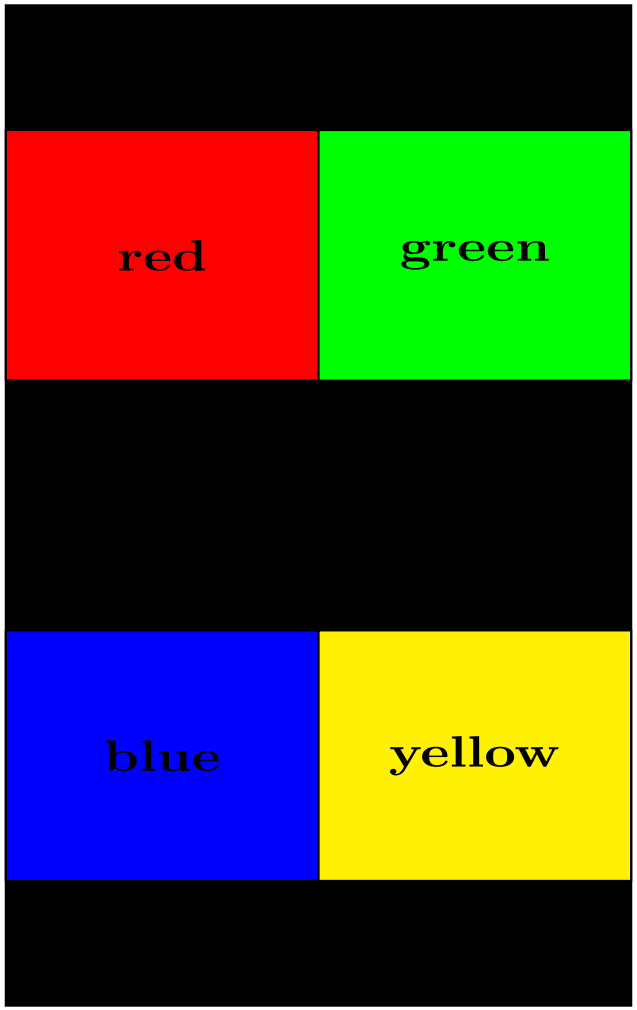
Emplacement de boites avec Flexbox

Exemple avec `align-content: stretch;`



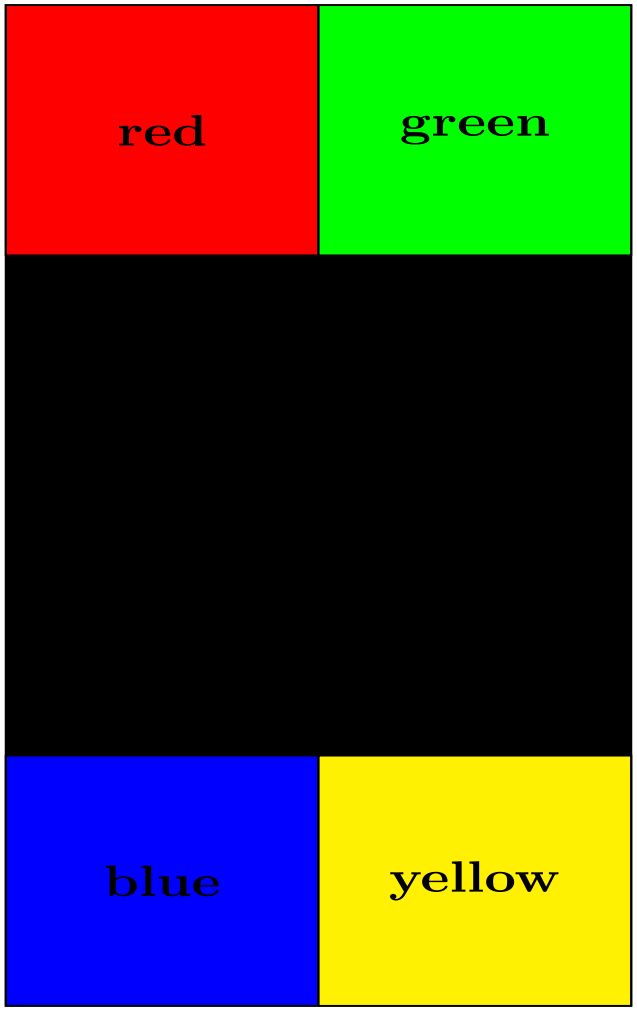
Emplacement de boites avec Flexbox

Exemple avec `align-content: space-around;`



Emplacement de boites avec Flexbox

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



Flexbox : autres propriétés

Propriétés pour les éléments enfants

Les propriétés précédentes sont à mettre dans le style du container. Mais il existe des propriétés qu'on peut préciser dans le style de chaque enfant

Flexbox : autres propriétés

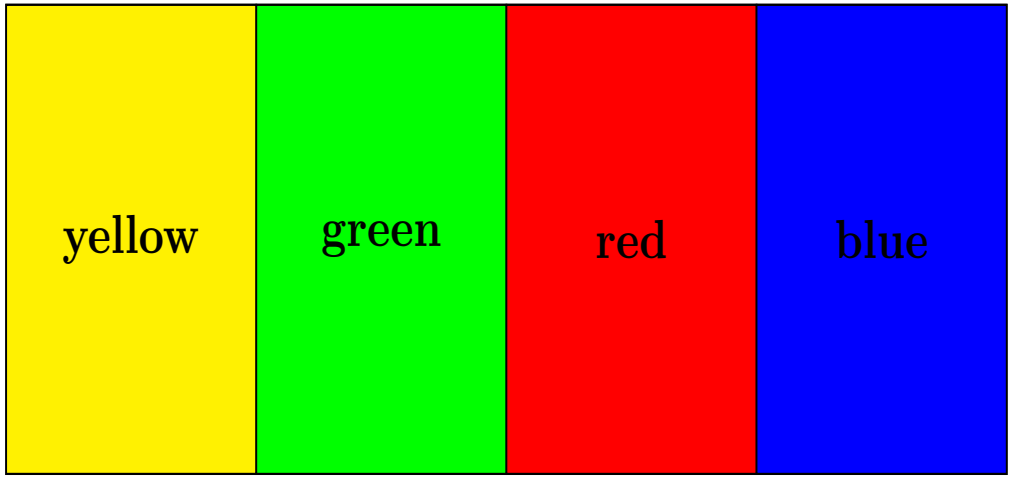
`order`

- permet de changer l'ordre d'affichage des composants d'un conteneur

Le fichier `flex.css`

```
.component1 {
  background-color: red;
  order:3;
}
.component2 {
  background-color: green;
  order: 2;
}
.component3 {
  background-color: blue;
  order:4;
}
.component4 {
  background-color: yellow;
  order:1;
}
.container {
  background-color: black;
  display: flex;
}
.container > div{
        width : 30%;
}
```

Le résultat



Flexbox : autres propriétés

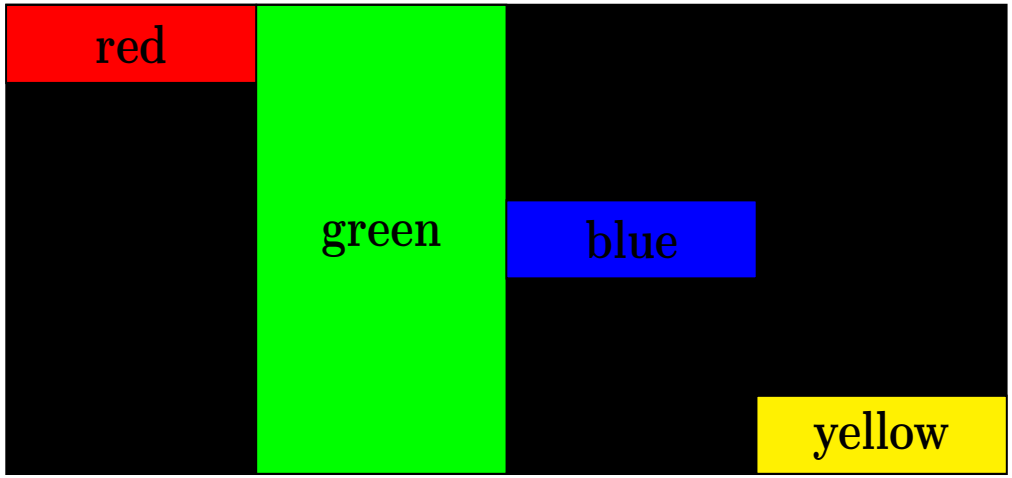
`align-self`

- permet de changer la valeur d'`align-items` pour un block donné.

Le fichier `flex.css`

```
.component1 {
  background-color: red;
  align-self: flex-start;
}
.component2 {
  background-color: green;
  align-self: stretch;
}
.component3 {
  background-color: blue;
  align-self: center;
}
.component4 {
  background-color: yellow;
  align-self: flex-end;
}
.container {
  background-color: black;
  display: flex;
  height: 150px;
}
.container > div{
        width : 30%;
}
```

Le résultat



Flexbox : autres propriétés

Propriétés pour modifier la largeur

- `flex-grow` : permet d'agrandir certains blocks par rapport à la taille des autres
- `flex-shrink` : permet de rétrécir certains blocks si nécessaire lorsqu'il n'y a pas assez d'espace dans une rangée
- `flex-basis` : permet de spécifies la largeur initiale d'un composant, avant qu'un éventuel espace disponible soit distribué selon les facteurs flex
- ...

Flexbox : autres propriétés

`flex-grow`

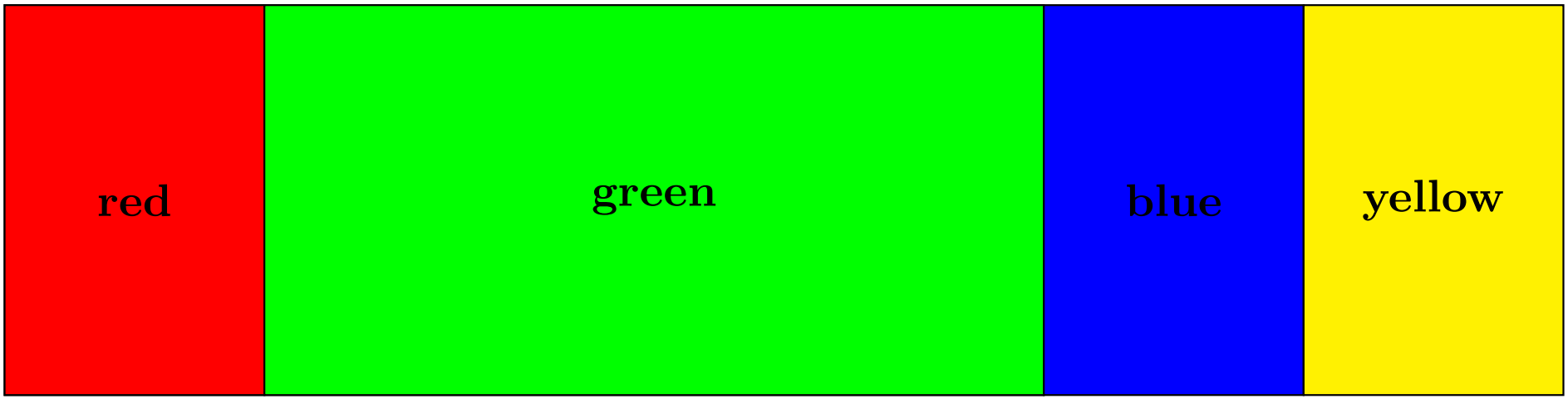
- précise de combien croître la taille des enfants

Le fichier `flex.css`

```
.component1 {
  background-color: red;
  flex-grow: 1;
}
.component2 {
  background-color: green;
  flex-grow: 3;
}
.component3 {
  background-color: blue;
  flex-grow: 1;
}
```

```
.component4 {
  background-color: yellow;
  flex-grow: 1;
}
.container {
  background-color: black;
  display: flex;
}
```

Le résultat



Emplacement de boites avec Flexbox

Pour mieux comprendre Flexbox

- http://flexboxfroggy.com/#fr
- http://www.flexboxdefense.com/

Emplacement de boites avec Grid

Considérons l'exemple suivant :

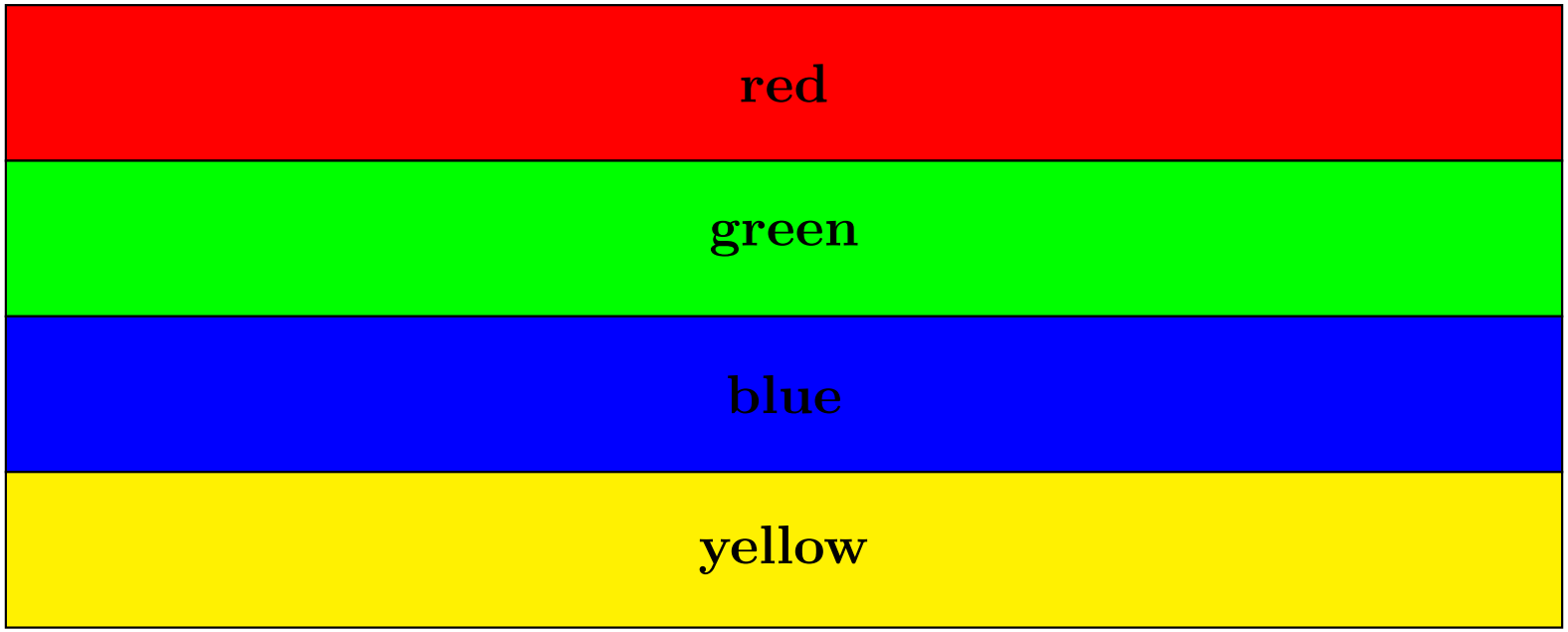
Le fichier `grid.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title> float et clear </title>
    <link rel="stylesheet" href="grid.css">
  </head>
  <body>
    <div class="container" >
      <div class= component1 >red</div>
      <div class= component2 >green</div>
      <div class= component3 >blue</div>
      <div class= component4 >yellow</div>
    </div>
  </body>
</html>
```

Le fichier `grid.css`

```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: grid;
}
.container > div{
  height: 100px;
}
```

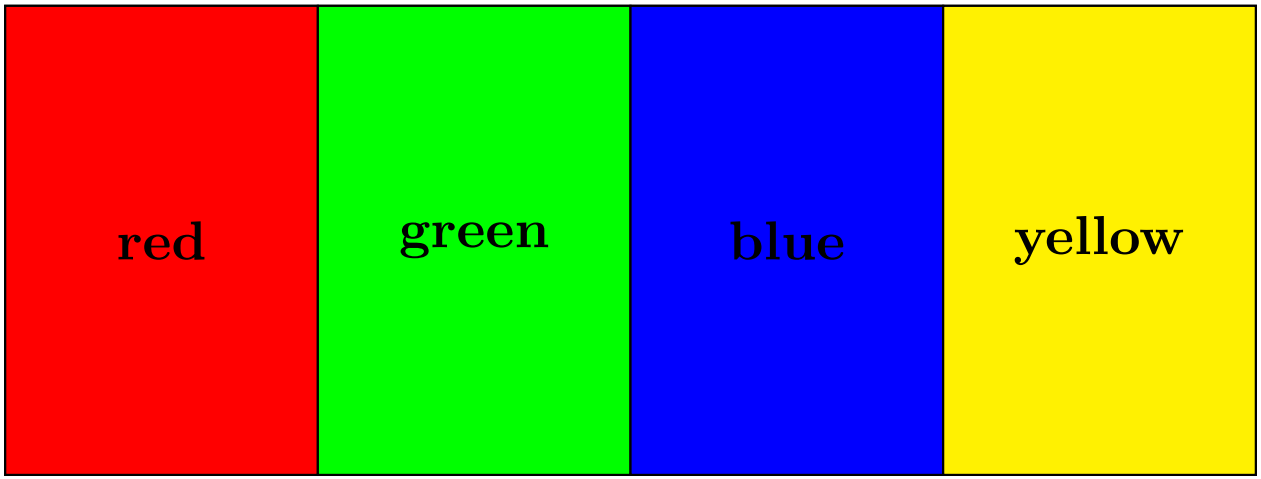
Emplacement de boites avec Grid



Emplacement de boites avec Grid

Définissons une grille composée de 4 colonnes

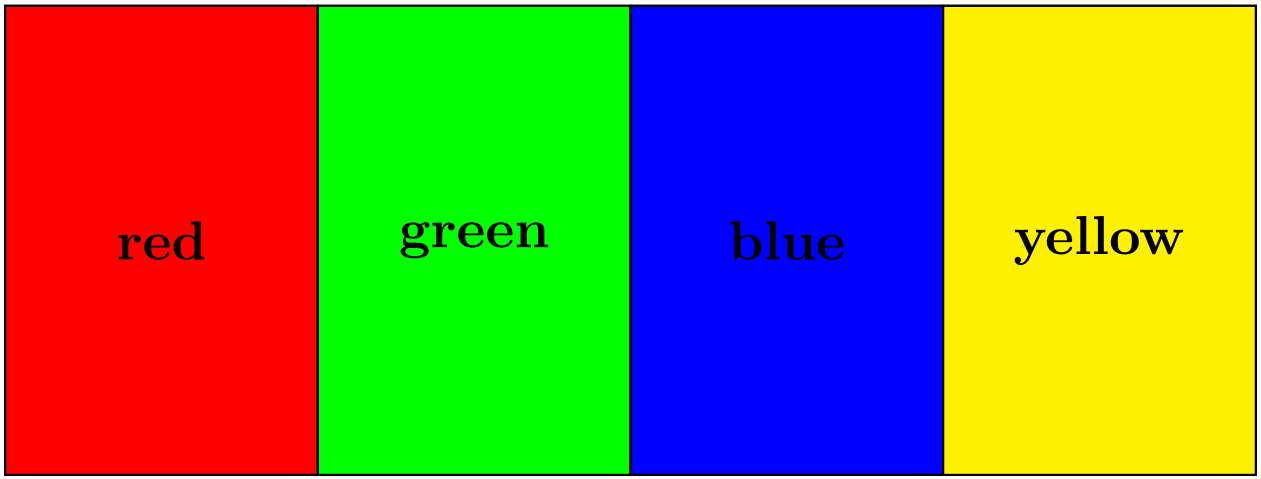
```
.container {
  background-color: black;
  display: grid;
  grid-template-columns: auto auto auto auto;
}
.container > div{
  height: 100px;
}
```



Emplacement de boites avec Grid

Définissons une grille composée de 4 colonnes

```
.container {
  background-color: black;
  display: grid;
  grid-template-columns: auto auto auto auto;
}
.container > div{
  height: 100px;
}
```

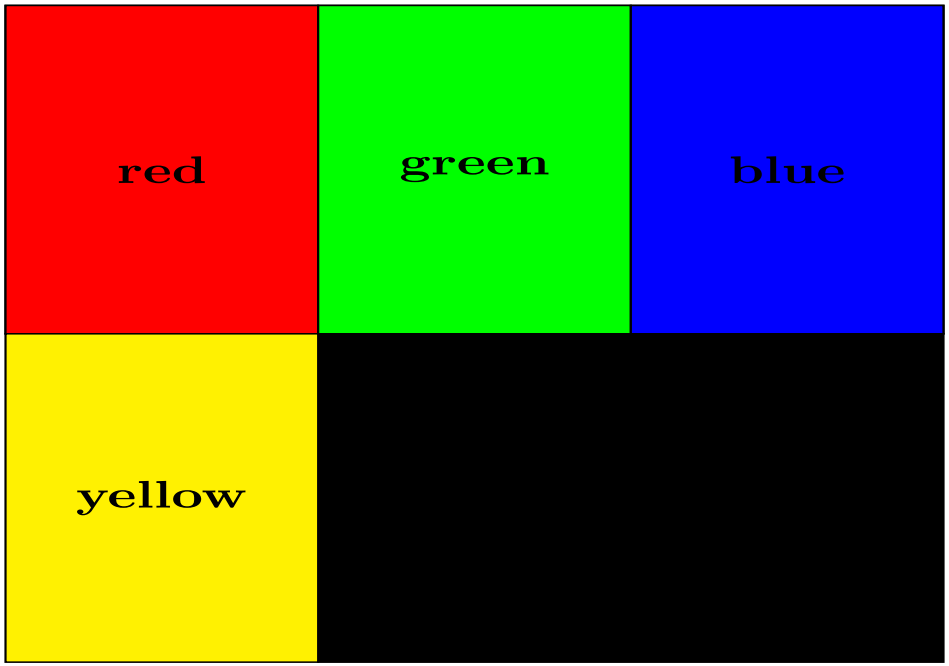


Si le nombre de composants dépasse le nombre de colonnes, alors la grille ajoutera automatiquement une ligne pour placer les composants manquants.

Emplacement de boites avec Grid

Exemple avec une grille composée de 3 colonnes

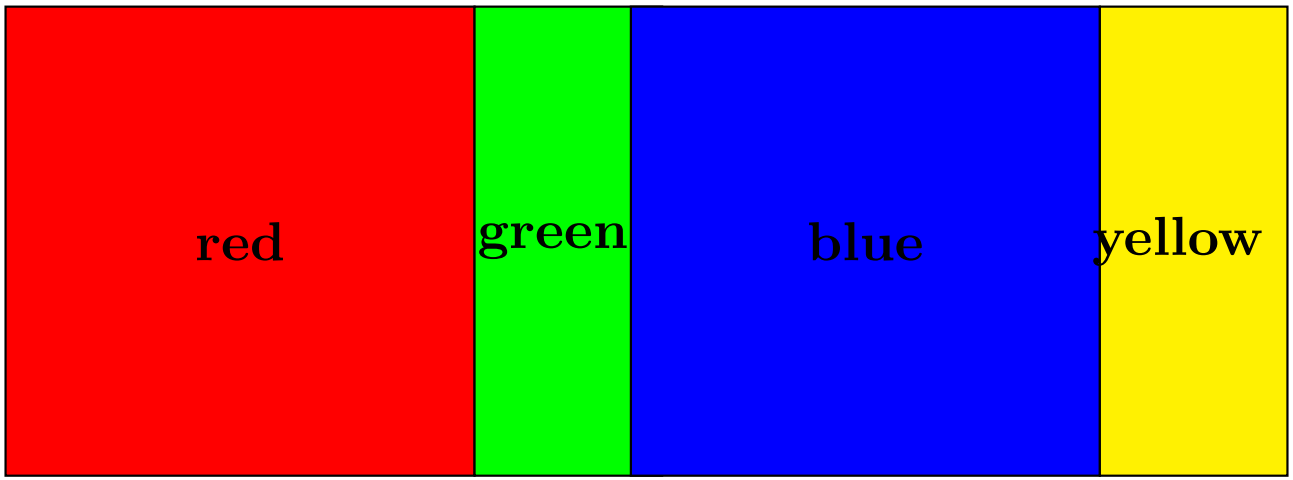
```
.container {
  background-color: black;
  display: grid;
  grid-template-columns: auto auto auto;
}
.container > div{
  height: 100px;
}
```



Emplacement de boites avec Grid

On peut aussi utiliser le nombre de fractions : nombre de part de l'espace disponible

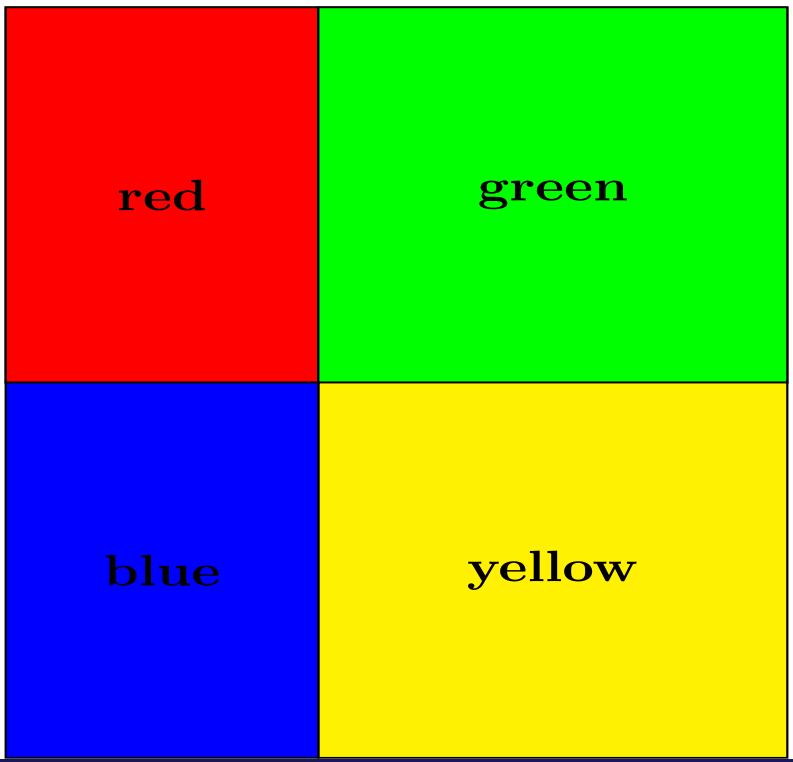
```
.container {
  background-color: black;
  display: grid;
  grid-template-columns: 1.5fr 0.5fr 1.5fr 0.5fr;
}
.container > div{
  height: 100px;
}
```



Cette écriture `grid-template-columns: 1.5fr 0.5fr 1.5fr 0.5fr;` à `grid-template-columns: repeat(2, 1.5fr 0.5fr);`.

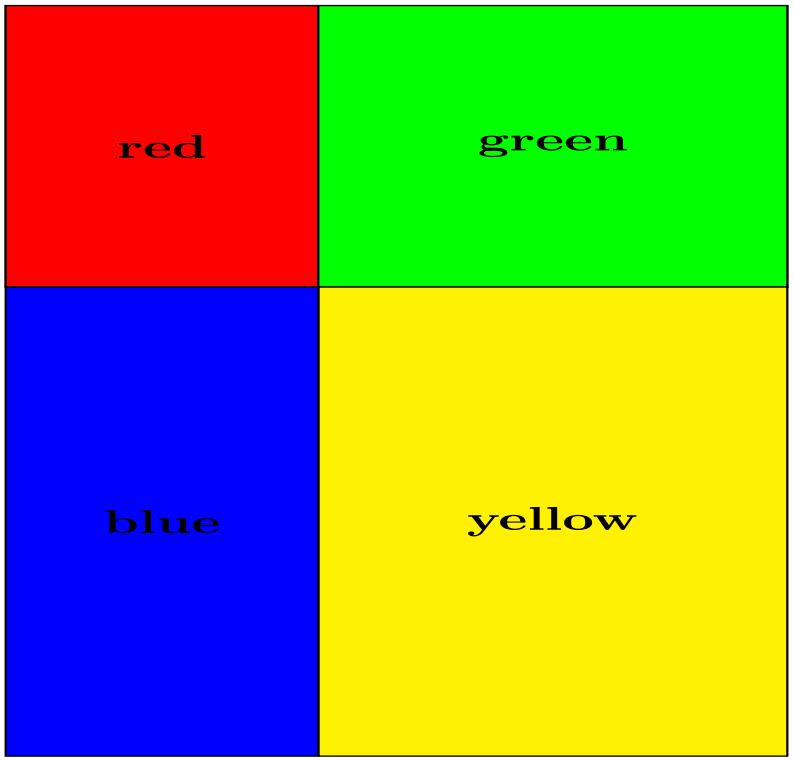
Exemple avec une grille composée de 2 colonnes avec deux largeurs différentes

```
.container {
  background-color: black;
  display: grid;
  grid-template-columns: 40% 60%; // on peut aussi utiliser px : pixel
   ou fr : fraction : nombre de part de l'espace disponible
}
.container > div{
  height: 100px;
}
```



On peut aussi définir la hauteur de chaque rangée (ligne)

```
.container {
  background-color: black;
  display: grid;
  grid-template-columns: 40% 60%;
  grid-template-rows: 100px 200px;
}
```



Grid : autres propriétés

Grid : autres propriétés

- `grid-column-gap` : permet de définir un espace entre les colonnes
- `grid-row-gap` : permet de définir un espace entre les lignes
- `justify-content` : permet de définir la façon dont l'espace doit être réparti entre et autour des composants par rapport à un axe vertical
- `align-content` : permet de définir la façon dont l'espace doit être réparti entre et autour des composants par rapport à un axe horizontal

On peut aussi modifier les propriétés des composants

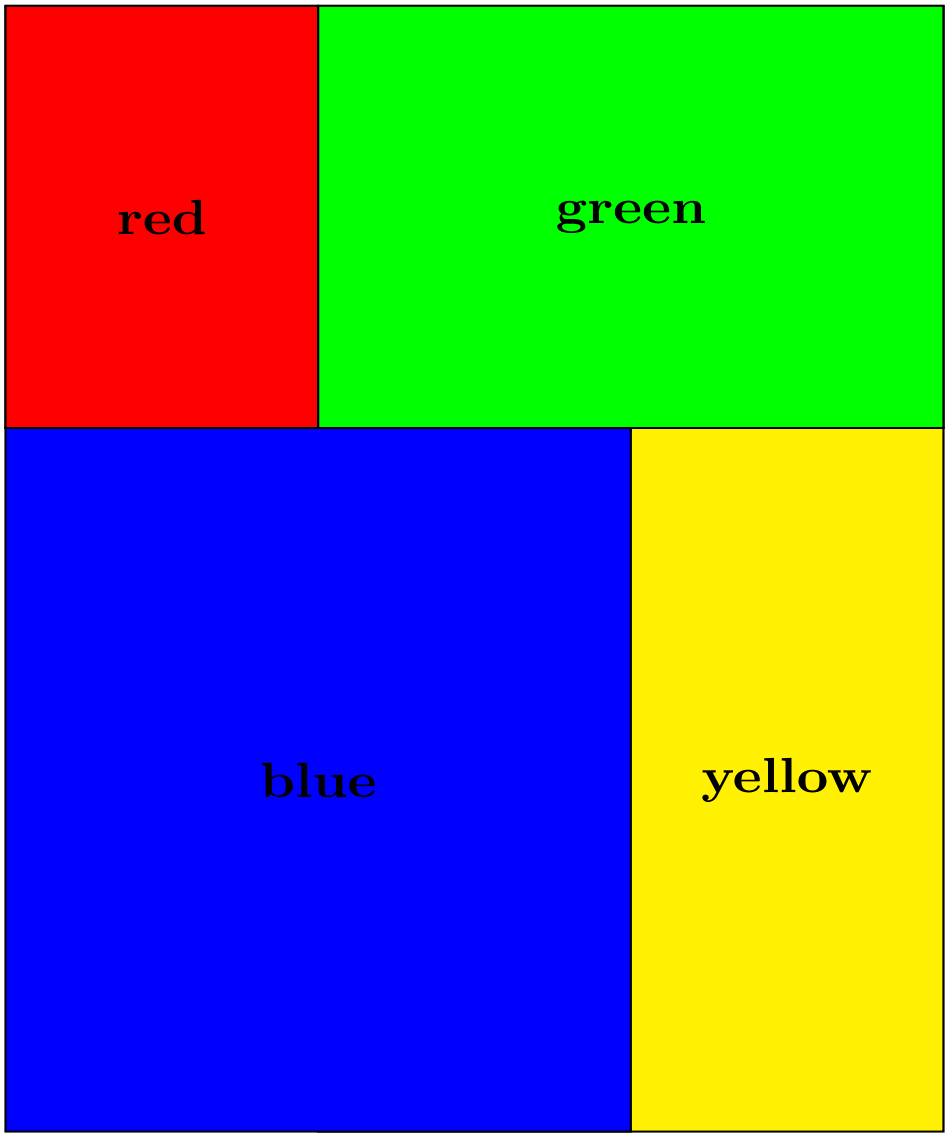
Le fichier `grid.css`

```
.component1 {
  background-color: red;
  grid-column-start: 1;
  grid-column-end: 3;
  grid-row-start: 1;
  grid-row-end: 3;
}
.component2 {
  background-color: green;
  grid-column-start: 2;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 2;
}
.component3 {
  background-color: blue;
  grid-column-start: 1;
  grid-column-end: 3;
  grid-row-start: 2;
  grid-row-end: 4;
}
```

```
.component4 {
  background-color: yellow;
  grid-column-start: 3;
  grid-column-end: 4;
  grid-row-start: 2;
  grid-row-end: 4;
}
.container {
  background-color: black;
  display : grid;
  height: 500px;
  grid-template-columns: repeat(3, 1fr);
}
```

Emplacement de boites avec Grid

Le résultat est :



On peut aussi utiliser la propriété `span` qui indique le nombre de colonnes (lignes)

Le fichier `grid.css`

```
.component1 {
  background-color: red;
  grid-column-start: 1;
  grid-column-end: span 2;
  grid-row-start: 1;
  grid-row-end: span 2;
}
.component2 {
  background-color: green;
  grid-column-start: 2;
  grid-column-end: span 2;
  grid-row-start: 1;
  grid-row-end: 2;
}
.component3 {
  background-color: blue;
  grid-column-start: 1;
  grid-column-end: span 2;
  grid-row-start: 2;
  grid-row-end: span 2;
}
```

```
.component4 {
  background-color: yellow;
  grid-column-start: 3;
  grid-column-end: 4;
  grid-row-start: 2;
  grid-row-end: span 2;
}
.container {
  background-color: black;
  height: 500px;
  display : grid;
  grid-template-columns: repeat(3, 1fr);
}
```

Emplacement de boites avec Grid

On peut encore le simplifier

```
.component1 {
  background-color: red;
  grid-column: 1 / span 2;
  grid-row: 1 / span 2;
}
.component2 {
  background-color: green;
  grid-column: 2 / span 2;
  grid-row: 1  / span 1;
}
.component3 {
  background-color: blue;
  grid-column: 1 / span 2;
  grid-row: 2 / span 2;
}
.component4 {
  background-color: yellow;
  grid-column: 3 / span 1;
  grid-row: 2 / span 2;
}
```

Emplacement de boites avec Grid

On peut encore le simplifier

```
.component1 {
  background-color: red;
  grid-column: 1 / span 2;
  grid-row: 1 / span 2;
}
.component2 {
  background-color: green;
  grid-column: 2 / span 2;
  grid-row: 1  / span 1;
}
.component3 {
  background-color: blue;
  grid-column: 1 / span 2;
  grid-row: 2 / span 2;
}
.component4 {
  background-color: yellow;
  grid-column: 3 / span 1;
  grid-row: 2 / span 2;
}
```

On peut utiliser la propriété `z-index` pour définir l'ordre dans lequel les composants s'empilent.

On peut aussi utiliser les area

```
.component1 {
  background-color: red;
  grid-area: rouge;
}
.component2 {
  background-color: green;
  grid-area: vert;
}
.component3 {
  background-color: blue;
  grid-area: bleu;
}
.component4 {
  background-color: yellow;
  grid-area: jaune;
}
.container {
  height: 500px;
  display : grid;
  grid-template-areas:
  'red green green'
  'blue blue yellow'
  'blue blue yellow';
}
```

Emplacement de boites avec Grid

Pour mieux comprendre Grid

- http://cssgridgarden.com/#fr

Media Queries

C'est quoi ? !

définit les techniques CSS d'adaptation du contenu de notre page web en fonction de la résolution d'écran de nos visiteurs

Media Queries

C'est quoi ? !

définit les techniques CSS d'adaptation du contenu de notre page web en fonction de la résolution d'écran de nos visiteurs

```
<head>
    <meta charset="utf-8">
    <title>Affichage selon resolution</title>
    <link rel="stylesheet" media="screen" href="file.css" type="text/
     css" />
    <link rel="stylesheet" media="print" href="print.css" type="text/
     css" />
</head>
```

Media Queries

C'est quoi ? !

définit les techniques CSS d'adaptation du contenu de notre page web en fonction de la résolution d'écran de nos visiteurs

```
<head>
    <meta charset="utf-8">
    <title>Affichage selon resolution</title>
    <link rel="stylesheet" media="screen" href="file.css" type="text/
     css" />
    <link rel="stylesheet" media="print" href="print.css" type="text/
     css" />
</head>
```

Depuis CSS2

On avait la possibilité de définir une version :

- `screen` : pour les écrans
- `print` : pour impression

Media Queries

Dans le fichier print.css

```
#menu, #footer, #header {
    display:none;
}
body {
    font-size:80%;
}
```

Media Queries

Tout mettre dans un fichier ?

Il faut juste préciser chaque fois le type de périphérique concerné par le style.

```
@media print {
    #menu, #footer, #header {
        display:none;
    }
    body {
        font-size:80%;
    }
}
@media screen
{
    ...
}
```

Media Queries

Depuis CSS2, autres types de média disponibles

- `handheld` : Périphériques mobiles ou de petite taille
- `projection` : Projecteurs (ou présentations avec slides)
- `tv` : Téléviseur
- `all` : Toutes les résolutions
- ...

Plusieurs navigateurs mobiles ignorent le media `handheld` (par exemple Safari Mobile) et se considèrent comme un média screen.

Media Queries

Depuis CSS3, on peut définir des règles

- `height` : hauteur de la zone d'affichage
- `width` : largeur de la zone d'affichage
- `device-height` : hauteur du périphérique
- `resolution` résolution du périphérique
- `max-height` : hauteur maximale de la zone d'affichage
- `max-width` : largeur maximale de la zone d'affichage
- `min-height` : hauteur minimale de la zone d'affichage
- `min-width` : largeur minimale de la zone d'affichage
- `orientation` : orientation du périphérique (portrait ou paysage)
- ...

Media Queries

```
<link rel="stylesheet" media="(orientation:portrait)
 " href="file1.css">

<link rel="stylesheet" media="(orientation:landscape
 )" href="file2.css">
...
```

Media Queries

Depuis CSS3, il est possible de combiner ces règles

- `and` : et
- `only` : seulement
- `not` : non

Media Queries

```
<link rel="stylesheet" media="screen␣and␣(max-width:␣640px)" href="
 file1.css" />

<link rel="stylesheet" media="only␣screen␣and␣print" href="file2.css" /
 >
...
```

Media Queries

```
<link rel="stylesheet" media="screen and (max-width: 640px)" href="
 file1.css" />

<link rel="stylesheet" media="only screen and print" href="file2.css" /
 >
...
```

ou dans un seul fichier CSS :

```
@media screen and (max-width: 1024px){

}
@media all and (min-width: 1024px) and (max-width: 1280px){

}
@media projection and tv{

}
@media all and (orientation: portrait){

}
```

Bootstrap

C'est quoi Bootstrap ?

Un framework CSS, open source depuis 2012, conçue par des développeurs `Twitter`, Mark Otto et Jacob Thornton.

Bootstrap

C'est quoi Bootstrap ?

Un framework CSS, open source depuis 2012, conçue par des développeurs `Twitter`, Mark Otto et Jacob Thornton.

Documentation officielle

`http://getbootstrap.com/`

Bootstrap

C'est quoi Bootstrap ?

Un framework CSS, open source depuis 2012, conçue par des développeurs `Twitter`, Mark Otto et Jacob Thornton.

Documentation officielle

`http://getbootstrap.com/`

Documentation sur w3schools

`https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp`

Bootstrap

Deux façons différentes pour intégrer Bootstrap

- en faisant référence à un fichier `bootstrap.min.css`, situé sur le site de Bootstrap, dans la balise `link` (CDN)
- téléchargeant Bootstrap et en faisant référençant `bootstrap.min.css`

Bootstrap

Deux façons différentes pour intégrer Bootstrap

- en faisant référence à un fichier `bootstrap.min.css`, situé sur le site de Bootstrap, dans la balise `link` (CDN)
- téléchargeant Bootstrap et en faisant référençant `bootstrap.min.css`

Commençons par l'intégrer dans notre page HTML

```
<link rel="stylesheet" href="https://maxcdn.
 bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.
 css">
```

Bootstrap

La grille de Bootstrap ($\neq$ `Grid`)

- Bootstrap utilise une grille composée de 12 colonnes pour chaque ligne (`row`)
- En cas d'imbrication, la nouvelle sous-ligne est aussi composée de 12 colonnes

Bootstrap

La grille de Bootstrap (≠ `Grid`)

- Bootstrap utilise une grille composée de 12 colonnes pour chaque ligne (`row`)
- En cas d'imbrication, la nouvelle sous-ligne est aussi composée de 12 colonnes

Exemple (remplacer bonjour par le paragraphe `lorem ipsum`)

```
<div class=container> <!-- essayer container-fluid aussi -->
        <div class=row>
                <div class=col-md-6>
                        Bonjour
                </div>
                <div class=col-md-2>
                        Bonjour
                </div>
                <div class=col-md-4>
                        Bonjour
                </div>
        </div>
</div>
```

Bootstrap

Des colonnes définies par rapport au type de média

- `md` : écrans
- `lg` : grands écrans
- `xs` : smart phones
- `sm` : tablettes

Bootstrap

Des colonnes définies par rapport au type de média

- `md` : écrans
- `lg` : grands écrans
- `xs` : smart phones
- `sm` : tablettes

Remarque

Il est possible de combiner ces propriétés pour adapter l'affichage aux différents médias.

Liens vers les classes Bootstrap

`https://www.w3schools.com/bootstrap/default.asp`

Liens vers les classes Bootstrap

https://www.w3schools.com/bootstrap/default.asp

Exemple de boutons bootstrap

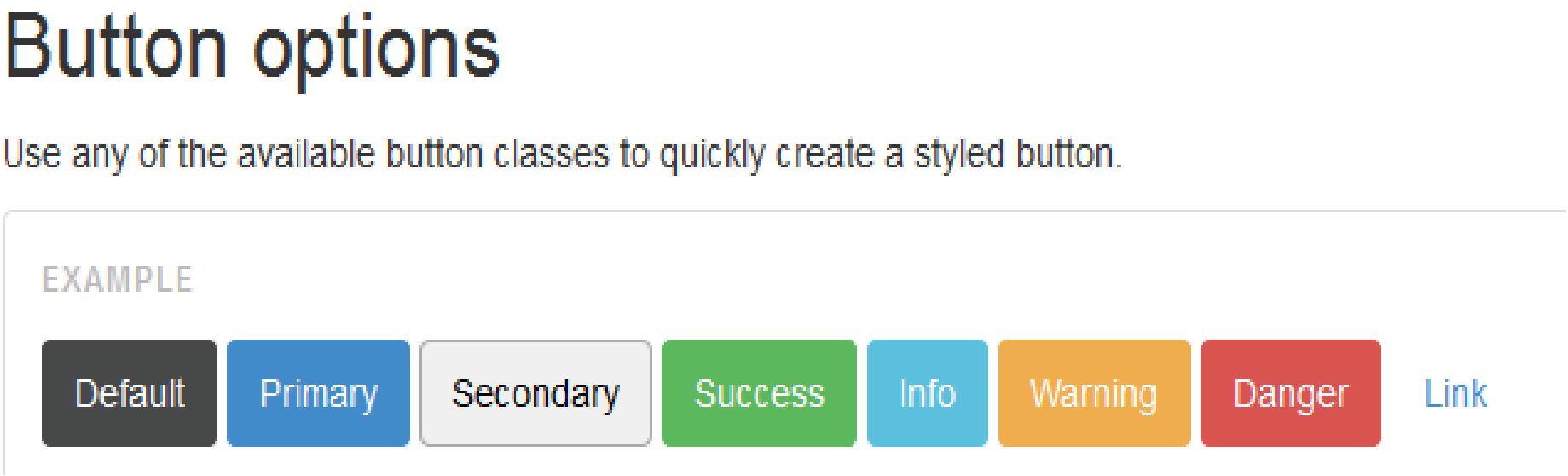
```
<button class = 'btn btn-success'> ajouter </button>
```

Liens vers les classes Bootstrap

https://www.w3schools.com/bootstrap/default.asp

Exemple de boutons bootstrap

```
<button class = 'btn btn-success'> ajouter </button>
```



Autres frameworks CSS

Autres frameworks CSS

- Semantic UI : `https://semantic-ui.com/`
- Foundation : `https://foundation.zurb.com/`
- Bulma : `https://bulma.io/`
- UIKit : `https://getuikit.com/`
- Material UI : `http://www.material-ui.com/#/`
- KNACSS : `https://knacss.com/`
- ...

Propriétés CSS3 et compatibilité des navigateurs

Pour tester la compatibilité d'une propriété CSS3 avec les navigateurs

`https://caniuse.com/`