



CSS : Cascading Style Sheets

Pr. M. OUTANOUTE

Année Universitaire : 2023/2024

1

FEUILLES DE STYLE EN CASCADE

- **N'oubliez pas : HTML pour le fond, CSS pour la forme**
 - La structure d'un document et son contenu sont décrits en HTML
 - Sa présentation est gérée par les CSS
- CSS :
 - **Cascading Style Sheets** ⇔ Feuilles de style en cascade
- **Description :**
 - Un document en mode texte
 - Utilisable par des documents HTML
 - Présentation identique de tous les documents HTML
 - Apporte la modularité du formatage
 - Séparation du contenu et de la présentation
- **Des exemples Avec CSS**
 - « *le site Zen garden : <http://www.csszengarden.com/>* » :
Le même site avec des CSS différents

FEUILLES DE STYLE EN CASCADE

Avantage

- Document HTML et feuille CSS peuvent être définis dans des fichiers séparés;
- Création plus efficace;
- Code HTML plus simple et plus lisible;
- On peut changer la feuille de style sans modifier le document;
- On peut avoir plusieurs feuilles de style pour un document;
- Plusieurs pages peuvent partager la même feuille de style.

3

FEUILLES DE STYLE EN CASCADE

Principe

- Le langage CSS définit un ensemble de **propriétés** qui ont une influence sur l'affichage des éléments d'une page;
- Pour chaque propriété il existe un ensemble de **valeurs** possibles;
- Il est possible de fixer ces propriétés pour chacun des éléments d'un document HTML;
- Les propriétés agissent sur l'apparence de la **boîte d'un élément**.
Ces propriétés concernent :
 - L'apparence du contenu (fonte, style, couleur, ...)
 - La taille de la boîte (largeur, marges, ...)
 - Le positionnement de la boîte (absolu ou relatif, visibilité)
 - ...

4

FEUILLES DE STYLE EN CASCADE

Inclusion d'une CSS

- Trois possibilités d'inclusion :

1) Directement dans les balises en utilisant l'**attribut style** (non recommandée)

```
<h2 style="color : red">Titre en rouge</h2>
```

2) Définition de la CSS dans le **head** du fichier html via la **balise <style>**

```
<head>
  <style>
    déclaration des styles
  </style>
</head>
```

3) Déclaration d'un **lien externe** vers la CSS via la **balise <link>**

```
<head>
  <link href="fichier.css" rel="stylesheet" >
</head>
```

5

FEUILLES DE STYLE EN CASCADE

1) Inclusion d'une CSS : Directement dans les balises (non recommandé)

- Vous pouvez ajouter un attribut **style** à n'importe quelle balise.

Vous insérerez votre code CSS directement dans cet attribut.

```
<html>
  <head>
    <meta charset="utf-8" />
    <title> Premiers tests du CSS</title>
  </head>
  <body>
    <h1> Mon super site </h1>
    <p style="color: blue;"> Bonjour et bienvenue sur mon site ! </p>
    <p> Pour le moment, mon site est un peu <em> vide </em>.
    Patinez encore un peu ! </p>
  </body>
</html>
```

6

FEUILLES DE STYLE EN CASCADE

2) Inclusion d'une CSS : Dans l'en-tête <head> du fichier HTML

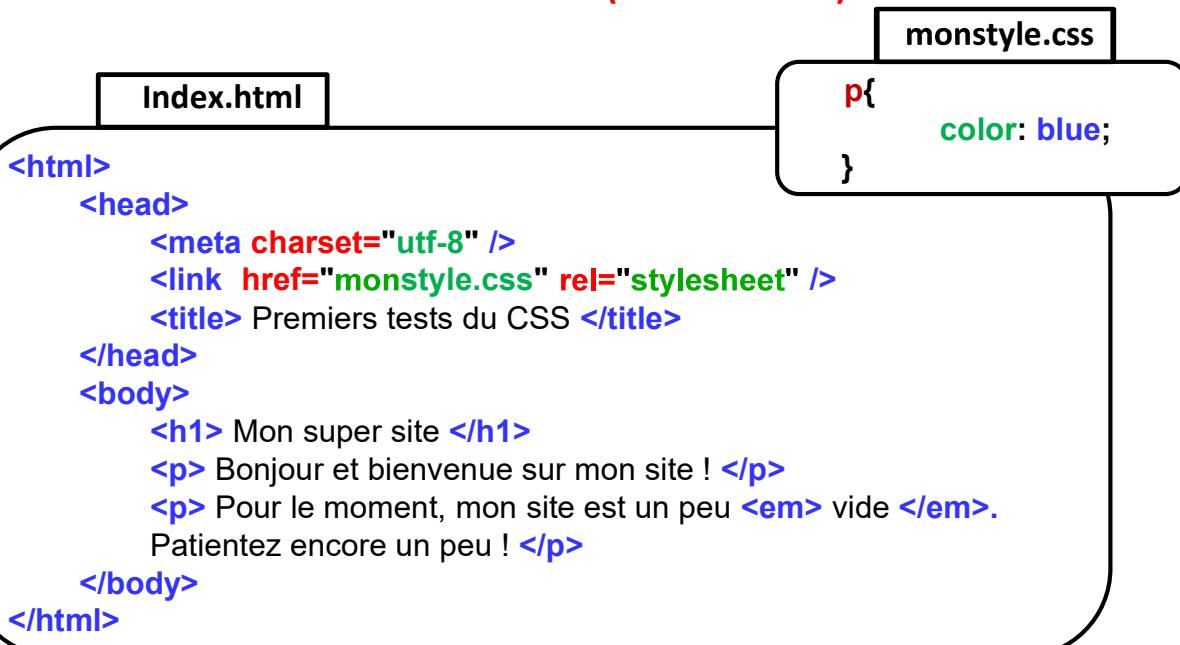
- Cela consiste à insérer le code CSS directement dans une balise **<style>** à l'intérieur de l'en-tête **<head>**.

```
<html>
  <head>
    <meta charset="utf-8" />
    <style>
      p {
        color: blue;
      }
    </style>
    <title> Premiers tests du CSS</title>
  </head>
  <body>
    <h1> Mon super site </h1>
    <p> Bonjour et bienvenue sur mon site ! </p>
    <p> Pour le moment, mon site est un peu <em> vide </em>.
      Patientez encore un peu ! </p>
  </body>
</html>
```

7

FEUILLES DE STYLE EN CASCADE

3) Inclusion d'une CSS : Dans un fichier .css (recommandé)



- Le contenu de la ligne 4, **<link href="monstyle.css" rel="stylesheet" />** : indique que ce fichier HTML est associé à un fichier appelé **monstyle.css** et chargé de la mise en forme (**rel="stylesheet"**).

8

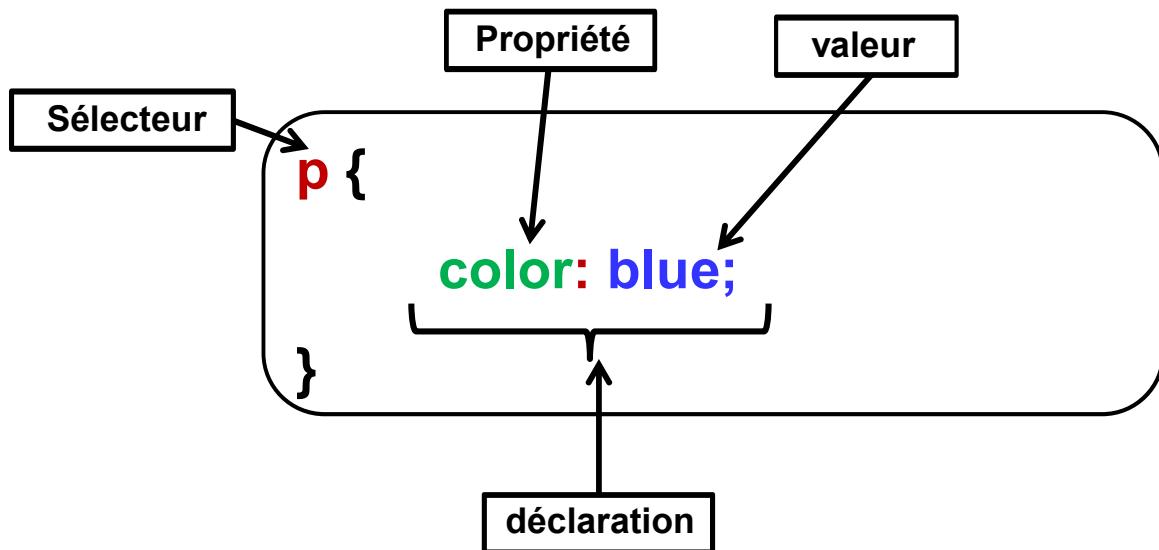
FEUILLES DE STYLE EN CASCADE

Déclaration d'une règle :

Une règle CSS définit la **valeur** d'une **propriété** CSS pour un **sélecteur** donné.

sélecteur { propriété : valeur; }

Le sélecteur détermine les éléments sur lesquels s'applique la règle.



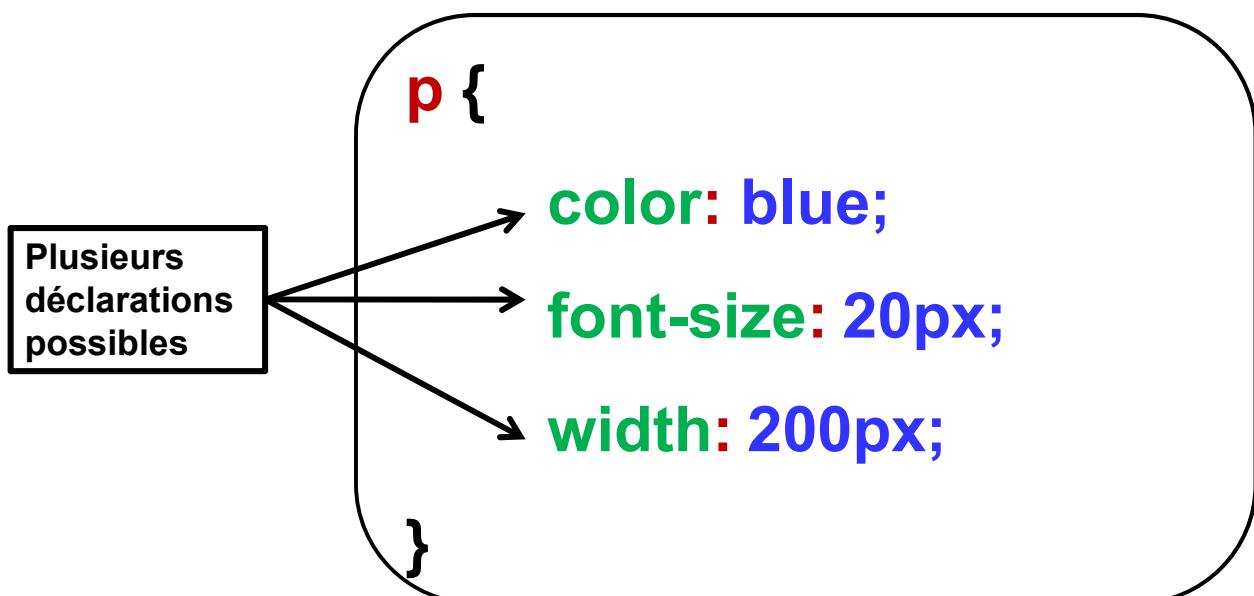
9

FEUILLES DE STYLE EN CASCADE

Déclaration d'une règle :

Il est possible de regrouper plusieurs règles d'un même sélecteur.

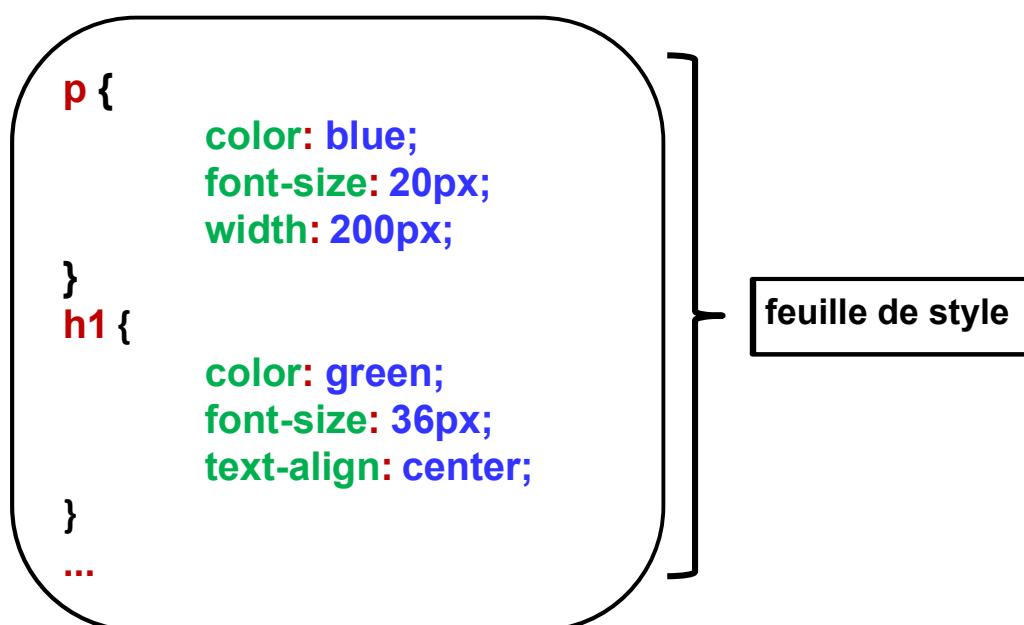
Les définitions sont alors séparées par des points-virgules.



10

FEUILLES DE STYLE EN CASCADE

Déclaration d'une règle :

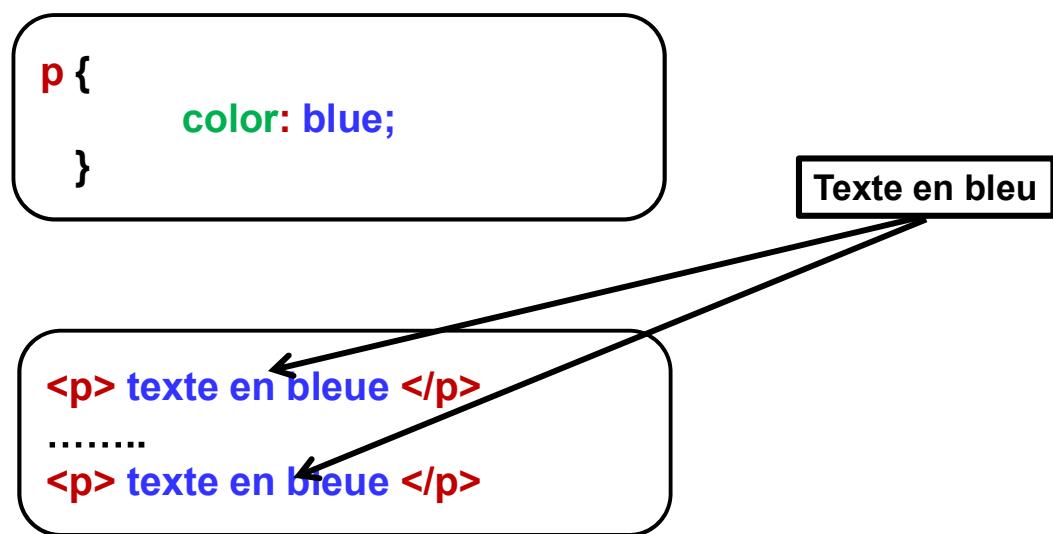


11

TYPE DE SÉLECTEUR

Sélecteur de type : nom de balise

Les sélecteurs de type CSS ciblent des éléments en fonction du nom de leur nœud. Ainsi, lorsqu'un sélecteur de type est utilisé seul, il ciblera tous les éléments de ce type (autrement dit tous les nœuds avec ce nom) contenus dans le document.

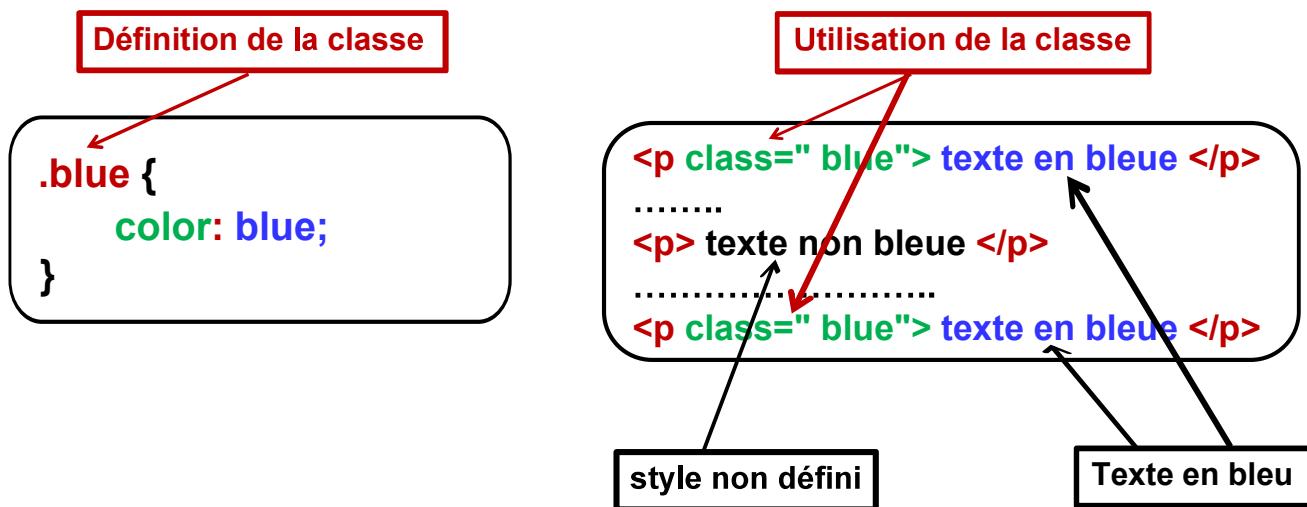


12

TYPE DE SÉLECTEUR

Sélecteur de Classe :

- Les sélecteurs de classe CSS permettent de cibler des éléments d'un document en fonction du contenu de l'attribut **class** de chaque élément.
- Plusieurs objets peuvent être de la même classe.
- Style : utilisation du symbole ":" (point).



13

TYPE DE SÉLECTEUR

Sélecteur de Classe :

- Cible tous les éléments ayant la classe "spacious"

```
.spacious
{
    margin: 2em;
}
```

- Cible tous les éléments **li** ayant la **classe "spacious"**

```
li.spacious
{
    margin: 2em;
}
```

14

TYPE DE SÉLECTEUR

Sélecteur de Classe :

Chaque **p** qui à la classe **.big** (pas d'espace entre l'élément et la classe)

```
p.big {  
    font-size: 20px;  
}
```

```
<p class="big">...</p>  
<div class="big">...</div>
```

Taille de
texte 20px

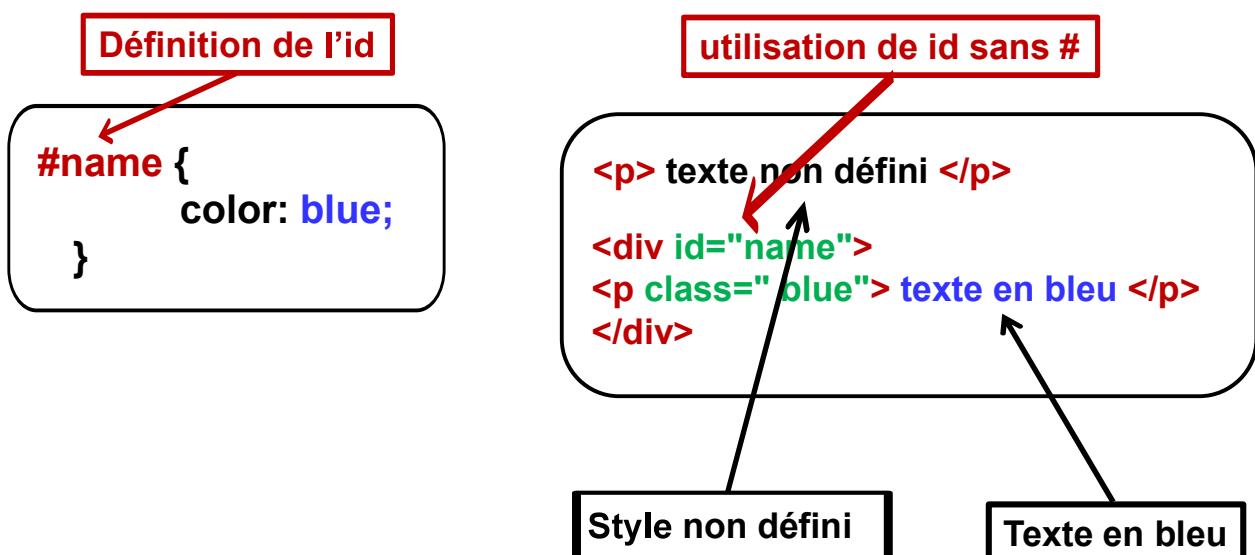
style non défini

15

TYPE DE SÉLECTEUR

Sélecteur d'ID :

- Un sélecteur d'identifiant (*ID selector*) permet, pour un document HTML, de cibler un élément grâce à la valeur de son attribut **id**.
Il faut que la valeur soit exactement la même que celle du sélecteur pour que l'élément soit effectivement ciblé.
- Style : utilisation du symbole **"#"**



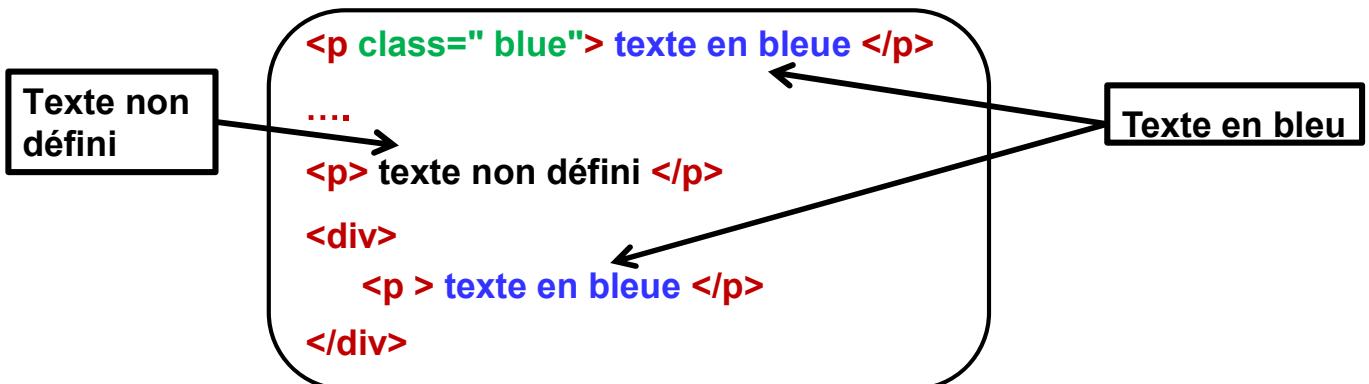
16

TYPE DE SÉLECTEUR

Groupement de Sélecteurs :

Sélecteurs séparés par des virgules

```
div, .blue{  
    color: blue;  
}
```



17

TYPE DE SÉLECTEUR

Les sélecteurs avancés

- * : sélecteur universel :

```
*
```

– Sélectionne toutes les balises sans exception. On l'appelle le sélecteur universel.

- A B : une balise contenue dans une autre

```
h3 em
```

– Sélectionne toutes les balises `` situées à l'intérieur d'une balise `<h3>`.

Notez qu'il n'y a pas de virgule entre les deux noms de balises

- A + B : une balise qui en suit une autre

```
H3+p
```

```
<h3>Titre</h3>  
<p>Paragraphe</p>
```

– Sélectionne la première balise `<p>` située après un titre `<h3>`.

18

TYPE DE SÉLECTEUR

Sélecteur d'attribut :

- *A[attribut] : une balise qui possède un attribut*

```
a[title]  
{  
}
```

```
<a href="http://site.com" title="Infobulle">
```

– Sélectionne tous les liens `<a>` qui possèdent un attribut **title**.

- *A[attribut="Valeur"] : une balise, un attribut et une valeur exacte*

```
a[title="Cliquez ici"]  
{  
}
```

```
<a href="http://site.com" title=" Cliquez ici ">
```

– l'attribut doit en plus avoir exactement pour valeur "Cliquez ici".

- *A[attribut*="Valeur"] : une balise, un attribut et une valeur*

```
a[title*="ici"]  
{  
}
```

```
<a href="http://site.com" title=" Cliquez ici ">
```

– l'attribut doit cette fois contenir dans sa valeur le mot "ici"
(peu importe sa position).

19

TYPE DE SÉLECTEUR

Sélecteur d'attribut : Exemples

<code>img[width]</code>	les images ayant un attribut width	<code></code>
<code>input[type=text]</code>	Les inputs de type text	<code><input type='text'></code>
<code>img[title^=red]</code>	Les images avec un titre commençant par red	<code>oui non</code>
<code>img[title*=red]</code>	Les images avec un titre contenant le mot red	<code>oui</code>
<code>[lang =en]</code>	Les éléments avec un attribut lang commençant par en-	<code><div lang="en-us">oui <div lang="fr">non</code>
<code>img[src\$=.jpg]</code>	Les images ayant un nom se terminant par .jpg	<code>oui non</code>
<code>td[colspan='3']</code>	Les cases d'un tableau qui sont sur 3 colonnes	<code><td colspan=3>oui</td> <td>non</td></code>

20

COMBINAISON DE SÉLECTEURS

- Sélecteur de descendant

`p h2 { color: green }`

Les h2 qui sont dans un p

- Sélecteur d'enfant

`p > h2 { font-size: 30px }`

Les h2 qui sont directement fils d'un p

- Sélecteur d'adjacent :

`p + h2 { font-size: 10px }`

Les h2 qui sont directement après un p

21

COMBINAISON DE SÉLECTEURS

Sélecteur de descendant :

Article p {
color: blue;
}

Texte bleu

Style
non
définit

<article>...
<p> un texte ... </p>
</article>
un texte ...
<p> un texte ... </p>
<article>...
<div><p> un texte ... </p></div>
</article>

Texte bleu

22

COMBINAISON DE SÉLECTEURS

Sélecteur d'enfant :

```
article > p {  
    color: blue;  
}
```

Texte bleu

Texte non définie

```
<article>...  
    <p> un texte ... </p>  
</article>  
    un texte ...  
    <p> un texte ... </p>  
<article>...  
    <div><p> un texte ... </p></div>  
</article>
```

Texte non définie

23

COMBINAISON DE SÉLECTEURS

Exemples :

Sélecteur	Signification	Exemple
*	Tous les éléments	
h1, p	Les h1 et p	<h1>...<p>...<p>...<h1>
div p	Les p situés dans div	<div>oui<p>...non<p>
p#gras	Les p ayant un id gras	<p id='gras'>oui<p>non
#gras	Tout élément ayant un 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>...</h1><p>oui</p><p>non
div > p	Les p enfants directs de div	<div>...<p>oui</p> <div>......<p>non
div ~ p	Les p précédés par div avec p et div ont le même parent	...<p>non</p> <div>...</div><p>oui</p>

24

PSEUDO-CLASSES

Les pseudo-classes :

- Une pseudo-classe est un sélecteur ciblant des éléments dans un état spécifique, par exemple le premier élément d'un type, ou un élément survolé par le pointeur de la souris. Leur comportement correspond à celui d'une classe.
- Les pseudo classes sont reconnaissables au fait qu'elles commencent tous par le symbole : « `:nom_pseudo-classe` »
 - **:first-child** : Sélectionne la première balise `<p>` du corps de la page (`body`) ou les premières balises `p` à l'intérieur d'une autre balise parent.
 - **:last-child** : Sélectionne la dernière balise `<p>` du corps de la page (`body`) ou les dernières balises `p` à l'intérieur d'une autre balise parent.
 - **:nth-child(n)** : Sélectionne toutes les balises `<p>` étant positionnées en 2^{ème} élément dans son parent ou dans le corps de page général (`body`).

`p:first-child{}`

```
<p> Mon premier paragraphe </p>
<p> Mon second paragraphe </p>
```

`p:last-child{}`

```
<p> Mon premier paragraphe </p>
<p> Mon second paragraphe </p>
```

`p:nth-child(2){}`

```
<p> Mon premier paragraphe </p>
<p> Mon second paragraphe </p>
<p> Mon 3ème paragraphe </p>
```

25

PSEUDO-CLASSES

Les pseudo-classes d'action utilisateur :

- Certaines pseudo-classes ne s'appliquent que lorsque l'utilisateur interagit avec le document d'une manière ou d'une autre.
Ces pseudo-classes d'action utilisateur, parfois appelées pseudo-classes dynamiques, agissent comme si une classe avait été ajoutée à l'élément lorsque l'utilisateur interagit avec lui.
- Par exemple, nous allons pouvoir afficher un paragraphe en gras lorsque l'utilisateur passe sa souris dessus ou changer la couleur d'un lien une fois celui-ci cliqué.
- **Exemples :**
 - **:link** va nous permettre de styliser un lien non visité;
 - **:visited** va nous permettre de styliser un lien une fois celui-ci visité.
 - **:hover** va nous permettre de changer l'aspect d'un élément lorsque les visiteurs poseront leur curseur dessus ;
 - **:active** va nous permettre de modifier l'aspect d'un lien lors du clic.

26

PSEUDO-CLASSES

```
<body>
  <h1>Pseudo classes CSS</h1>
  <div class="test">
    <p> un
      <a href="http://www.site.ma">
        lien1
      </a>
      non visité
    </p>
    <p> un paragraphe </p>
  </div>
  <div>
    <p> un
      <a href="http://www.google.ma">
        lien2
      </a>
      déjà visité
    </p>
  </div>
  <p>un paragraphe</p>
  <P><strong>paragraphe importatnt</strong></p>
</body>
```

```
h1:hover{
  color: orange;
  font-family : verdana, sans-serif;
}
a:link{
  color: red;
  text-decoration : none;
}
a:active
{
  background-color : #FFCC66;
}
a:visited{
  color : green;
}
```

27

PSEUDO-ELEMENTS

Les pseudo-éléments :

Les pseudo-éléments se comportent de manière similaire.

Cependant, ils agissent comme si vous aviez ajouté un tout nouvel élément HTML dans le balisage, plutôt que d'appliquer une classe aux éléments existants.

- **:first-letter** : Sélectionne la 1ère lettre du texte contenue dans tous les éléments `<p>`

```
p:first-letter{}
<p> Ce texte comporte une première lettre </p>
```

- **:first-line** : Sélectionne chaque 1ère ligne de chaque balise `<p>/`

```
p:first-line{}
<p> Ma première ligne <br> Ma seconde ligne</p>
```

28

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 = 'en'
p:first-of-type	les premiers enfants de type p de leur parent
p:first-child	les p qui sont les premiers enfants de leur parent
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-child(even)	les p d'indice pair
p:nth-child(odd)	les p d'indice impair
p:nth-child(3n+2)	les p d'indice 2, 5, 8...
p:nth-last-of-type(2)	les seconds enfants de type p de leur parent en commençant par le dernier

29

PSEUDO-SÉLECTEURS

p:only-child	les p qui sont uniques enfants de leur parents
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

30

RÉSOLUTION DES CONFLITS

- La cascade est une caractéristique fondamentale de CSS.
- C'est un algorithme qui définit comment combiner les valeurs de propriétés provenant de différentes sources.
- La cascade combine l'**importance**, l'**origine**, la **spécificité** et l'**ordre** source des déclarations de style applicables pour déterminer exactement quelle déclaration doit être appliquée à un élément donné.
- S'il y a un conflit, comment résoudre ce conflit?

En d'autres termes, comment dire quelle règle CSS gagne?

31

RÉSOLUTION DES CONFLITS

Calcul de priorité

- On compte pour chaque sélecteur :
 - **a** : le style lié à la balise
 - **b** : nombre de **sélecteurs d'id** (= nombre de #)
 - **c** : nombre de sélecteurs **de classes, pseudo-classes** ou d'**attributs**
 - **d** : nombre d'**éléments** ou de **pseudo-éléments**
- Le sélecteur reçoit la priorité **a b c d**.
- Le sélecteur avec la plus grande priorité l'emporte.
- En cas d'égalité, la dernière déclaration l'emporte.

32

RÉSOLUTION DES CONFLITS

Calcul de priorité : Exemple

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

a

Style = "..."

b

ID

c

Class, pseudo-class
ou attribut

d

Eléments ou
pseudo-éléments

0

0

0

2

33

RÉSOLUTION DES CONFLITS

Calcul de priorité : Exemples

sélecteur	style	ID	Classe, Pseudo-Classe, Attribut	Élément, pseudo-élément
* {...}	0	0	0	0
h1 {...}	0	0	0	1
div.reponse{...}	0	0	1	1
#joconde{...}	0	1	0	0
div a{...}	0	0	0	2
div a:visited{...}	0	0	1	2
p span.fichier{...}	0	0	1	2
p a[href\$=".pdf"] {...}	0	0	1	2
p.enonce a[href\$=".pdf"] {...}	0	0	2	2
ol.exercice li.question{...}	0	0	2	2
div#diaporama img.gauche{...}	0	1	1	2
article#special p:hover{...} <p style="..."> texte </p>	1	1	1	2

34

RÉSOLUTION DES CONFLITS

```
<p> zero </p> 1  
<h1> Titre </h1>  
<p> premier </p> 1 2  
<p> second </p> 1 7  
<p> troisième </p> 1 7  
<p lang="en"> quatre </p> 1 5 7  
<p class="bleu"> cinquième </p> 1 6 7  
<p class="bleu"> sixième </p> 1 6 7  
<p class="bleu" id="special"> 7ème </p> 1 4 6 7  
<p huitième </p> 1 7  
<div>  
<p neuvième </p> 1 3  
<p lang="en"> dix </p> 1 3 5 7  
</div>  
<p onzième </p> 1  
<h1> Second titre </h1>  
<p douzième </p> 1 2
```

1	p { b-g :pink }
2	h1+p { b-g:red; }
3	div>p { b-g:yellow; }
4	p#special { b-g:gold; }
5	p[lang=en] { b-g:green; }
6	p.bleu { b-g:lightblue; }
7	p+p { b-g:lightgreen; }

	a	b	c	d
1	0	0	0	1
2	0	0	0	2
3	0	0	0	2
4	0	1	0	1
5	0	0	1	1
6	0	0	1	1
7	0	0	0	2

4 > 6 > 5 > 7 > 3 > 2 > 1

35

FORMATAGE DU TEXTE

Le formatage de texte signifie simplement que l'on va modifier l'apparence du texte.

On dit qu'on le *met en forme*.

La taille

■ Pour modifier la taille du texte, on utilise la propriété CSS **font-size**

■ plusieurs techniques sont proposées :

— **Taille absolue** : en **pixels**, en **centimètres** ou **millimètres**.

Cette méthode est très précise mais il est conseillé de ne l'utiliser que si c'est absolument nécessaire, car on risque d'indiquer une taille trop petite pour certains lecteurs.

— **Taille relative** : en pourcentage, « **em** » ou « **ex** ».

Cette technique a l'avantage d'être plus souple. Elle s'adapte plus facilement aux préférences de taille des visiteurs.

36

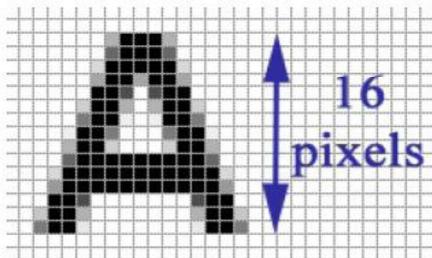
FORMATAGE DU TEXTE

Taille absolue

- Pour indiquer une taille absolue, on utilise généralement les pixels.

Pour avoir un texte de 16 pixels de hauteur, vous devez donc écrire :

font-size: 16px;



- Exemple

```
p{  
    font-size: 14px;  
}  
h1{  
    font-size: 40px;  
}
```

37

FORMATAGE DU TEXTE

Taille relative

- C'est la méthode recommandée, car le texte s'adapte alors plus facilement aux préférences de tous les visiteurs.

Il y a plusieurs moyens d'indiquer une valeur relative.

- Vous pouvez par exemple écrire la taille avec des mots en anglais comme ceux-ci :

- xx-small : minuscule ;
- x-small : très petit ;
- small : petit ;
- medium : moyen ;
- large : grand ;
- x-large : très grand ;
- xx-large : gigantesque.

```
p{  
    font-size: small;  
}  
h1{  
    font-size: large;  
}
```

38

FORMATAGE DU TEXTE

Taille relative

- La technique précédente a un défaut : il n'y a que sept tailles disponibles.
Il existe d'autres moyens.
Cette technique consiste à indiquer la taille en « **em** ».
 - Si vous écrivez **1em**, le texte a une taille normale.
 - Si vous voulez grossir le texte, vous pouvez inscrire une valeur supérieure à 1, comme **1.3em** par exemple (pour multiplier la taille par 1.3).
 - Si vous voulez réduire le texte, inscrivez une valeur inférieure à 1, comme **0.8em**.
- D'autres unités sont disponibles :
 - « **ex** » : qui fonctionne sur le même principe que le **em**, mais qui est plus petit de base ;
 - Le **pourcentage (80%, 130%...)**.

39

FORMATAGE DU TEXTE

La police

- La propriété CSS qui permet d'indiquer la police à utiliser est **font-family**.
- La liste de polices qui fonctionnent bien sur la plupart des navigateurs :
 - Arial ;
 - Arial Black ;
 - Comic Sans MS ;
 - Courier New ;
 - Georgia ;
 - Impact ;
 - Times New Roman ;
 - Trebuchet MS ;
 - Verdana.

```
p{  
    font-family: Arial;  
}
```

40

FORMATAGE DU TEXTE

Italique, gras ...

- Pour mettre en *italique*, on utilise **font-style** qui peut prendre trois valeurs :
 - **italic** : le texte sera mis en italique.
 - **oblique** : le texte sera passé en oblique (les lettres sont penchées, le résultat est légèrement différent de l'italique proprement dit).
 - **normal** : le texte sera normal (par défaut). Cela vous permet d'annuler une mise en italique.
- Pour mettre en **gras**, on utilise **font-weight** qui prend les valeurs suivantes :
 - **bold** : le texte sera en gras ;
 - **normal** : le texte sera écrit normalement (valeur par défaut).
- Pour changer la taille de la police, on utilise **font-size**.
 - **font-size: 40px;**
 - **font-size: 2.5em;**

```
p{  
    font-style: italic;  
    font-weight: bold;  
}
```

41

FORMATAGE DU TEXTE

Les propriétés CSS de type « text-...

- Parmi ces propriétés :
 - La propriété **text-align** (gère l'alignement) ;
 - La propriété **text-transform** (gère la mise en majuscules / minuscules) ;
 - La propriété **text-decoration** (gère la décoration) ;
 - La propriété **text-indent** (gère l'indentation) ;
 - La propriété **text-shadow** (gère les ombres).
- La propriété **text-decoration** permet, entre autres, de souligner le texte.
Voici les différentes valeurs qu'elle peut prendre :
 - **underline** : souligné.
 - **line-through** : barré.
 - **overline** : ligne au-dessus.
 - **blink** : clignotant. Ne fonctionne pas sur tous les navigateurs.
 - **none** : normal (valeur par défaut).

42

FORMATAGE DU TEXTE

Alignment de texte :

- Le langage CSS nous permet de faire tous les alignements connus : à gauche, centré, à droite et justifié.
- On utilise la propriété **text-align** et on indique l'alignement désiré :
 - **left** : le texte sera aligné à gauche (c'est le réglage par défaut).
 - **center** : le texte sera centré.
 - **right** : le texte sera aligné à droite.
 - **justify** : le texte sera « justifié ». Justifier le texte permet de faire en sorte qu'il prenne toute la largeur possible sans laisser d'espace blanc à la fin des lignes.

```
p{  
    text-decoration: underline;  
    text-align: center;  
}
```

43

FORMATAGE DU TEXTE

La propriété CSS **text-transform**

- Cette propriété permet de transformer un texte ou une partie d'un texte en majuscules ou en minuscules.

Cette propriété nous permet de choisir parmi cinq valeurs :

- **Lowercase** : Met tout le texte en minuscules ;
- **Uppercase** : Met tout le texte en majuscules ;
- **Capitalize** : Met la première lettre de chaque mot en majuscule ;
- **Inherit** : Hérite de la valeur de l'élément parent ;
- **None** : Pas de transformation du texte.

Utile pour annuler une transformation par défaut donnée par héritage.

```
p{  
    text-transform: capitalize;  
}
```

44

FORMATAGE DU TEXTE

La propriété CSS **text-shadow**

- La propriété CSS **text-shadow** va nous permettre de créer des ombres autour de nos textes, afin que ceux-ci se détachent de l'arrière plan.

On va devoir indiquer quatre valeurs dans un ordre précis à cette propriété afin que celle-ci fonctionne correctement :

- **La projection horizontale de l'ombre** (en px) ;
- **La projection verticale de l'ombre** (en px) ;
- **Le rayon de propagation de l'ombre** (le « radius », en px) ;
- **La couleur de l'ombre.** Accepte les mêmes valeurs que la propriété color.

```
h1{  
    text-shadow : -1px 1px 1px blue;  
}
```

45

FORMATAGE DU TEXTE

Les couleurs

- **Le nom de la couleur :**

La figure suivante vous montre les seize couleurs que vous pouvez utiliser en tapant simplement leur nom.

- **La notation hexadécimale.** : cela ressemble à #FF23B8

On doit toujours commencer par écrire un dièse (#), suivi de six lettres ou chiffres allant de **0** à **9** et de **A** à **F**.

Les deux premiers indiquent une **quantité du rouge**, les deux suivants une **quantité du vert** et les deux derniers une **quantité du bleu**.

- **La méthode RGB** : cela ressemble à **RGB(240,96,204)**

S'écrit **Red-Green-Blue** et signifie en anglais : Rouge-Vert-Bleu .

Les valeurs varient entre 0 et 255.

white	
silver	
gray	
black	
red	
maroon	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	
aqua	
teal	

46

FORMATAGE DU TEXTE

L'opacité des textes

- Il y a deux manières de gérer l'opacité des textes (le fait d'être plus ou moins transparent) :

- En utilisant la propriété CSS **opacity**. Cette propriété va prendre une valeur entre **0** (texte totalement transparent) et **1** (texte totalement opaque).

opacity: 0.5;

- Indiquer une quatrième valeur définissant l'opacité dans nos notations **RGB**. Dans ce cas là, la notation devient **RGBa**. Cette quatrième valeur, doit être comprise entre **0** (texte transparent) et **1** (texte opaque).

color: RGBa(255,000,000,0.9);

47

EXEMPLE

```
<body>
  <h1> Liste des cours </h1>
  <p id="P1">
    Cours Base de données
  </p>
  <p id="P2">
    Cours Technologie Web
  </p>
  <p id="P3">
    Cours langage de
    Programmation C
  </p>
  <p id="P4">
    Cours Architecture des ordinateurs
  </p>
  <p id="P5">
    Cours Programmation JAVA
  </p>
</body>
```

```
h1{
  text-shadow : 3px 5px 1px blue;
}
#P1{
  font-family: Courier New;
}
#P2{
  font-style: italic;
  font-weight: bold;
}
#P3{
  text-decoration: underline;
  text-align: center;
}
#P4{
  text-transform : capitalize;
}
#P5{
  color: RGBa(255,000,000,0.4);
}
```

48

L'ARRIÈRE PLAN

Background

- La propriété CSS **background** va nous permettre de modifier le fond.
- On peut définir directement en CSS le fond d'une page HTML et même une image.
- Grâce à la propriété **background-color**, nous allons tout aussi bien pouvoir rajouter une couleur derrière un texte ou une couleur de fond pour un élément ou une page en l'appliquant à l'élément body.
- Pour ajouter une image de fond derrière un élément nous allons cette fois-ci utiliser la propriété **background-image**.

49

L'ARRIÈRE PLAN

Background-color

```
body{  
    background-color: purple;  
}  
  
h{  
    background-color: #069;  
}  
  
div{  
    background-color: RGBa(200, 200, 000, 0.5);  
}
```



50

L'ARRIÈRE PLAN

Background-image

```
body{  
    background-image: url(Koala.jpg);  
    background-size: 25%;  
}
```



51

LE MODÈLE DES BOÎTES

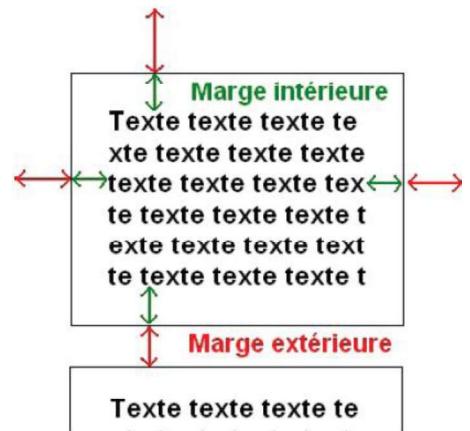
- En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :
 - **Les balises inline** : se trouvent obligatoirement à l'intérieur d'une balise block.
Une balise **inline** ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne.
 - **Les balises block** : créent automatiquement un retour à la ligne avant et après.
Il est possible de mettre un bloc à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design de notre site !
- Il existe deux balises génériques et, comme par hasard, la seule différence entre les deux est que l'une d'elle est **inline** et l'autre est **block** :
 - **** (**inline**) ;
 - **<div></div>** (**block**).
- Contrairement à un **inline**, un bloc a des dimensions précises.
Un bloc possède une **largeur** et une **hauteur**. On dispose de deux propriétés CSS :
 - **width** : c'est la largeur du bloc. À exprimer en pixels (**px**) ou en pourcentage (**%**).
 - **height** : c'est la hauteur du bloc. On l'exprime soit en pixels (**px**), soit en pourcentage(**%**).

52

LE MODÈLE DES BOÎTES

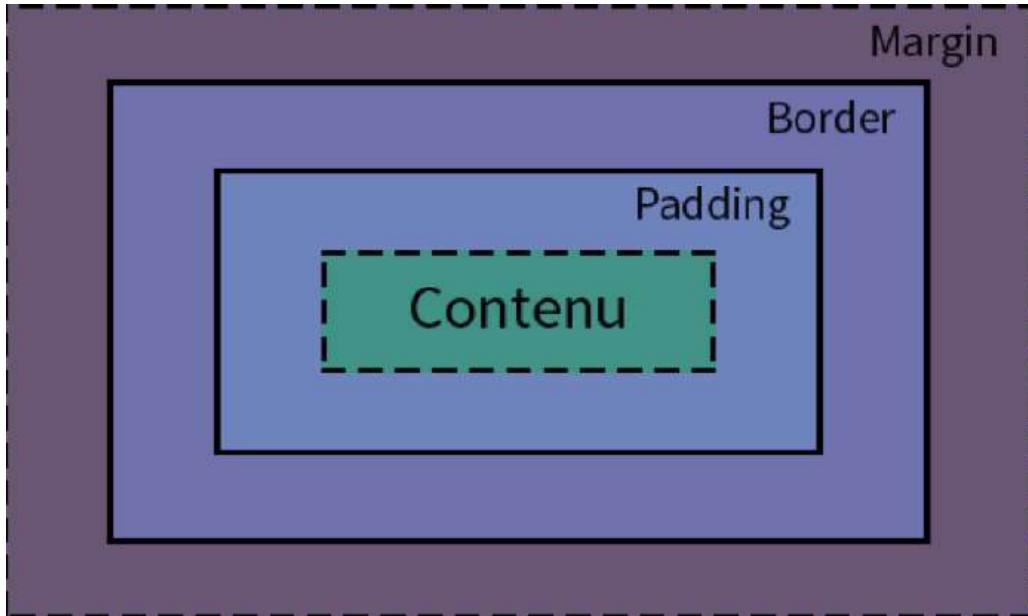
- Pour que notre site s'adapte aux différentes résolutions d'écran de nos visiteurs, un **bloc** a des dimensions minimales et maximales :
 - **min-width** : largeur minimale ;
 - **min-height** : hauteur minimale ;
 - **max-width** : largeur maximale ;
 - **max-height** : hauteur maximale.
- Les blocs possèdent des marges. Il existe *deux types de marge* :
 - **padding** : les marges intérieures.
L'espace entre le texte et la bordure (**en vert**).
 - **margin** : les marges extérieures.
L'espace entre la bordure et le bloc suivant (**en rouge**).

```
div{  
    width: 350px;  
    border: 1px solid black;  
    padding: 12px;  
    margin: 50px;  
}
```



53

LE MODÈLE DES BOÎTES



- La première boîte (la boîte centrale) représente le contenu de l'élément ou l'élément en soi.
- Ensuite, autour du contenu, la deuxième boîte contient en plus la marge intérieure de l'élément qu'on appelle en CSS le "**padding**".
- La troisième boîte va contenir en plus la **bordure** de l'élément.
- Enfin, la dernière boîte va également contenir la **marge** extérieure de l'élément.

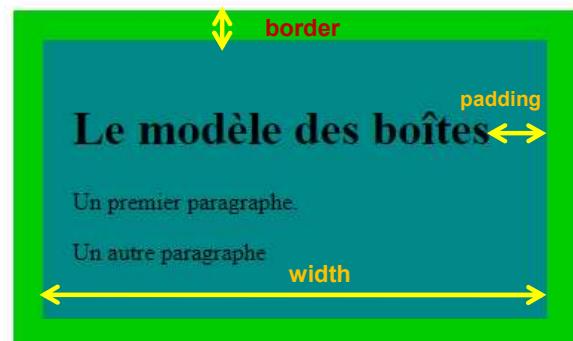
54

LE MODÈLE DES BOÎTES

```
<!DOCTYPE html>
<html>
  <head>
    <title>Le modèle des boîtes</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <div>
      <h1>Le modèle des boîtes</h1>
      <p>Un premier paragraphe.</p>
      <p>Un autre paragraphe</p>
    </div>
  </body>
</html>
```

```
div{
  /*Couleur de fond (bleu-vert)*/
  background-color: #088;
  /*Largeur de l'élément en soi*/
  width: 300px;
  /*Marge intérieure*/
  padding: 20px;
  /*Bordure (vert)*/
  border: 20px solid #0C0;
  /*Marge extérieure*/
  margin: 50px;
}
```



55

LE MODÈLE DES BOÎTES

- **margin** (comme **padding** d'ailleurs) s'applique aux quatre côtés du bloc;
- Si vous voulez spécifier des marges différentes en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises.
- Voici la liste pour **margin** :
 - **margin-top** : marge extérieure en haut ;
 - **margin-bottom** : marge extérieure en bas ;
 - **margin-left** : marge extérieure à gauche ;
 - **margin-right** : marge extérieure à droite.
- Et la liste pour **padding** :
 - **padding-top** : marge intérieure en haut ;
 - **padding-bottom** : marge intérieure en bas ;
 - **padding-left** : marge intérieure à gauche ;
 - **padding-right** : marge intérieure à droite.
- Il y a d'autres façons pour spécifier les marges avec les propriétés **margin** et **padding**.
Par exemple : **margin: 2px 0px 3px 1px;**
signifie : 2 px de marge en haut, 0 px à droite, 3 px en bas, 1 px à gauche ».

56

LE MODÈLE DES BOÎTES

- Pour régler efficacement la hauteur et la largeur d'un élément, il faut avant tout bien avoir compris le modèle des boîtes.
- En effet, vous devez bien vous rappeler que **vous ne définissez que la hauteur et la largeur du contenu de l'élément en soi.**
Les tailles des marges intérieures, extérieures et des bordures viendront s'ajouter à cette hauteur et à cette largeur afin de former la taille totale de l'élément.
- Si vous ne faites pas attention, vous risquez de créer des éléments plus grands au final que leurs éléments parents, ce qui est généralement considéré comme une mauvaise pratique en HTML et en CSS.

57

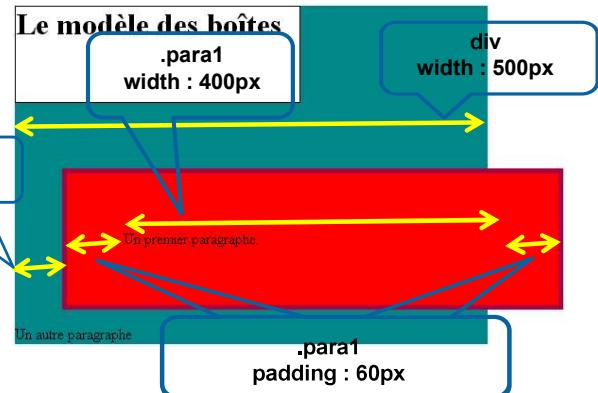
LE MODÈLE DES BOÎTES

```
<!DOCTYPE html>
<html>
  <head>
    <title>Le modèle des boîtes</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <div>
      <h1>Le modèle des boîtes</h1>
      <p class="para1">
        Un premier paragraphe.
      </p>
      <p class="para2">
        Un autre paragraphe
      </p>
    </div>
  </body>
</html>
```

```
div{
  background-color: #088;
  width: 500px;
}
h1{
  width:300px;
  background-color: white;
  height: 100px;
  border: 1px solid black;
}
.para1{
  width: 400px;
  background-color: red;
  padding: 60px;
  border: 5px solid #A04;
  margin-top:70px;
  margin-left:50px;
}
```

L'élément **p** fait donc au final:
 $400 + 2*60 + 2*5 + 50 = 580\text{px}$ de large,
soit **80px** de plus que le **div** (son élément parent)
ce qui fait qu'il déborde.



58

LE MODÈLE DES BOÎTES

Les bordures

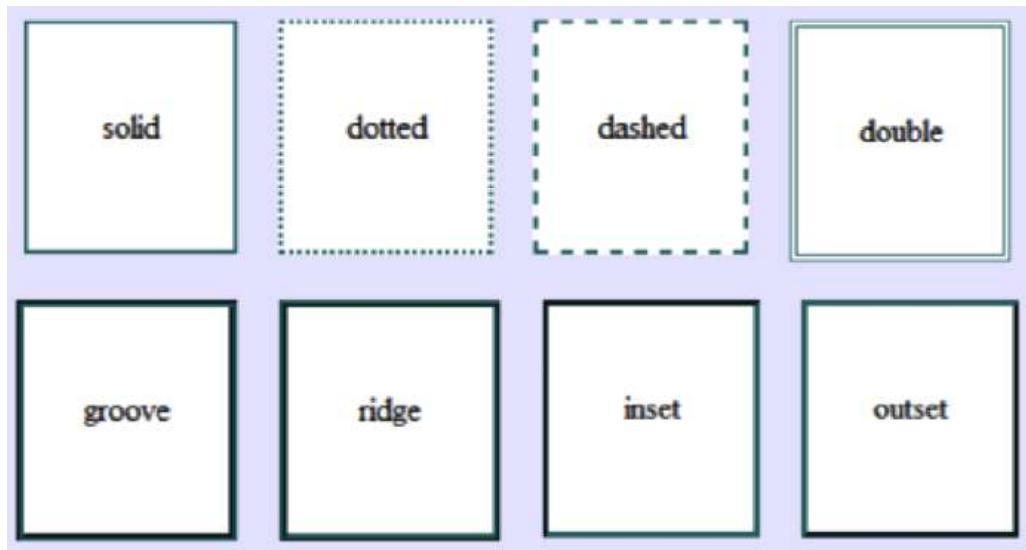
- L'espace pris par la bordure va se trouver entre la marge intérieure et la marge extérieure d'un élément HTML.
- Nous pouvons définir les bordures d'un élément de différentes manières en CSS : soit en utilisant les trois propriétés
 - **border-style**,
 - **border-width**
 - **border-color**,
- soit en utilisant directement la notation courte **border**.
- Le CSS nous permet également de définir des bordures différentes au dessus, à droite, en dessous et à gauche de nos éléments HTML. Pour cela, nous allons devoir ajouter les mots clés **top** (haut), **right** (droit), **bottom** (bas) et **left** (gauche) dans le nom de nos propriétés
- Nous pouvons créer des bordures arrondies avec la propriété **border-radius**.

59

LE MODÈLE DES BOÎTES

Les bordures

- La propriété **border-style** peut prendre huit valeurs différentes qui vont définir l'allure générale (le style) de notre bordure.



60

LE MODÈLE DES BOÎTES

- Il est tout à fait possible de centrer des blocs. C'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur.
- Pour centrer, il faut respecter les règles suivantes :
 - donnez une largeur au bloc (avec la propriété **width**) ;
 - indiquez que vous voulez des marges extérieures automatiques, comme ceci : **margin: auto;**.

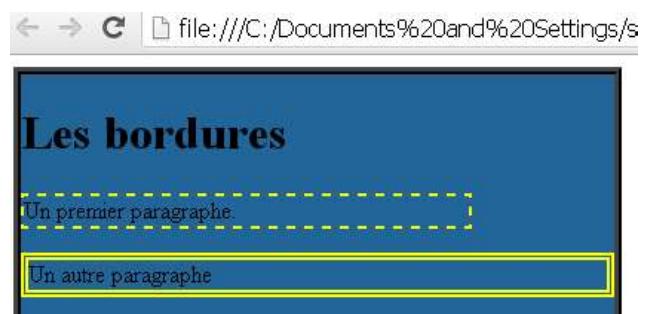
```
p{  
    width: 350px; /* On a indiqué une largeur (obligatoire) */  
    margin: auto; /* On peut donc demander à ce que le bloc soit centré avec auto */  
    border: 1px solid black;  
    text-align: justify;  
    padding: 12px;  
    margin-bottom: 20px;  
}
```

61

LE MODÈLE DES BOÎTES

Les bordures

```
div{  
    background-color: #269;  
    width: 400px;  
    border : 5px ridge #444;  
}  
.para1{  
    width:75%;  
    border-width : 2px;  
    border-style: dashed;  
    border-color: yellow;  
}  
.para2{  
    border: 5px double yellow;  
}
```



62

LE MODÈLE DES BOÎTES

Les bordures

```
div{  
background-color: #269;  
width: 400px;  
border-bottom : 5px ridge #444;  
border-top: 5px ridge #444;  
}  
.para1{  
width:75%;  
border-top-width : 2px;  
border-bottom-width: 10px;  
border-right-width: 5px;  
border-left-width: 5px;  
border-style: inset;;  
border-color: yellow;  
}  
.para2{  
border-top: 5px solid red;  
border-right: 5px solid lime;  
border-bottom: 5px solid red;  
border-left: 5px solid lime;  
}
```

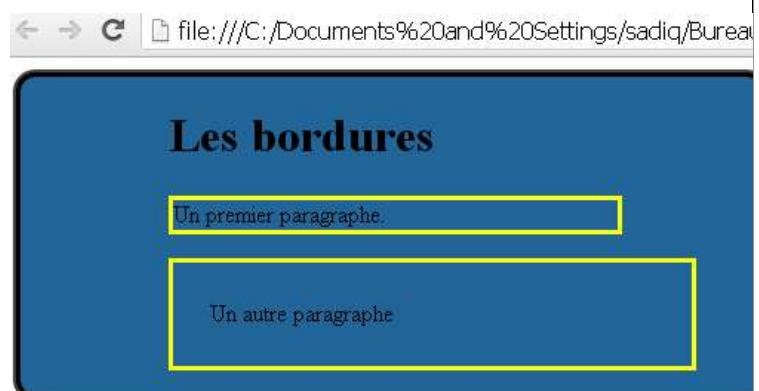


63

LE MODÈLE DES BOÎTES

Les bordures

```
div{  
background-color : #269;  
width : 400px;  
border : 5px ridge #444;  
border-radius : 15px;  
padding-left: 100px;  
}  
.para1{  
width:300px;  
border : 3px solid yellow;  
}  
.para2{  
width:300px;  
border : 3px solid yellow;  
padding: 25px;  
}
```



64

LE MODÈLE DES BOÎTES

La propriété Overflow :

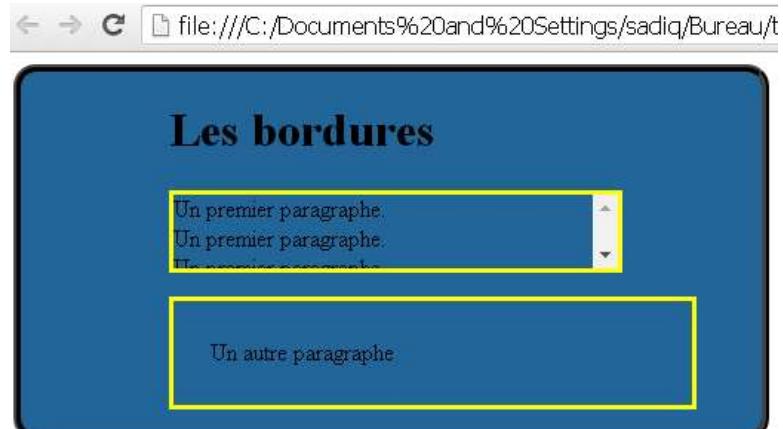
- Supposons que vous ayez un long paragraphe et que vous vouliez qu'il fasse **250 px de large et 110 px de haut**. Pour que le texte ne dépasse pas des limites du paragraphe, il va falloir utiliser la propriété **overflow**. Voici les valeurs qu'elle peut accepter :
 - **visible** (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc.
 - **hidden** : si le texte dépasse les limites, il sera tout simplement coupé. On ne pourra pas voir tout le texte.
 - **scroll** : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse lire l'ensemble du texte. C'est un peu comme un cadre à l'intérieur de la page.
 - **auto** : c'est le mode « pilote automatique ». En gros, c'est le navigateur qui décide de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire). C'est la valeur que je conseille d'utiliser le plus souvent.

65

LE MODÈLE DES BOÎTES

La propriété Overflow :

```
div{  
background-color : #269;  
width : 400px;  
border : 5px ridge #444;  
border-radius : 15px;  
padding-left: 100px;}  
.para1{  
width:300px;  
height : 50px;  
border : 3px solid yellow;  
overflow : auto;}  
}  
.para2{  
width:300px;  
border : 3px solid yellow;  
padding: 25px;}
```



66

LE MODÈLE DES BOÎTES

La propriété box-shadow :

- On peut créer des ombres autour de nos boîtes (à l'extérieur ou à l'intérieur des bordures) grâce à la propriété CSS **box-shadow**.
- Comme la propriété **text-shadow**, **box-shadow** prend quatre valeurs différentes dans l'ordre suivant :
 - Le déplacement horizontal (vers la droite ou la gauche) de l'ombre ;
 - Le déplacement vertical (vers le bas ou le haut) de l'ombre ;
 - Le rayon de propagation de l'ombre ;
 - La couleur de l'ombre.

67

LE MODÈLE DES BOÎTES

```
div{  
background-color: #89B;  
width: 400px;  
border: 5px solid #777;  
border-radius: 15px;  
box-shadow: 3px 3px 9px orange;  
}  
  
.para1{  
width: 50%;  
margin: 0px auto;  
padding: 20px;  
border: 2px solid #BB0;  
box-shadow: -4px -4px 5px yellow;  
}  
  
.para2{  
width: 50%;  
margin: 50px auto;  
padding: 20px;  
border: 2px solid #0B0;  
box-shadow: -4px 4px 5px lime inset;  
}
```



68

LE MODÈLE DES BOÎTES

Les propriétés float et clear :

- Les CSS sont conçues pour afficher les éléments au sein d'un flux :
 - Les éléments en ligne (ou **inline**) sont affichés les uns à la suite des autres;
 - Les éléments conteneurs (ou **block**) amènent un retour à la ligne après leur affichage.
- Le flux fait en sorte que ces éléments s'affichent naturellement, les uns à la suite des autres, en respectant les spécificités de chaque type.
- La propriété **float** permet de sortir un élément du flux, elle définit si l'élément suivant doit s'aligner à droite ou à gauche, sauf des éléments boites définis en position absolue.
- Valeurs possibles :
 - **left** : l'élément génère une boite de bloc qui flotte à gauche et le contenu s'écoule sur son flanc droit en commençant en haut.
 - **right** : identique mais les éléments suivants flottent à droite.
 - **none** : pas de flottement.
- Lorsqu'il n'y a pas assez de place pour le flottant dans la largeur de la ligne, celui-ci se déplace vers le bas, d'une ligne à l'autre, jusqu'à en trouver une suffisamment large. Une boite flottante doit avoir une largeur explicite (sinon, elle prendrait tout l'espace horizontal disponible).

69

LE MODÈLE DES BOÎTES

Les propriétés float et clear :

- La propriété **float** va nous permettre de faire « flotter » des éléments HTML à gauche ou à droite dans une page web.
- On utilisera généralement soit la valeur **right** (l'élément ciblé flotte à droite), soit **left** (l'élément ciblé flotte à gauche) avec cette propriété.
- La propriété **float** va faire sortir un élément du flux normal de la page.
Il y a trois choses à retenir lorsque l'on utilise cette propriété :
 - Les éléments suivants un élément flottant vont se positionner à côté de celui-ci par défaut. Nous allons pouvoir annuler ce comportement grâce à la propriété **clear** ;
 - Les éléments positionnés de façon absolue avec **position:absolute** ignoreront la propriété **float** (ils ne pourront pas flotter) ;
 - Un élément flottera toujours dans les limites (en terme de largeur) de son élément parent conteneur.

70

LE MODÈLE DES BOÎTES

Arrêter le flottement avec clear :

- La propriété **clear** va nous permettre de contrôler le comportement d'éléments flottants.
- Cette propriété va nous permettre d'empêcher des éléments de venir se positionner aux côtés d'éléments flottants.
- Cette propriété peut prendre les valeurs suivantes :
 - **Left** : neutralise l'effet d'un **float:left** ;
 - **Right** : neutralise l'effet d'un **float:right** ;
 - **Both** : neutralise l'effet d'un **float:left** et d'un **float:right** ;
- Faîtes bien attention à appliquer la propriété **clear** à l'élément qui va venir flotter par défaut autour de l'élément flottant, pas à celui qui possède la propriété **float**.

71

LE MODÈLE DES BOÎTES

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>
```

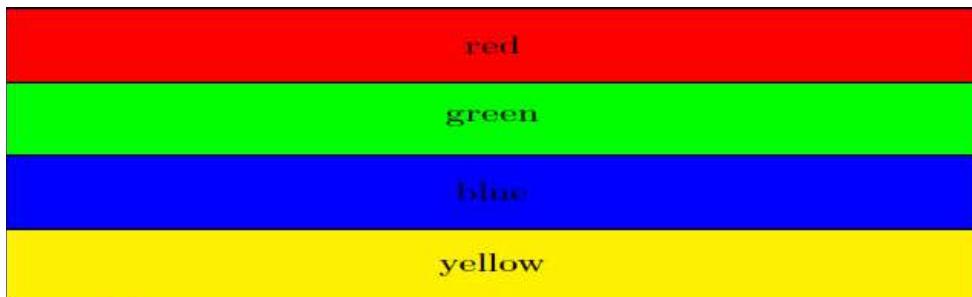
Le fichier **style.css**

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

72

LE MODÈLE DES BOÎTES

- Le résultat du code précédent :



On veut placer le bleu à gauche du vert

73

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

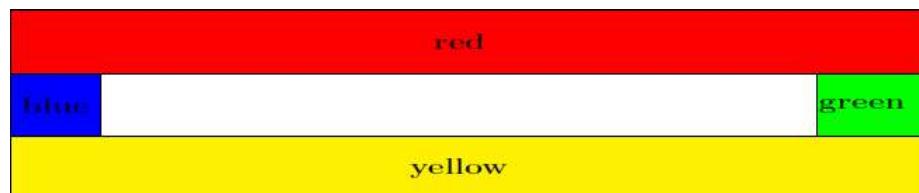


Mais le jaune, on voulait qu'il reste à la ligne

74

Dans ce cas, il faut ajouter :

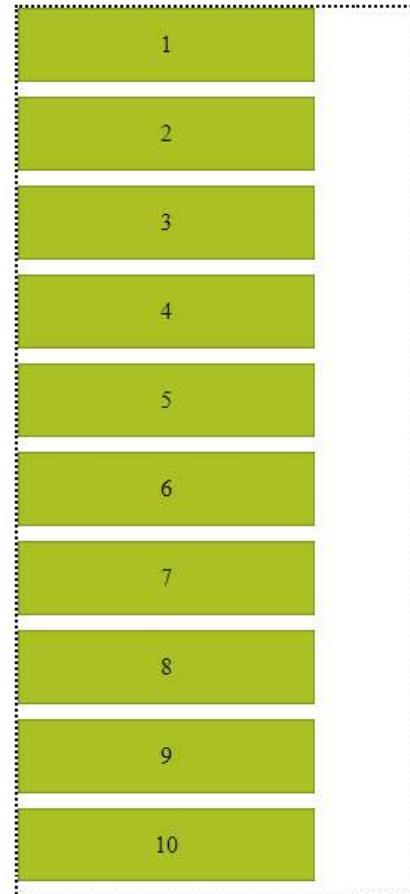
```
.component4 {  
    clear: both;  
    background-color: yellow;  
}
```



75

```
<div class="main">  
    <div class="left">1</div>  
    <div class="right">2</div>  
    <div class="left clear-left">3</div>  
    <div class="left">4</div>  
    <div class="left clear-left">5</div>  
    <div class="left">6</div>  
    <div class="left">7</div>  
    <div class="left">8</div>  
    <div class="left">9</div>  
    <div class="right clear-left">10</div>  
</div>
```

```
.main { width: 800px;  
        border: 2px dotted;  
    }  
.main div {background: #abc123;  
            text-align: center;  
            line-height: 50px;  
            width: 200px;  
            box-shadow: inset 0 0 1px #000;  
            margin-bottom: 10px;  
    }
```



76

```

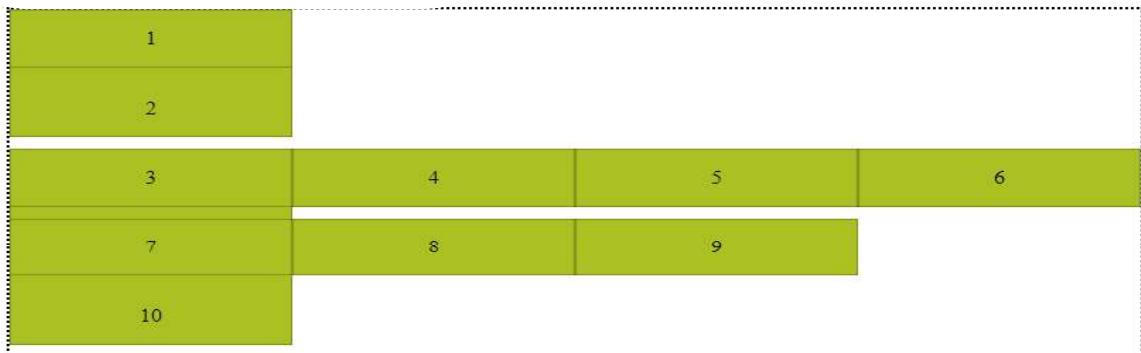
<div class="main">
  <div class="left">1</div>
  <div class="right">2</div>
  <div class="left clear-left">3</div>
  <div class="left">4</div>
  <div class="left clear-left">5</div>
  <div class="left">6</div>
  <div class="left">7</div>
  <div class="left">8</div>
  <div class="left">9</div>
  <div class="right clear-left">10</div>
</div>

```

```

.main { width: 800px; border: 2px dotted; }
.main div { background: #abc123; text-align: center; line-height: 50px; width: 200px; box-shadow: inset 0 0 1px #000; margin-bottom: 10px; }
div.left { float: left; }

```



77

```

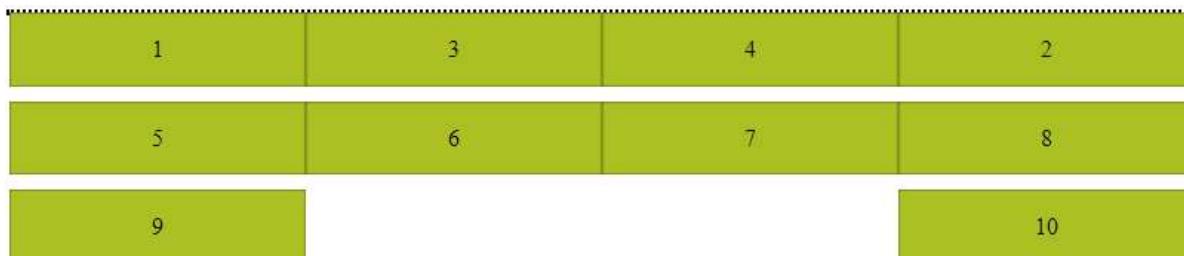
<div class="main">
  <div class="left">1</div>
  <div class="right">2</div>
  <div class="left clear-left">3</div>
  <div class="left">4</div>
  <div class="left clear-left">5</div>
  <div class="left">6</div>
  <div class="left">7</div>
  <div class="left">8</div>
  <div class="left">9</div>
  <div class="right clear-left">10</div>
</div>

```

```

.main { width: 800px; border: 2px dotted; }
.main div { background: #abc123; text-align: center; line-height: 50px; width: 200px; box-shadow: inset 0 0 1px #000; margin-bottom: 10px; }
div.left { float: left; }
div.right { float: right; }

```



78

```

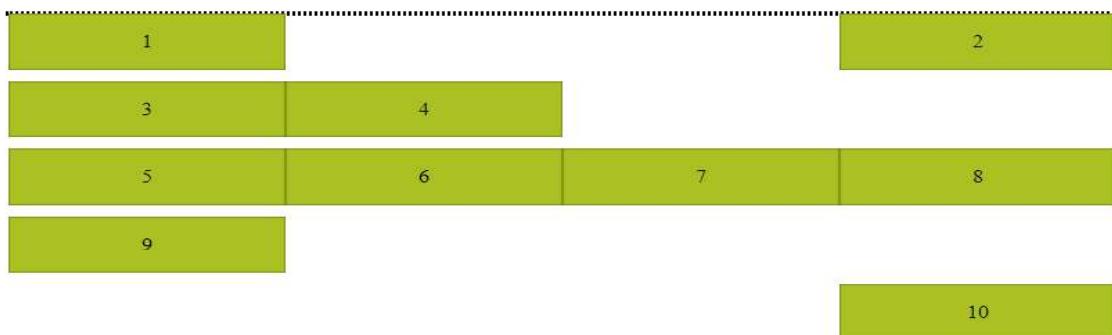
<div class="main">
  <div class="left">1</div>
  <div class="right">2</div>
  <div class="left clear-left">3</div>
  <div class="left">4</div>
  <div class="left clear-left">5</div>
  <div class="left">6</div>
  <div class="left">7</div>
  <div class="left">8</div>
  <div class="left">9</div>
  <div class="right clear-left">10</div>
</div>

```

```

.main { width: 800px; border: 2px dotted; }
.main div { background: #abc123; text-align: center; line-height: 50px; width: 200px; box-shadow: inset 0 0 1px #000; margin-bottom: 10px; }
div.left { float: left; }
div.right { float: right; }
div.clear-left { clear: left; }

```



79

LE MODÈLE DES BOÎTES

La propriété position :

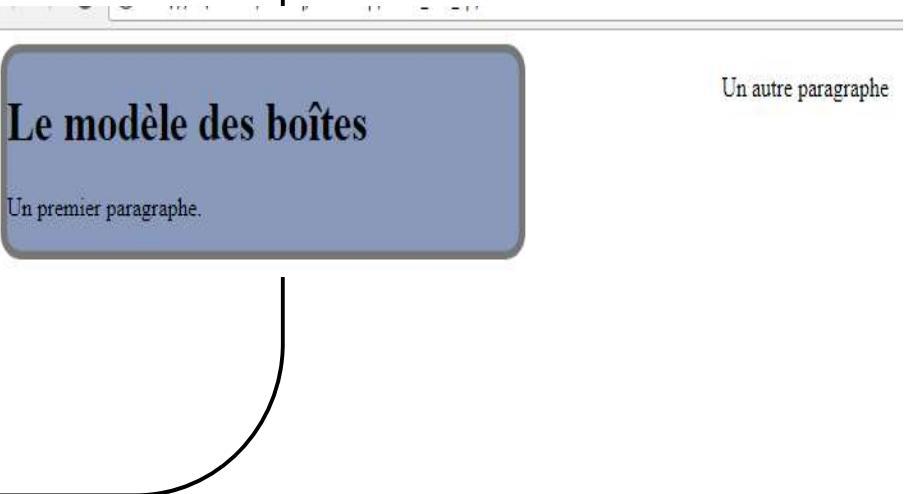
- On gère le positionnement de nos éléments HTML en CSS grâce à la propriété **position** qui supporte quatre valeurs différentes :
 - static** : correspond au positionnement par défaut des éléments HTML dans une page.
 - relative** : relativement par rapport à sa position par défaut.
Si on positionne un élément HTML de façon relative et qu'on lui ajoute **right:50px**, l'élément sera déplacé de 50 pixels vers la droite par rapport à sa position par défaut.
 - fixed** : toujours rester à la même place, même si l'un de vos visiteurs descend ("scroll") dans la page. très utile pour conserver un élément constamment visible, comme un menu ou un sommaire par exemple.
 - Absolute**: positionner un élément de façon relative par rapport à son parent le plus proche auquel on a appliqué un positionnement spécifique (relative, fixed ou absolute).

LE MODÈLE DES BOÎTES

```
div{  
background-color: #89B;  
width: 400px;  
border: 5px solid #777;  
border-radius: 15px;  
}
```

```
.para1{  
position: relative;  
left: 50px;  
width: 50%;  
}
```

```
.para2{  
position: absolute;  
right: 70px;  
top :10px  
}
```



81

LE MODÈLE DES BOÎTES

La propriété display :

- Les éléments HTML vont s'afficher soit sous forme de « bloc », soit « en ligne ».
- La propriété display est très puissante, puisqu'elle va nous permettre de maîtriser la façon dont un élément va être affiché, et ainsi de gérer la mise en page de nos éléments.
- La propriété display peut prendre de nombreuses valeurs nous permettant de gérer précisément la façon dont chaque élément va être affiché :
 - **Display : inline** l'élément ciblé va se comporter comme un élément de type *inline* et donc n'occuper que la largeur qui lui est strictement nécessaire.
 - **Display : block** les éléments ciblés vont se comporter comme des éléments HTML de type *block* et ainsi prendre toute la largeur disponible.
 - **Display : inline-block** permet d'emprunter certaines propriétés des éléments de type *block* et certaines propriétés des éléments de type *inline*.
Ainsi, l'élément en soi va être de type *inline* tandis que ce qu'il contient (l'intérieur de la boîte) va être considéré comme étant de type *block*.
 - **Display : none** permet tout simplement de ne pas afficher un élément.
Cela peut se révéler très pratique dans de nombreux cas, notamment lorsqu'on veut modifier l'affichage de nos pages selon l'appareil utilisé par nos visiteurs (ordinateur de bureau, téléphone portable, tablette, etc.).

82

LE MODÈLE DES BOÎTES

```
div{  
background-color: #89B;  
width: 400px;  
border: 5px solid #777;  
border-radius: 15px;  
box-shadow: 3px 3px 9px orange;  
}  
  
.para1{  
display: inline-block;  
border: 1px solid black;  
width: 50%;  
}  
  
.para2{  
display: inline-block;  
border: 1px solid black;  
width: 40%;  
}
```



83

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

Considérons l'exemple suivant :

Le fichier `flex.html`

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title> flex </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`

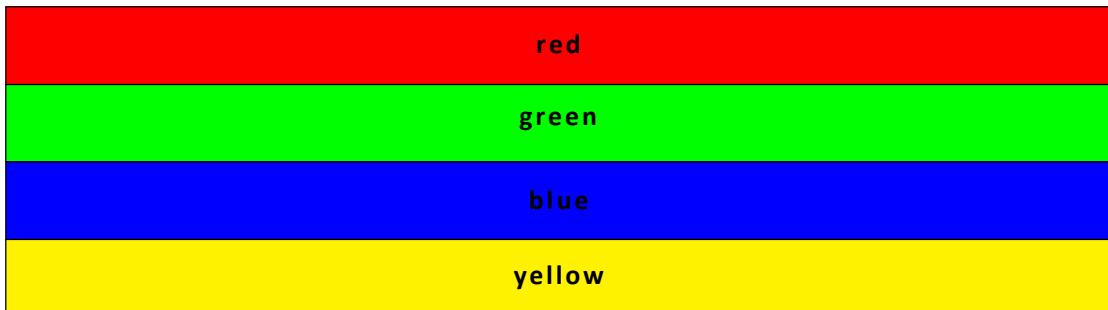
```
.component1 {  
background-color: red;  
}  
.component2 {  
background-color: green;  
}  
.component3 {  
background-color: blue;  
}  
.component4 {  
background-color: yellow;  
}  
.container {  
background-color: black;  
}
```

84

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

Le résultat du code précédent :



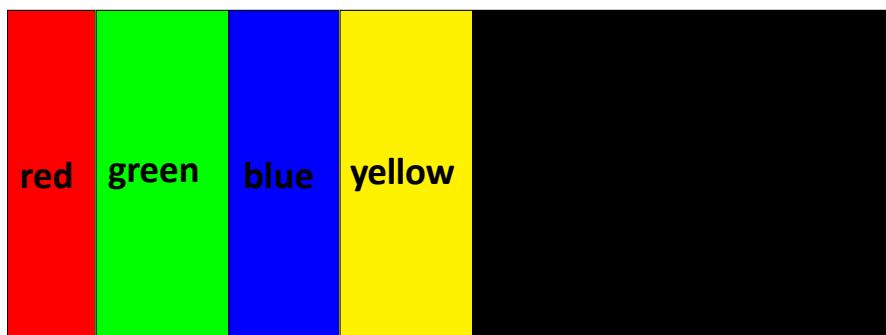
Et si on veut placer les balises `div` les unes à côté des autres?

85

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

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



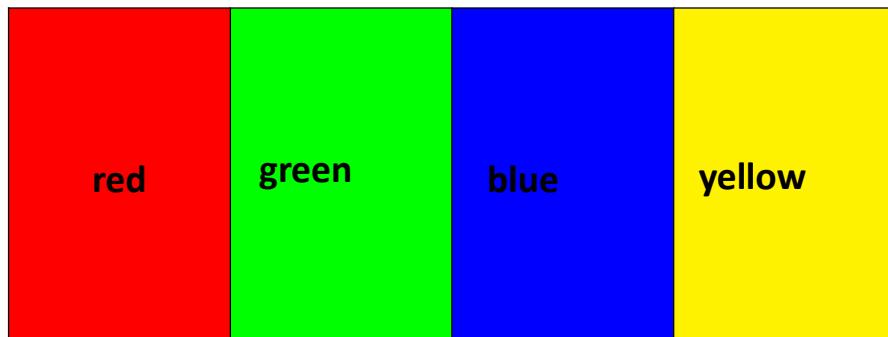
Et si on veut donner la même largeur à toutes les `div` ?

86

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

```
.container {  
    background-color: black;  
    display: flex;  
}  
.container > div {  
    width: 30%;  
}
```



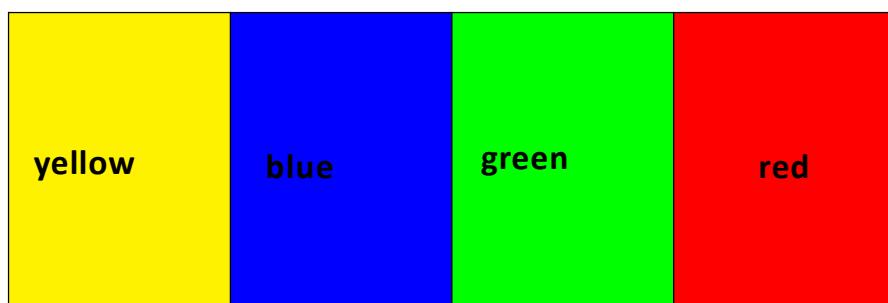
Et si on veut afficher les `div` dans le sens opposé?

87

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

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



88

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

Autres valeurs de flex-direction

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

Remarque

- Les 4 div de l'exemple précédent avaient 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 ?
⇒ il faut utiliser la propriété **flex-wrap**.

89

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

La propriété flex-wrap

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

- **nowrap** (par défaut) : organiser les boîtes 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.

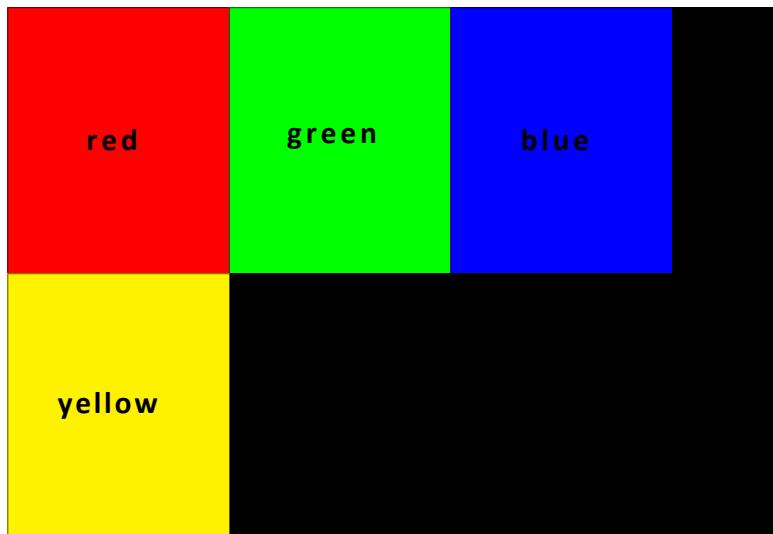
90

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

Revenons à l'exemple précédent

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



91

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

La propriété `flex-flow`

- C'est le raccourci de `flex-direction` et `flex-wrap`
- Exemple : `flex-flow : row wrap;`

92

LE MODÈLE DES BOÎTES

La propriété display : la valeur Flex

Trois propriétés pour l'alignement des boîtes

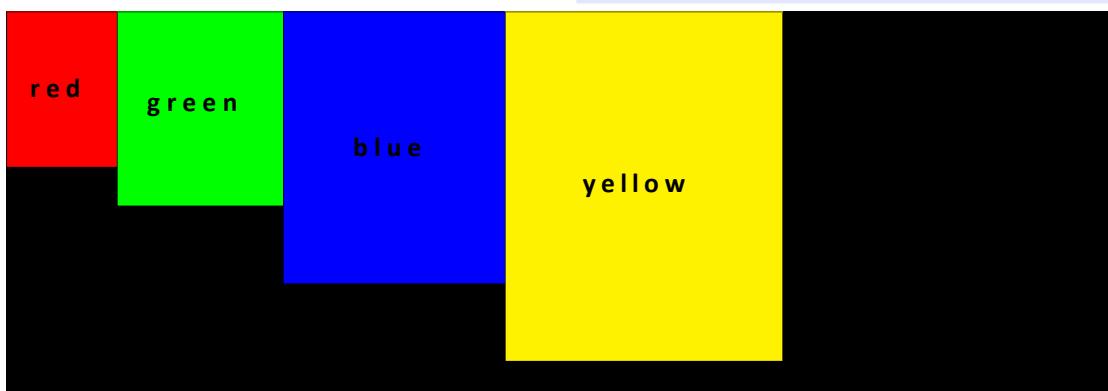
- **justify-content** : permet d'aligner horizontalement les boîtes d'une même ligne (si elles n'utilisent pas tout l'espace valable).
- **align-items** : permet de définir la disposition verticale des boîtes d'une même ligne (si elles n'utilisent pas tout l'espace valable).
- **align-content** : permet d'aligner verticalement les lignes de boîtes d'un conteneur (sans effet si toutes les boîtes sont sur une seule ligne).

93

LE MODÈLE DES BOÎTES

Pour la suite, on considère le code CSS suivant :

```
.component1 {  
    background-color: red;  
    width : 10%;  
    height: 80px;  
}  
.component2 {  
    background-color: green;  
    width : 15%;  
    height: 100px;  
}  
.component3 {  
    background-color: blue;  
    width : 20%;  
    height: 140px;  
}  
.component4 {  
    background-color: yellow;  
    width : 25%;  
    height: 180px;  
}  
.container {  
    background-color: black;  
    display: flex;  
    height: 250px;  
}
```



94

La propriété display : la valeur Flex

`justify-content` : pour l'alignement horizontal des boîtes

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

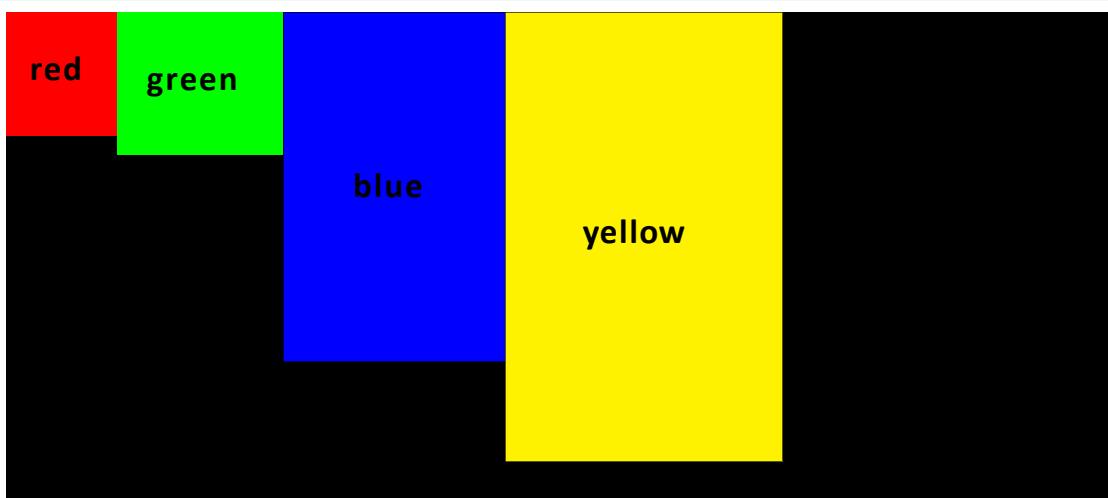
95

LE MODÈLE DES BOÎTES

Résultat de: `justify-content: flex-start;`

`flex-start` est la valeur par défaut de `justify-content` et ne change donc rien de la configuration précédente.

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    justify-content: flex-start;  
}
```

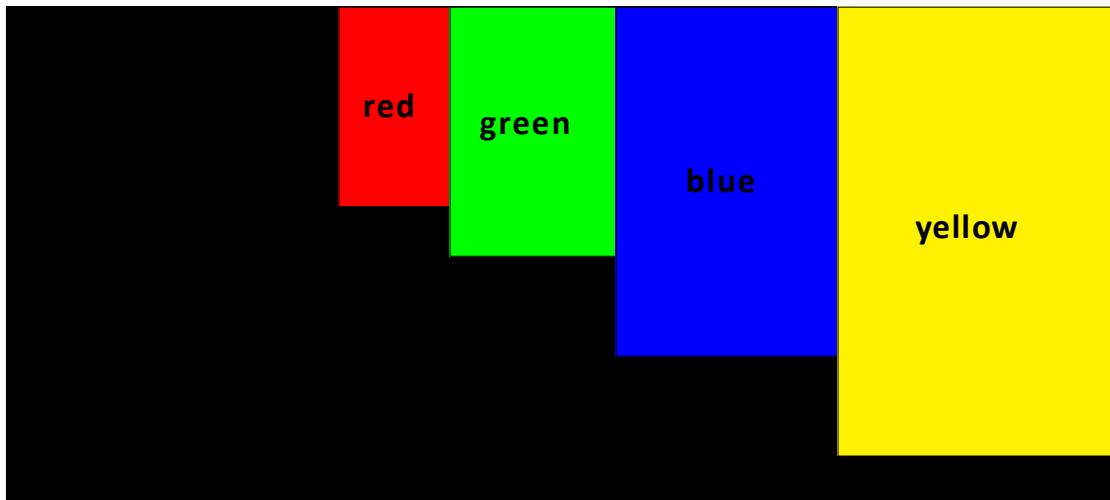


96

LE MODÈLE DES BOÎTES

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

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    justify-content: flex-end;  
}
```

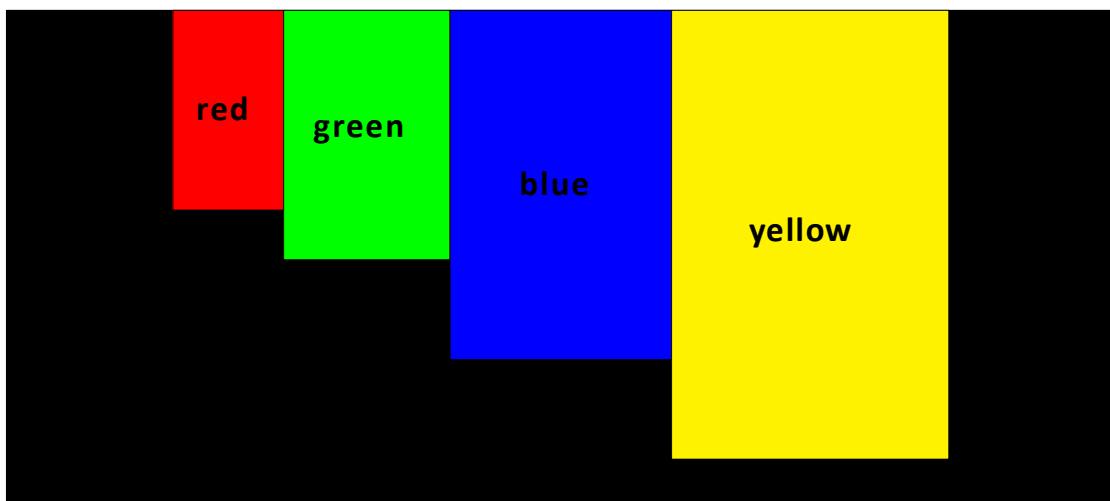


97

LE MODÈLE DES BOÎTES

Résultat de : `justify-content: center;`

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    justify-content: center;  
}
```

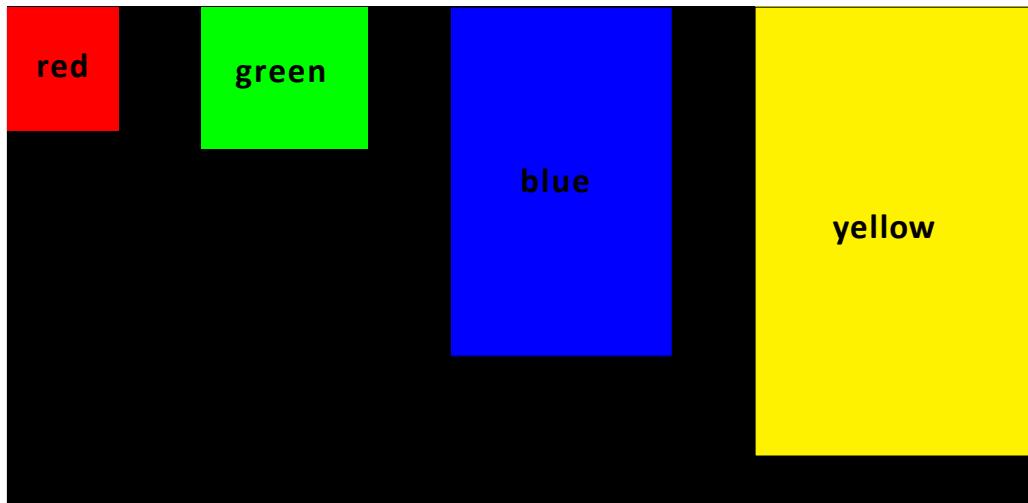


98

LE MODÈLE DES BOÎTES

Résultat de : `justify-content: space-between;`

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    justify-content: space-between;  
}
```

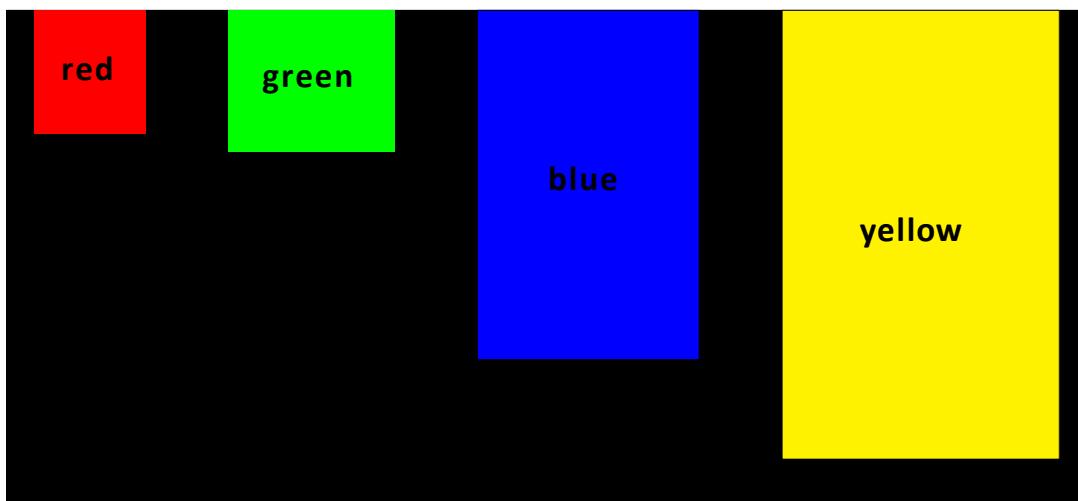


99

LE MODÈLE DES BOÎTES

Résultat de : `justify-content: space-around;`

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    justify-content: space-around;  
}
```



100

align-items : pour l'alignement vertical

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

101

LE MODÈLE DES BOÎTES

Pour la suite, on considère les codes HTML et CSS suivants

```
<div class="container" >
  <div class=component1>
    red
  </div>
  <div class=component2>
    green <br> green
  </div>
  <div class=component3>
    blue <br> blue <br> blue
  </div>
  <div class=component4>
    yellow <br> yellow <br>
    yellow <br> yellow
  </div>
</div>
```

```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container > div {
  width: 30%;
}
.container {
  background-color: black;
  display: flex;
  height: 200px;
}
```

102

LE MODÈLE DES BOÎTES

Le code précédent permet l'emplacement suivant :



103

LE MODÈLE DES BOÎTES

Résultat de : align-items: stretch;

stretch est la valeur par défaut de align-items et ne change donc rien de la configuration de départ.

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    align-items: stretch;  
}
```

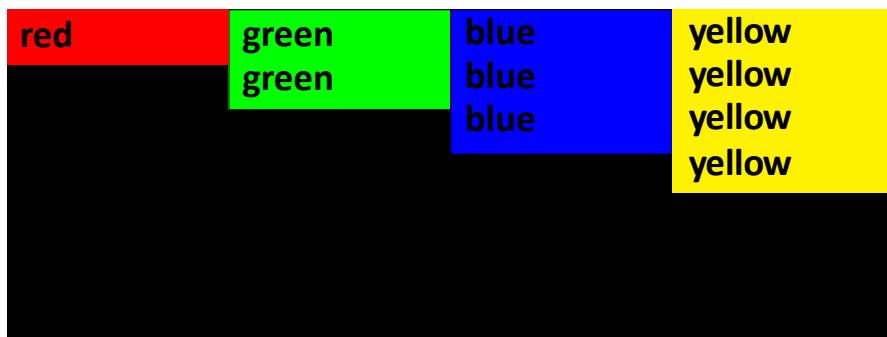


104

LE MODÈLE DES BOÎTES

Résultat de : align-items: flex-start;

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    align-items: flex-start;  
}
```

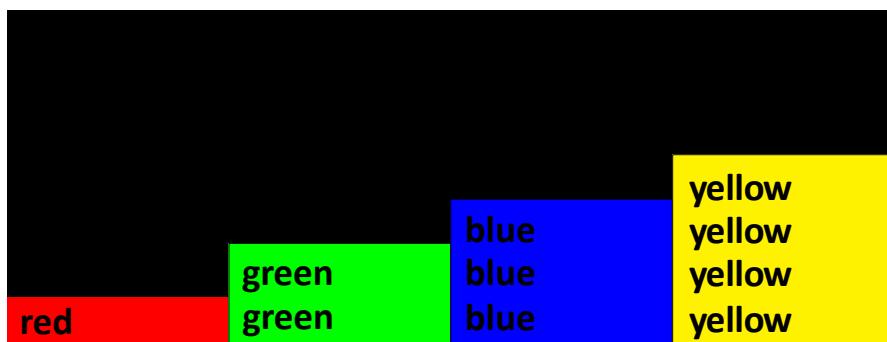


105

LE MODÈLE DES BOÎTES

Résultat de : align-items: flex-end;

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    align-items: flex-end;  
}
```

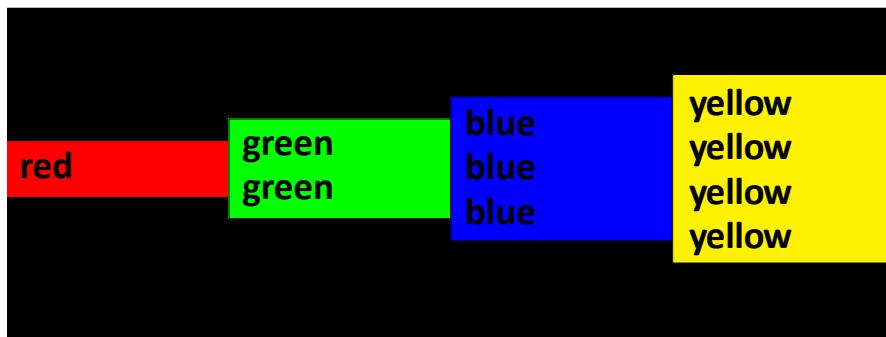


106

LE MODÈLE DES BOÎTES

Résultat de : `align-items: center;`

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    align-items: center;  
}
```



107

LE MODÈLE DES BOÎTES

Pour pouvoir tester la valeur `baseline`, on ajoute un `padding-top` pour quelques components :

```
.component1 {  
    background-color: red;  
}  
.component2 {  
    background-color: green;  
    padding-top: 20px;  
}  
.component3 {  
    background-color: blue;  
    padding-top: 10px;  
}  
.component4 {  
    background-color: yellow;  
    padding-top: 30px;  
}
```

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    align-items: baseline;  
}  
.container > div {  
    width: 30%;  
}
```

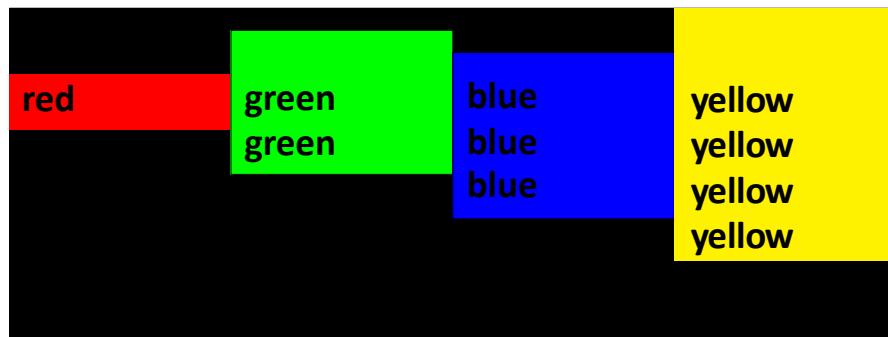
Le HTML ne change pas !

108

LE MODÈLE DES BOÎTES

Résultat de : align-items: baseline;

```
.container {  
    background-color: black;  
    display: flex;  
    height: 200px;  
    align-items: baseline;  
}
```

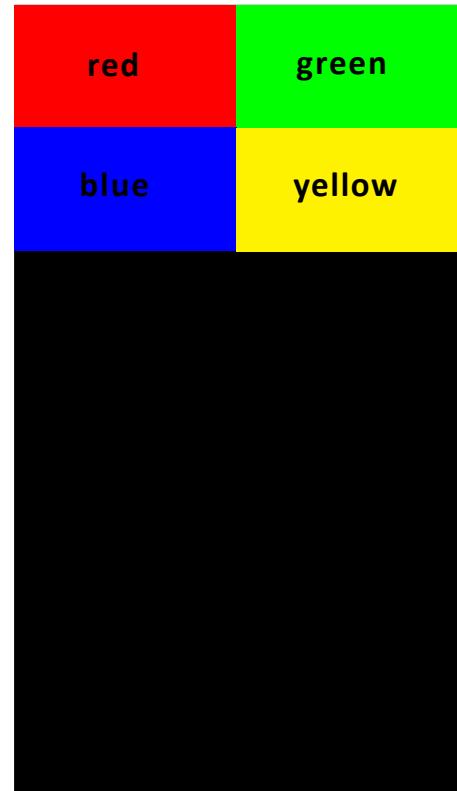


109

LE MODÈLE DES BOÎTES

Résultat de : align-content: flex-start;

```
.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%;  
    height: 50px;  
}
```

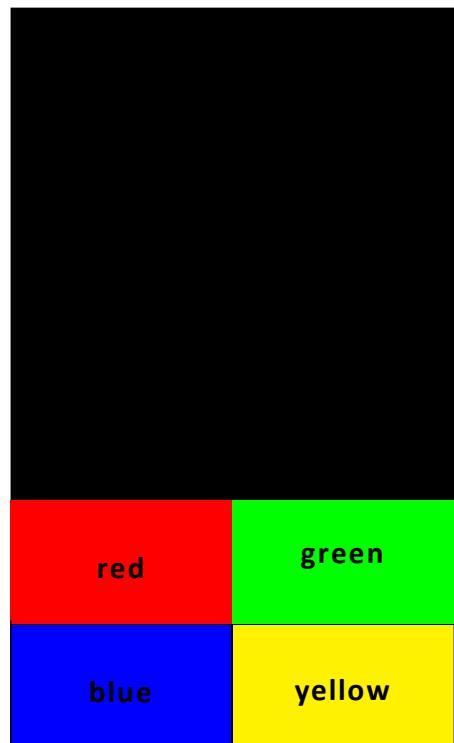


110

LE MODÈLE DES BOÎTES

Résultat de : align-content: flex-end;

```
.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-end;  
    height: 150px;  
}  
.container > div {  
    width : 50%;  
    height: 50px;  
}
```

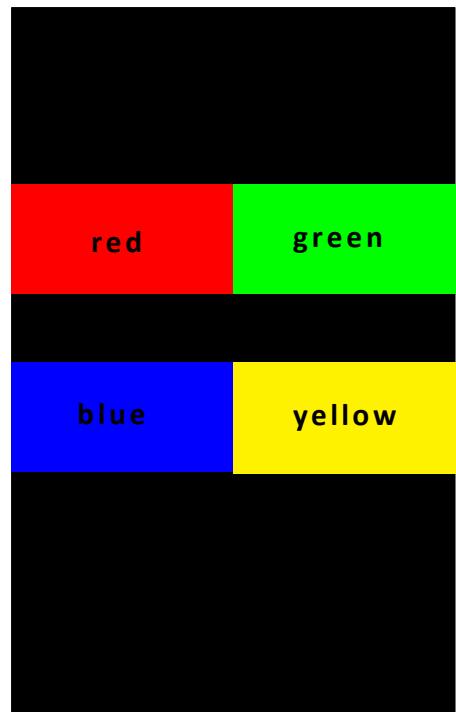


111

LE MODÈLE DES BOÎTES

Résultat de : align-content: center;

```
.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: center;  
    height: 150px;  
}  
.container > div {  
    width : 50%;  
    height: 50px;  
}
```

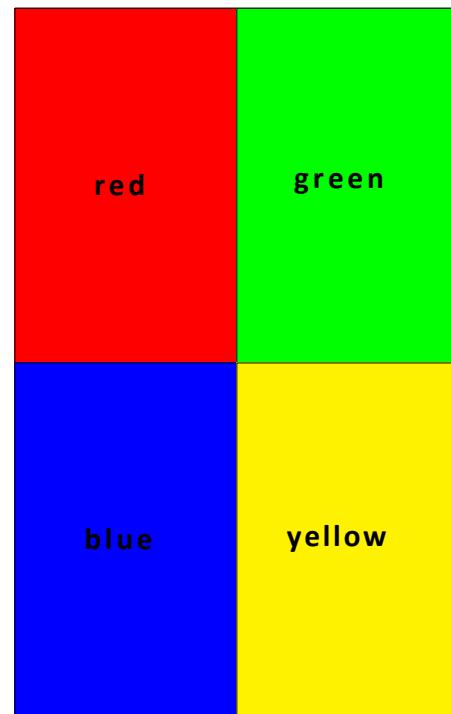


112

LE MODÈLE DES BOÎTES

Résultat de : align-content: stretch;

```
.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: stretch;  
    height: 150px;  
}  
.container > div {  
    width : 50%;  
    height: 50px;  
}
```

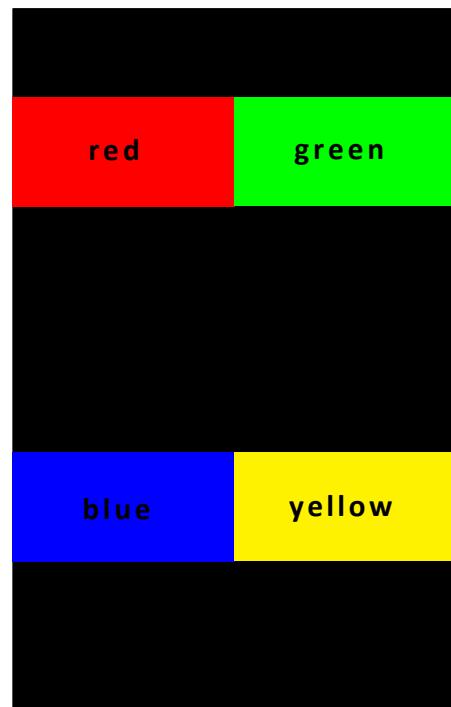


113

LE MODÈLE DES BOÎTES

Résultat de : align-content: space-around;

```
.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: space-around;  
    height: 150px;  
}  
.container > div {  
    width : 50%;  
    height: 50px;  
}
```

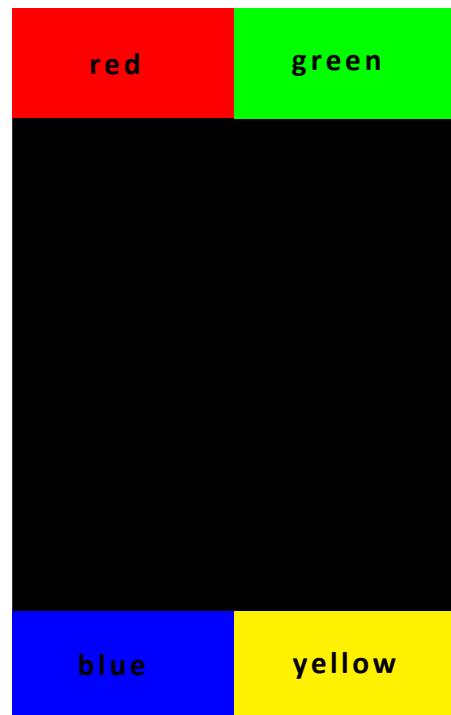


114

LE MODÈLE DES BOÎTES

Résultat de : **align-content: space-between;**

```
.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: space-between;  
    height: 150px;  
}  
.container > div {  
    width: 50%;  
    height: 50px;  
}
```



115

LE MODÈLE DES BOÎTES

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.

order

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

align-self

- Permet de changer la valeur d'**align-items** pour un block donné.

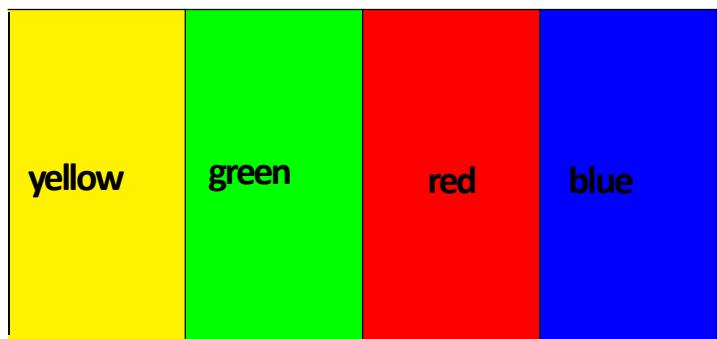
116

LE MODÈLE DES BOÎTES

La propriété order :

```
.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 :



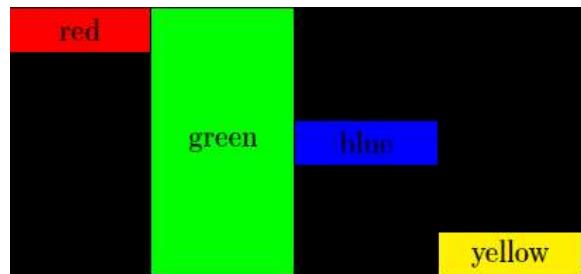
117

LE MODÈLE DES BOÎTES

La propriété align-self :

```
.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 :



118

Propriétés pour modifier la largeur

- **flex-grow** : permet d'agrandir certains blocks par rapport à la taille des autres. Elle précise de combien croître la taille des enfants.
- **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 spécifier la largeur initiale d'un composant, avant qu'un éventuel espace disponible soit distribué selon les facteurs flex.
- ...

119

LE MODÈLE DES BOÎTES

La propriété **flex-grow**:

```
.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



120

Pour mieux comprendre Flexbox

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

121

LE MODÈLE DES BOÎTES

La propriété display : la valeur Grid

Considérons l'exemple suivant :

Le fichier grid.html

```
<!DOCTYPE html>
<html>
  <head>
    <title> grid </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

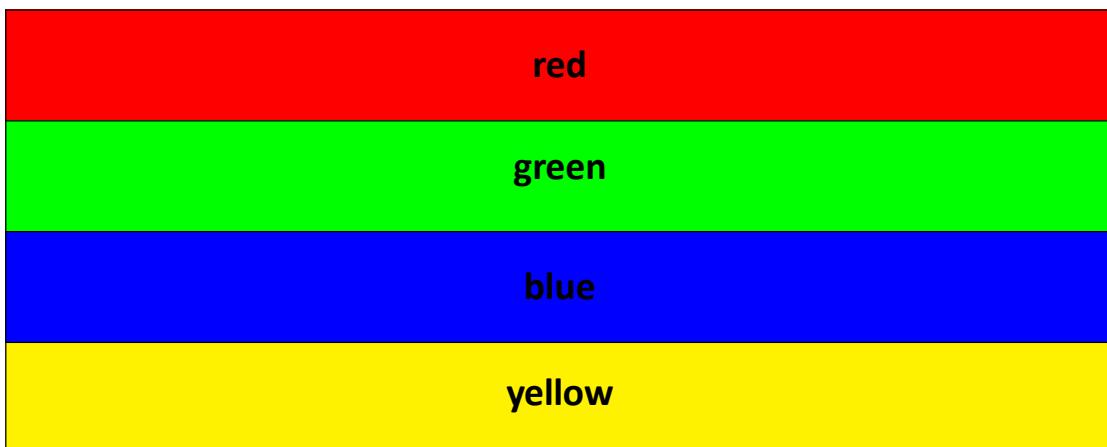
```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: grid;
}
.container > div {
  height: 100px;
}
```

122

LE MODÈLE DES BOÎTES

La propriété display : la valeur Grid

On aura le résultat suivant :



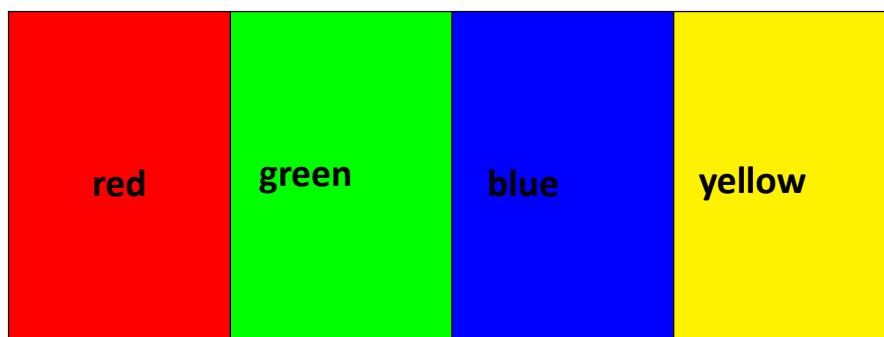
123

LE MODÈLE DES BOÎTES

La propriété display : la valeur 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.

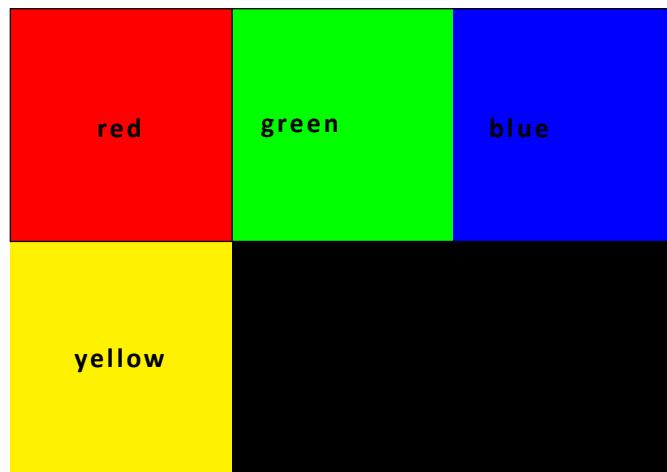
124

LE MODÈLE DES BOÎTES

La propriété display : la valeur 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;  
}
```



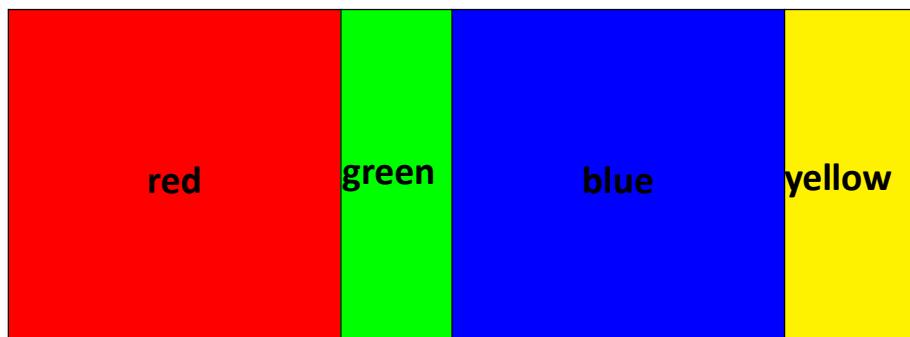
125

LE MODÈLE DES BOÎTES

La propriété display : la valeur 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;  
}
```



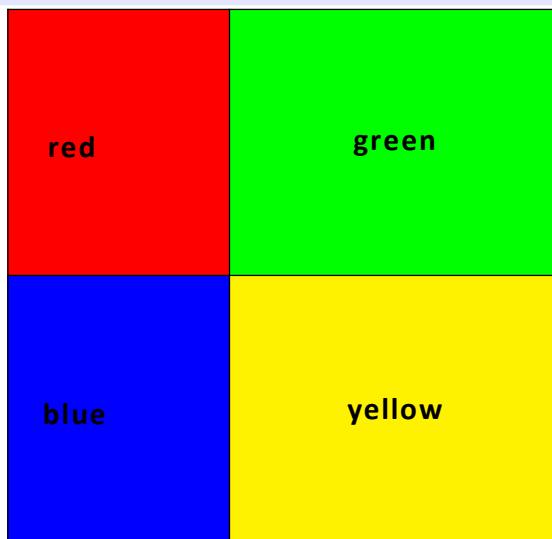
On peut remplacer l'écriture `grid-template-columns: 1.5fr 0.5fr 1.5fr 0.5fr;`
par `grid-template-columns: repeat(2, 1.5fr 0.5fr);`

126

LE MODÈLE DES BOÎTES

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

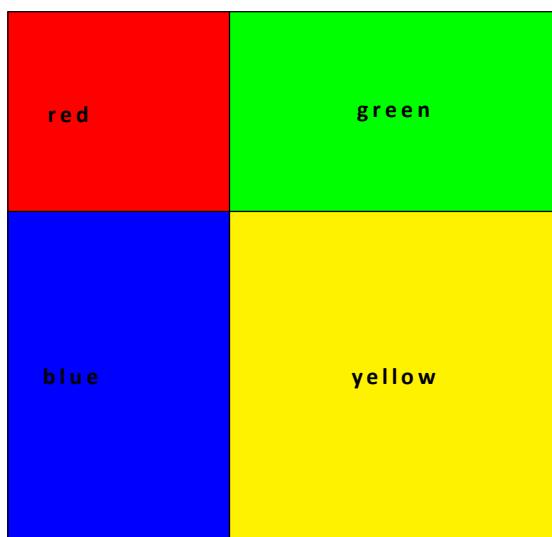
```
.container {  
background-color: black;  
display: grid;  
grid-template-columns: 40% 60%; /*on peut aussi utiliser px:pixel  
ou fr:fraction:nombre de part de l'espace disponible*/  
}  
.container > div {  
height: 100px;  
}
```



127

LE MODÈLE DES BOÎTES

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



128

On peut aussi remplacer les deux propriétés `grid-template-columns` et `grid-template-rows` par `grid-template`

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

129

LE MODÈLE DES BOÎTES

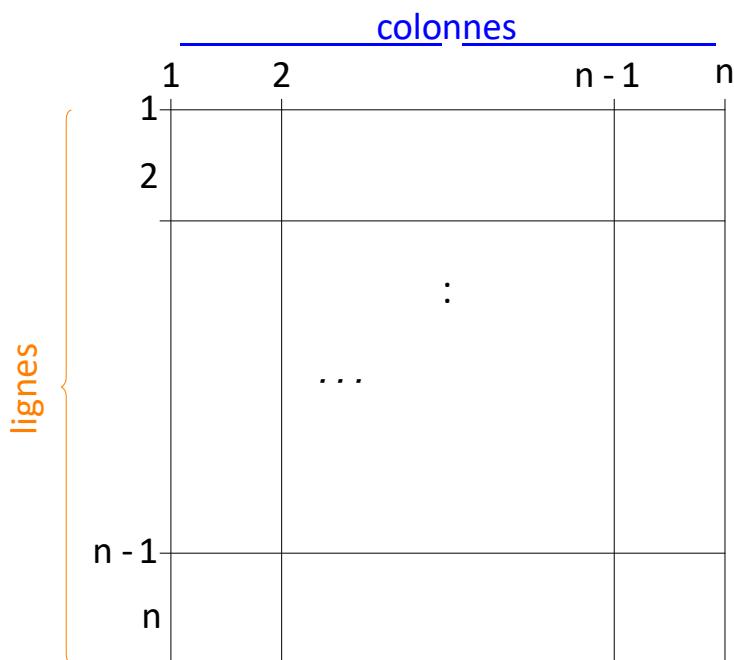
Grid : autres propriétés

- `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.

130

LE MODÈLE DES BOÎTES

On peut aussi placer les composants en précisant pour chacun entre quelles colonnes et lignes il faut le placer :

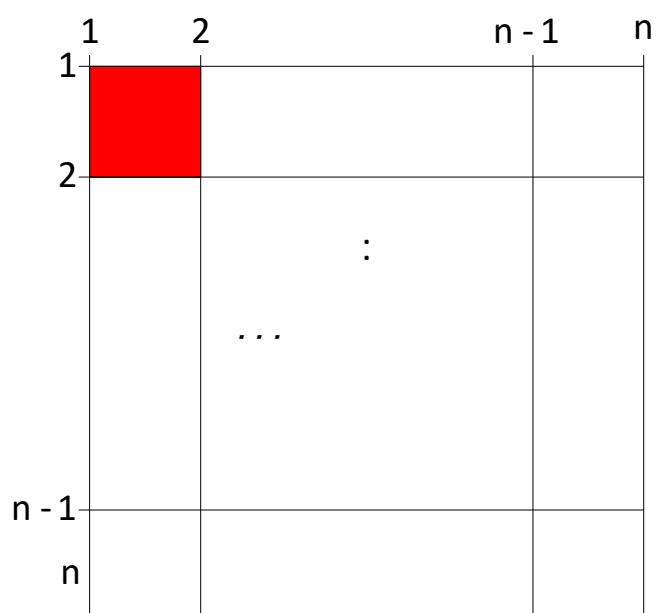


131

LE MODÈLE DES BOÎTES

Exemple

```
grid-column-start: 1;  
grid-column-end: 2;  
grid-row-start: 1;  
grid-row-end: 2;
```



132

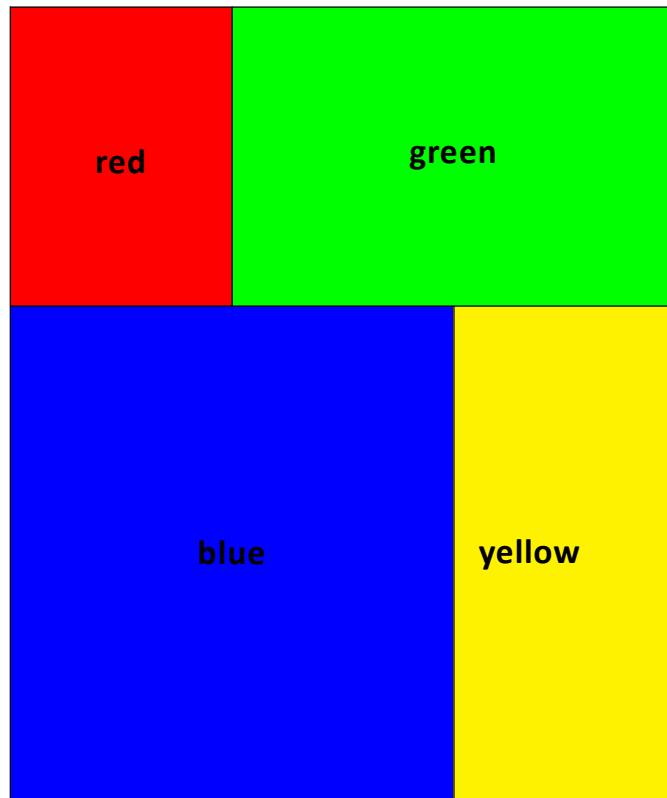
LE MODÈLE DES BOÎTES

```
.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);  
}
```

133

LE MODÈLE DES BOÎTES

Le résultat est :



134