

CSS3 : Cascading Style Sheets

Institut supérieur de gestion 2024-2025

Moez Hammami



Plan

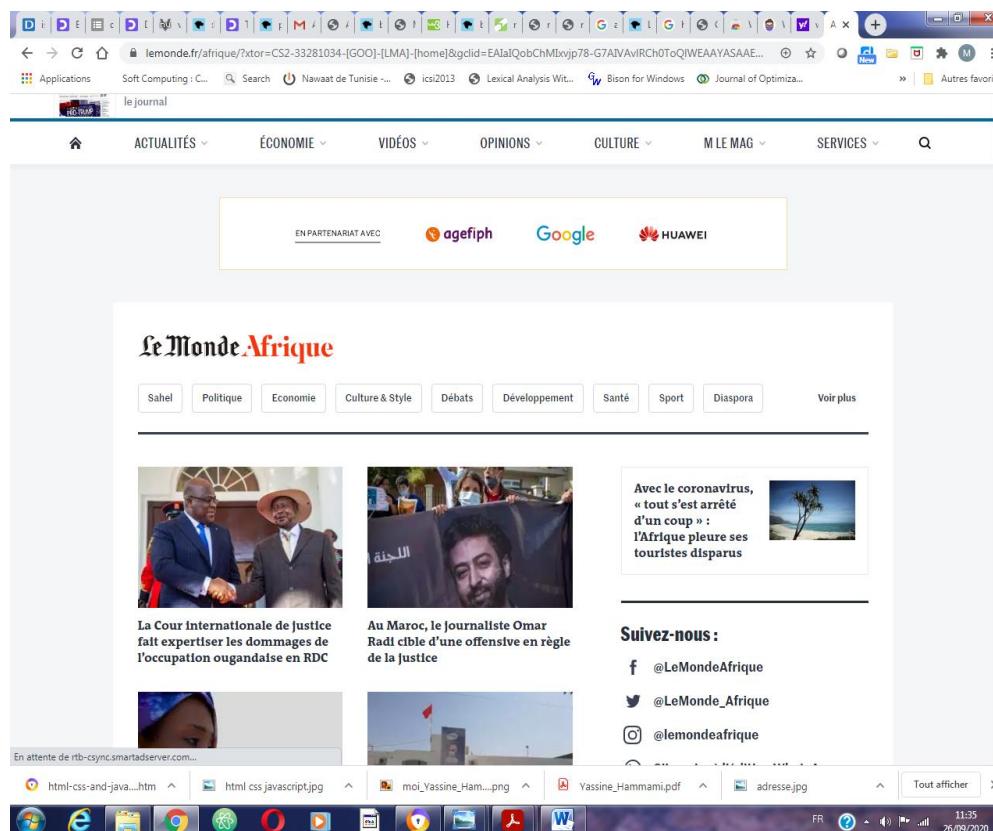
- Introduction
- Comment définir ces styles
- Les sélecteurs CSS
 - Sélecteurs de classe
 - Sélecteurs #id
 - Sélecteur *
 - Sélecteur élément
 - Sélecteur d'attributs
 - Pseudo-éléments
 - Pseudo-classes
- Les propriétés basiques
- Emplacement des boites avec flex-box
- Emplacement des boites avec Grid

CSS: c'est quoi?

- C'est un langage utilisé pour décrire la présentation d'un document HTML (ou XML)
- Il permet de définir l'apparence des textes (police, couleur, taille,...)
- Il permet de définir l'agencement de la page (marges, arrière plan..)
- Il définit la façon dont les éléments doivent être affichés dans le navigateur.
- Il permet une définition homogène des styles dans un site Web
- Il facilite la réutilisation des styles et leur généralisation sur les différentes pages du site

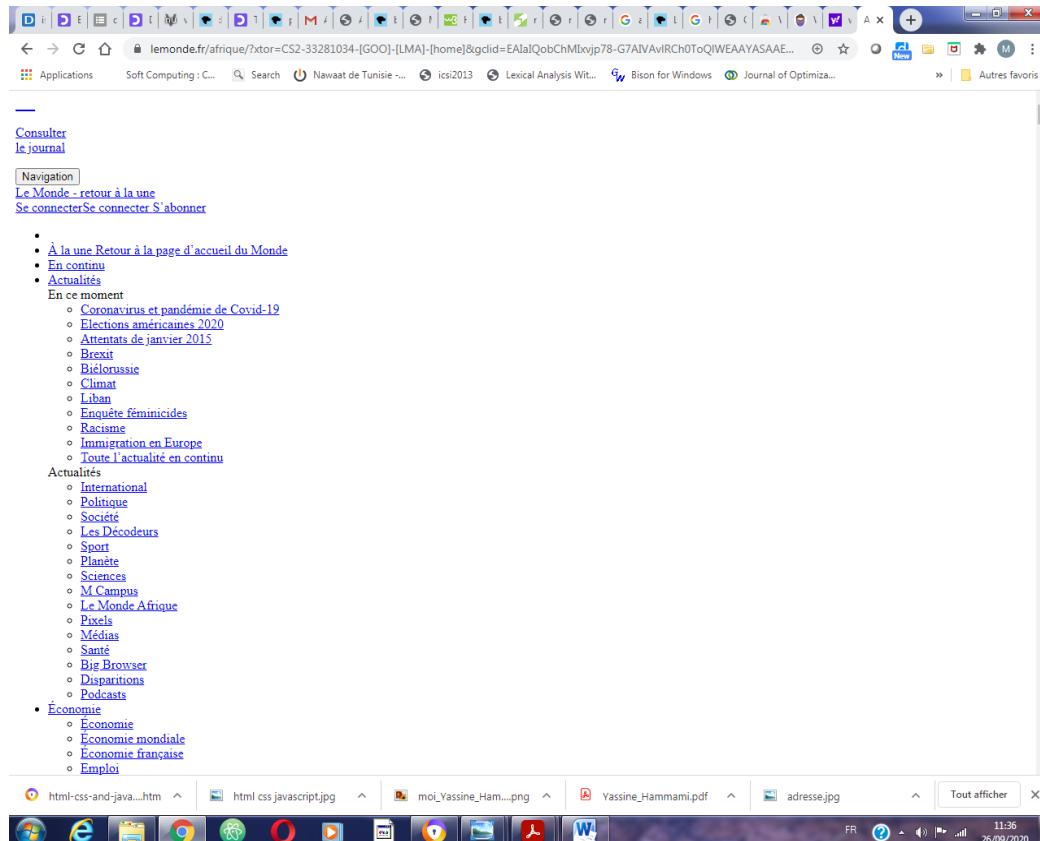
Pourquoi CSS+HTML :

- CSS améliore fortement le rendu de la page web. Le HTML définit le contenu. Le CSS permet, lui, d'arranger le contenu et de définir la présentation : couleurs, image de fond, marges, taille du texte...
- Voici un site utilisant HTML+CSS



Pourquoi CSS+HTML :

- CSS améliore fortement le rendu de la page web. Le HTML définit le contenu. Le CSS permet, lui, d'arranger le contenu et de définir la présentation : couleurs, image de fond, marges, taille du texte...
- Le même site en utilisant HTML uniquement (on peut désactiver le CSS en installant le addon « web developer » et en désactivant tous les styles)



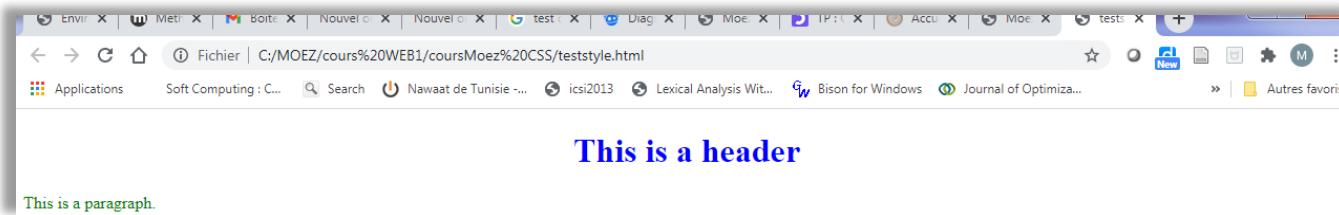
Comment définir ces styles (3 façons)

première façon

- Utiliser l'attribut **style** après le nom de la balise ouvrante dans le code HTML **<balise style="property: value;"> ...</balise>**

Exemple

```
<!DOCTYPE html>  
  
<html>  
  
<body>  
    <h1 style="color:blue;text-align:center;">This is a header</h1>  
    <p style="color:green;">This is a paragraph.</p>  
</body>  
</html>
```



Comment définir ces styles (3 façons) deuxième façon

- Intégrer la balise `<style>` dans le partie `<head>` du fichier HTML
syntaxe

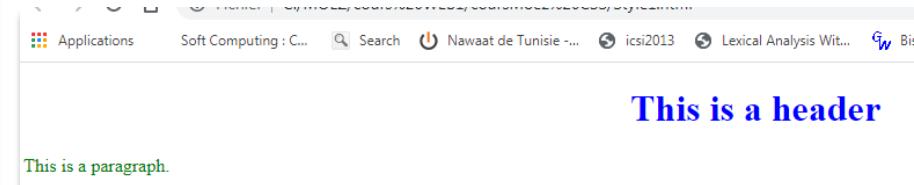
```
<head>
...
<style type="text/css">
    selector
    {
        property: value;
    }
</style>
</head>
```

Comment définir ces styles (3 façons) deuxième façon

- Intégrer la balise `<style>` dans le partie `<head>` du fichier HTML

Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Le Site Web</title>
    <style type="text/css">
      h1
      {
        color:blue;
        text-align:center;
      }
      p
      {
        color:green;
      }
    </style>
  </head>
  <body>
    <h1>This is a header</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```



Remarques :

- Tous les paragraphes du document seront affichés en vert et tous les titres I seront affichés en bleu et centrés.
- Avec HTML5 plus besoin de préciser le type pour la balise style. `<style type="text/css">`

Comment définir ces styles (3 façons) troisième façon

- **on place le contenu CSS dans un fichier (file.css) et on met le lien à ce fichier dans la partie head du fichier HTML**

Syntaxe du lien dans le fichier HTML

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

Exemple

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="utf-8" />  
    <title>Le Site Web</title>  
    <link rel="stylesheet" type="text/css" href="file.css">  
  </head>  
  <body>  
    <h1>This is a header</h1>  
    <p>This is a paragraph.</p>  
  </body>  
</html>
```

Contenu du fichier « file.css »

```
h1  
{  
  color:blue;  
  text-align:center;  
}  
p  
{  
  color:green;  
}
```

Sélecteurs CSS

- Pour pouvoir appliquer un style à un contenu, il va falloir le cibler, c'est-à-dire trouver un moyen d'indiquer qu'on souhaite appliquer tel style à un contenu en particulier du document HTML.
- Il existe différents types de sélecteurs en CSS : certains sélecteurs vont s'appuyer sur le nom des éléments, comme le sélecteur CSS **p** par exemple qui va servir à cibler tous les éléments **p** d'une page. Ce type de sélecteurs est appelé « **sélecteur d'éléments** »
- D'autres sélecteurs vont cibler les éléments selon leur **classe** ou leur **identifiant** déjà définis dans le fichier HTML
- D'autres, en revanche, vont être plus complexes et nous permettre de sélectionner un élément HTML en particulier ou un jeu d'éléments HTML **en fonction de leurs attributs ou même de leur état** : on va ainsi pouvoir appliquer des styles à un élément uniquement lorsque la souris de l'utilisateur passe dessus par exemple.

Sélecteurs CSS

Sélecteurs de classe

Le selecteur **.class** est un sélecteur qui permet de cibler tous les éléments HTML ayant comme classe « class »

Exemple

```
<!DOCTYPE html>
<html>
<head>
<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Le Site Web</title>
</head>
<body>
    <h1>page personnelle</h1>
    <div class="présentation">
        <h1>Mark Zuckerberg </h1>
        <p>créateur de Facebook</p>
    </div>

    <p class="principes">La clé dans tout ce que l'on fait, c'est de prioriser.  
C'est-à-dire de cerner les points importants à traiter pour nous, et de  
s'y tenir</p>
    <p class="principes">Une règle simple pour vraiment changer les  
choses, c'est de commencer toujours par le plus simple, et non par le  
plus difficile</p>
</body>
</html>
```

Contenu du Ficher style.css

```
.présentation
{
    background-color: yellow;
}

.principes
{
    color:red;
    font-weight: bold;
}

/*Les éléments ayant comme classe
“présentation” auront un fond jaune
et les elements ayant comme classe
“principes“ auront une couleur rouge
et une police ‘gras’*/
```

Sélecteurs CSS

Sélecteurs de classe

```
<!DOCTYPE html>
<html>
<head>
<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Le Site Web</title>
</head>
<body>
    <h1>page personnelle</h1>
    <div class="présentation">
        <h1>Mark Zuckerberg </h1>
        <p>créateur de Facebook</p>
    </div>

    <p class="principes">La clé dans tout ce que l'on fait, c'est de prioriser.  
C'est-à-dire de cerner les points importants à traiter pour nous, et de  
s'y tenir</p>
    <p class="principes">Une règle simple pour vraiment changer les  
choses, c'est de commencer toujours par le plus simple, et non par le  
plus difficile</p>
</body>
</html>
```

Contenu du Ficher style.css

```
.présentation
{
    background-color: yellow;
}

.principes
{
    color:red;
    font-weight: bold;
}

/*Les éléments ayant comme classe
“présentation” auront un fond jaune
et les elements ayant comme classe
“principes“ auront une couleur rouge
et une police ‘gras’ */
```



Sélecteurs CSS

Sélecteurs de classe

Le sélecteur `.class1.class2` est un sélecteur qui sélectionne tous les éléments avec à la fois class1 et class2 définis dans leur attribut de classe

Exemple

```
<body>
    <h1>page personnelle</h1>
    <div class="présentation">
        <h1>Mark Zuckerberg </h1>
        <p>créateur de Facebook</p>
    </div>

    <p class="premier principe">La clé dans tout ce que l'on fait, c'est de prioriser. C'est-à-dire de cerner les points importants à traiter pour nous, et de s'y tenir</p>
    <p class="principe premier">Une règle simple pour vraiment changer les choses, c'est de commencer toujours par le plus simple, et non par le plus difficile</p>
    <p class="principe">En faire toujours plus que ce que vous devez</p>
</body>
```

page personnelle

Mark Zuckerberg

créateur de Facebook

La clé dans tout ce que l'on fait, c'est de prioriser. C'est-à-dire de cerner les points importants à traiter pour nous, et de s'y tenir

Une règle simple pour vraiment changer les choses, c'est de commencer toujours par le plus simple, et non par le plus difficile

En faire toujours plus que ce que vous devez

Contenu du Ficher style.css

```
.principe.premier
{
    color:red;
    background-color: yellow;
}
/*Les éléments ayant à la fois le terme "principe" et le terme "premier" auront un fond jaune et une couleur rouge*/
```

Sélecteurs CSS

Sélecteurs de classe

Le sélecteur `.class1 .class2` est un sélecteur qui sélectionne chaque élément avec l'attribut `class=« class2 »` qui est un descendant d'un élément avec l'attribut `class = « class1 »`

Exemple

```
<body>
<h1>Foot Européen</h1>
<div class = "championnats">
    <h1> Championnats </h1>
    <p class="angleterre">Premier league</p>
    <p class="allemande">Bundesliga</p>
    <p class="espagne">Laliga</p>
</div>
<div class="equipes">
    <h1> Equipes </h1>
    <p class="angleterre">Barnsley, Birmingham City,
        Manchester City, Manchester united</p>
    <p class="allemande">Bayer Leverkusen, Bayern Munich,
        Borussia Dortmund</p>
    <p class="espagne">Athletic Bilbao, Atlético Madrid, Real
        Madrid, FC Barcelone</p>
</div>
</body>
```



Contenu du Ficher
style.css

```
.championnats .angleterre
{
    color:red;
    background-color: yellow;
}

.equipes .angleterre
{
    color:white;
    background-color: green;
}

/*Les éléments ayant comme classe
angleterre et qui sont des descendants
d'un élément de classe "championnats"
auront un fond jaune et une couleur
rouge,
```

Les éléments ayant comme classe
angleterre et qui sont des descendants
d'un élément de classe "equipes" auront
un fond vert et une couleur blanche*/

Sélecteurs CSS

Sélecteurs #id

Le selecteur **#id** est un sélecteur qui permet de cibler l'élément HTML ayant comme id « id ». Un id concerne un seul élément de la page

Exemple

```
<!DOCTYPE html>
<html>
<head>
<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Le Site Web</title>
</head>
<body>
    <h1>Introduction</h1>
    <p id="intro">an efficient flight to gate assignment must ensure
        the safety of the activities around the parked aircrafts. Knowing
        that the use of aerobridges is the best solution to minimize the
        risks associated with operations around aircrafts</p>
    <h1>Literature review</h1>
    <p id="literature"> Since the basic gate assignment problem
        (GAP) is a quadratic assignment problem and was shown to be
        NP-hard by Obata [1] as mentioned in [2], different solving
        approaches were developed and can be classified to three main
        categories.</p>
</body>
</html>
```

Contenu du Ficher style.css

```
#intro
{
    color:red;
    font-weight: bold;
}

#literature
{
    color:magenta;
}

/*L'élément ayant comme id "intro"
aura une couleur rouge et une police
'gras' et l'élément ayant comme id
"literature" aura une couleur magenta*/
```

Sélecteurs CSS

Sélecteurs #id

Exemple

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Le Site Web</title>
</head>
<body>
    <h1>Introduction</h1>
    <p id="intro">an efficient flight to gate assignment must ensure
        the safety of the activities around the parked aircrafts. Knowing
        that the use of aerobridges is the best solution to minimize the
        risks associated with operations around aircrafts</p>
    <h1>Literature review</h1>
    <p id="literature"> Since the basic gate assignment problem
        (GAP) is a quadratic assignment problem and was shown to be
        NP-hard by Obata [1] as mentioned in [2], different solving
        approaches were developed and can be classified to three main
        categories. </p>
</body>
</html>
```

Contenu du Ficher style.css

```
#intro
{
    color:red;
    font-weight: bold;
}

#literature
{
    color:magenta;
}

/*L'élément ayant comme id "intro"
aura une couleur rouge et une police 'gras' et l'élément ayant comme id
"literature" aura une couleur magenta*/

```

Introduction

an efficient flight to gate assignment must ensure the safety of the activities around the parked aircrafts. Knowing that the use of aerobridges is the best solution to minimize the risks associated with operations around aircrafts

Literature review

Since the basic gate assignment problem (GAP) is a quadratic assignment problem and was shown to be NP-hard by Obata [1] as mentioned in [2], different solving approaches were developed and can be classified to three main categories.

Sélecteurs CSS

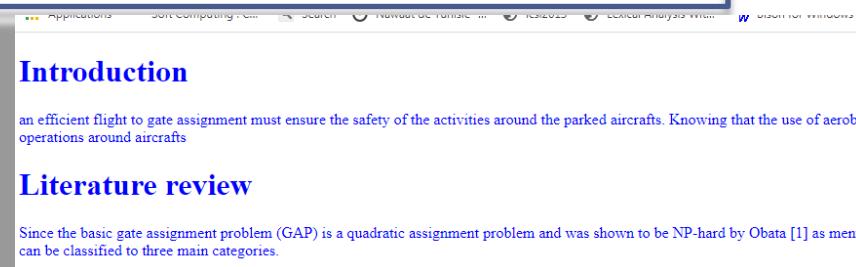
Sélecteur * sélectionne tous les éléments

Exemple

```
<!DOCTYPE html>
<html>
<head>
<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Le Site Web</title>
</head>
<body>
    <h1>Introduction</h1>
    <p id="intro">an efficient flight to gate assignment must ensure
        the safety of the activities around the parked aircrafts. Knowing
        that the use of aerobridges is the best solution to minimize the
        risks associated with operations around aircrafts</p>
    <h1>Literature review</h1>
    <p id="literature"> Since the basic gate assignment problem
        (GAP) is a quadratic assignment problem and was shown to be
        NP-hard by Obata [1] as mentioned in [2], different solving
        approaches were developed and can be classified to three main
        categories.</p>
</body>
</html>
```

Contenu du Ficher style.css

```
*
{
    color:blue;
}
/*tous les éléments auront une
couleur bleu*/
```



Sélecteurs CSS

Sélecteurs **element** par exemple le sélecteur **p** sélectionne tous les éléments **<p>**

Exemple

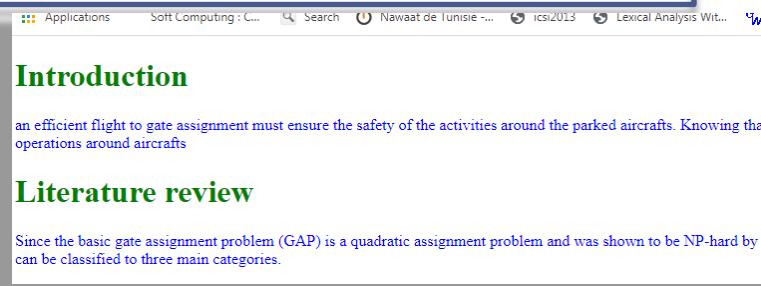
```
<!DOCTYPE html>
<html>
<head>
<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Le Site Web</title>
</head>
<body>
    <h1>Introduction</h1>
    <p id="intro">an efficient flight to gate assignment must ensure
        the safety of the activities around the parked aircrafts. Knowing
        that the use of aerobridges is the best solution to minimize the
        risks associated with operations around aircrafts</p>
    <h1>Literature review</h1>
    <p id="literature"> Since the basic gate assignment problem
        (GAP) is a quadratic assignment problem and was shown to be
        NP-hard by Obata [1] as mentioned in [2], different solving
        approaches were developed and can be classified to three main
        categories.</p>
</body>
</html>
```

Contenu du Ficher style.css

```
p
{
    color:blue;
}

h1
{
    color:green;
}

/*tous les paragraphes auront la
couleur bleu et les titres de niveau1
auront la couleur verte*/
```



Sélecteurs CSS

Sélecteurs **element.class** sélectionne tous les **<element class="class">**

Exemple

```
<body>
<h1>Foot Européen</h1>
<div class = "championnats">
    <h1 class= " champ " > Championnats </h1>
    <p class="angleterre">Premier league</p>
    <p class="allemande">Bundesliga</p>
    <p class="espagnie">Laliga</p>
</div>
<div class="equipes">
    <h1 class= " equip " > Equipes </h1>
    <p class="angleterre">Barnsley, Birmingham City,
        Manchester City, Manchester united</p>
    <p class="allemande">Bayer Leverkusen, Bayern Munich,
        Borussia Dortmund</p>
    <p class="espagnie">Athletic Bilbao, Atlético Madrid,
        Real Madrid, FC Barcelone</p>
</div>
</body>
```

Foot Européen

Championnats

Premier league

Bundesliga

Laliga

Equipes

Barnsley, Birmingham City, Manchester City, Manchester united

Bayer Leverkusen, Bayern Munich, Borussia Dortmund

Athletic Bilbao, Atlético Madrid, Real Madrid, FC Barcelone

Contenu du Ficher style.css

```
p.allemagne
{
    color:red;
}
p.angleterre
{
    color:blue;
}
p.espagnie {
    color:green;
}
h1.champ
{
background-color:green;
}
h1.equip
{
background-color:yellow;
}
/* tous les paragraphes le la classe
“allenagne” auront la couleur rouge,
tous les paragraphes de la classe
“angleterre” auront la couleur bleu,
tous les paragraphes de la classe
“espagnie” auront la couleur verte.
Les les tires h1 de la classe “champ”
auront un fond vert et tous les titres
h1 de la classe “equip” auront un fond
jaune. */
```

Sélecteurs CSS

Sélecteurs **element1,element2** sélectionne tous les **<element1>** et tous les **<element2>**

Exemple

```
<body>
```

```
<h1>Foot Européen</h1>
```

```
<h2> Championnats </h2>
```

```
<ul>
```

- Premier league
- Bundesliga
- Laliga

```
</ul>
```

```
<h2> Classement Equipes européennes </h2>
```

```
<ol>
```

- FC Barcelone
- Real Madrid
- Bayern Munich
- Manchester City

```
</ol>
```

```
</body>
```

Foot Européen

Championnats

- Premier league
- Bundesliga
- Laliga

Classement Equipes européennes

1. FC Barcelone
2. Real Madrid
3. Bayern Munich
4. Manchester City

Contenu du Ficher style.css

```
ul,ol
```

```
{
```

```
color:red;
```

```
font-weight: bold;
```

```
}
```

```
/* toutes les listes non ordonnées et toutes les listes ordonnées auront la couleur rouge et la police "gras" */
```

Sélecteurs CSS

Sélecteurs **element1 element2** sélectionne tous les **<element2>** qui sont à l'intérieur de **<element1>**

Exemple

```
<div>
  <ul>
    <li> championnats européens
      <ol>
        <li>Premier league</li>
        <li>Bundesliga</li>
        <li>Laliga</li>
      </ol>
    </li>
    <li> championnats africains
      <ol>
        <li>championnat de tunisie</li>
        <li>championnat camrounais</li>
        <li>championnat égyptien</li>
      </ol>
    </li>
  </ul>
</div>
<h2> classement des Equipes européennes </h2>
<ol>
  <li>FC Barcelone </li>
  <li>Real Madrid </li>
  <li>Bayern Munich </li>
  <li>Manchester City </li>
</ol>
</body>
```

Contenu du Ficher style.css

```
div ol
{
  color:red;
  font-weight: bold;
}

/* toutes les listes ordonnées <ol>
qui se trouvent à l'intérieur d'un <div>
(pas forcement un descendant direct
de div) auront une couleur rouge et
une police "gras" */
```

Foot Mondial

- championnats européens
 1. Premier league
 2. Bundesliga
 3. Laliga
- championnats africains
 1. championnat de tunisie
 2. championnat camrounais
 3. championnat égyptien

classement des Equipes européennes

1. FC Barcelone
2. Real Madrid
3. Bayern Munich
4. Manchester City

Sélecteurs CSS

Sélecteurs **element1>element2** sélectionne tous les **<element2>** dont le parent est **<element1>**

Exemple

```
<body>
<h1>Foot Mondial</h1>
<div>
  <ul>
    <li> championnats européens
      <ol>
        <li>Premier league</li>
        <li>Bundesliga</li>
        <li>Laliga</li>
      </ol>
    </li>
    <li> championnats africains
      <ol>
        <li>championnat de tunisie</li>
        <li>championnat camrounais</li>
        <li>championnat égyptien</li>
      </ol>
    </li>
  </ul>
</div>
<div>
  <h2> classement des Equipes européennes </h2>
  <ol>
    <li>FC Barcelone </li>
    <li>Real Madrid </li>
    <li>Bayern Munich </li>
    <li>Manchester City </li>
  </ol>
</div>
</body>
```

Contenu du Ficher style.css

```
div>ol
{
  color:red;
  font-weight: bold;
}

/* toutes les listes ordonnées <ol>
don't le parent est un <div>
(descendant direct de div) auront une
couleur rouge et une police "gras" */
```

Foot Mondial

- championnats européens
 1. Premier league
 2. Bundesliga
 3. Laliga
- championnats africains
 1. championnat de tunisie
 2. championnat camrounais
 3. championnat égyptien

classement des Equipes européennes

1. FC Barcelone
2. Real Madrid
3. Bayern Munich
4. Manchester City

Sélecteurs CSS

Sélecteurs **element1+element2** sélectionne tous les **<element2>** qui sont placés immédiatement après **<element1>**

Exemple

```
<body>
  <h1>Foot Mondial</h1>
  <p>La Ligue des champions de l'UEFA, parfois abrégée en C1 et
      anciennement dénommée Coupe des clubs champions européens,
      est une compétition annuelle de football organisée par l'Union des
      associations européennes de football</p>
  <div>
    <h2> classement des Equipes européennes </h2>
    <ol>
      <li>FC Barcelone </li>
      <li>Real Madrid </li>
      <li>Bayern Munich </li>
      <li>Manchester City </li>
    </ol>
  </div>
  <p> Le Bayern Munich est sans conteste le club de football allemand le
      plus titré et il dirige la Bundesliga. L'équipe qui se targue de
      compter des stars dynamiques et énergiques </p>
</body>
```

Contenu du Ficher style.css

```
div+p
{
  color:red;
  background-color: yellow;
}

/* tous les paragraphes qui sont
   placés immédiatement après un <div>
   auront une couleur rouge et un fond
   jaune */
```

Foot Mondial

La Ligue des champions de l'UEFA, parfois abrégée en C1 et anciennement dénommée Coupe des clubs champions européennes de football

classement des Equipes européennes

1. FC Barcelone
2. Real Madrid
3. Bayern Munich
4. Manchester City

Le Bayern Munich est sans conteste le club de football allemand le plus titré et il dirige la Bundesliga

Sélecteurs CSS

Sélecteurs **element1~element2** sélectionne tous les **<element2>** qui sont précédés par **<element1>** avec element1 et element2 ayant le même parent

Exemple

```
<body>
  <div>Liste des boissons</div>
  <ul>
    <li>café</li>
    <li>thé</li>
    <li>lait</li>
  </ul>

  <p>liste des gateaux</p>
  <ul>
    <li>opera</li>
    <li>mille-feuilles</li>
    <li>cake</li>
  </ul>

  <h2>Liste des sandwichs</h2>
  <ul>
    <li>Panini</li>
    <li>Hamburger</li>
    <li>croque-Monsieur</li>
  </ul>
</body>
```

Liste des boissons

- café
- thé
- lait

liste des gateaux

- opera
- mille-feuilles
- cake

Liste des sandwichs

- Panini
- Hamburger
- croque-Monsieur

Contenu du Ficher style.css

```
p~ul
{
  color:red;
  background-color:yellow;
}

/*(~tilde) toutes les listes non
ordonnées qui sont précédées par un
paragraphe .Avec ul et p ayant le
même parent auront une couleur
rouge et un fond jaune */
```

Sélecteurs CSS

Les Sélecteurs D'attributs

Les sélecteurs d'attribut permettent de cibler un élément selon la présence d'un **attribut** ou selon la valeur donnée d'un **attribut**.

	Exemple	Explication
[attribut]	[src]	Cible tous les éléments ayant src comme attribut
[attribut=valeur]	[title="bugatti"]	Cible tous les éléments ayant un attribut title ayant la valeur "bugatti"
[attribut~=valeur]	[title~=foot]	Cible tous les éléments avec un attribut title contenant le mot "foot"
[attribut =valeur]	[lang=en]	Cible les éléments avec un attribut lang dont la valeur peut être exactement "en" ou peut commencer par "en" immédiatement suivie d'un trait d'union.
[attribut^=valeur]	a[href^="https"]	Cible chaque élément <a> dont la valeur de l'attribut href commence par "https"
[attribut\$=valeur]	a[href\$=".pdf"]	Cible chaque élément <a> dont la valeur de l'attribut href se termine par ".pdf"
[attribut*=valeur]	a[href*="tunisie"]	Cible chaque élément <a> dont la valeur de l'attribut href contient la sous-chaine "tunisie"

Sélecteurs CSS

Les Sélecteurs D'attributs

Remarque

The asterisk attribute selector `*=` matches any substring occurrence. You can think of it as a string contains call.

Input	Matches <code>*=bar</code>
foo	No
foobar	Yes
foo bar	Yes
foo barbaz	Yes
foo bar baz	Yes

The tilde attribute selector `~=` matches whole words only.

Input	Matches <code>~=bar</code>
foo	No
foobar	No
foo bar	Yes
foo barbaz	No
foo bar baz	Yes

Sélecteurs CSS

Les Pseudo-éléments

Un Pseudo-élément CSS est utilisé pour styler des parties spécifiées d'un élément. Par exemple, il peut être utilisé pour:

- Styler la première lettre, ou ligne, d'un élément
- Insérer du contenu avant ou après le contenu d'un élément

Syntaxe

```
sélecteur::pseudo-élément {  
    propriété: valeur;  
}
```

Sélecteurs CSS

Les Pseudo-éléments

Le pseudo-élément `::first-line` est utilisé pour ajouter un style spécial à la première ligne d'un texte. L'exemple suivant met en forme la première ligne du texte dans tous les éléments `<p>`:

```
<body>
  <h1>Introduction</h1>
  <p>an efficient flight to gate assignment must ensure
    the safety of the activities around the parked aircrafts. Knowing
    that the use of aerobridges is the best solution to minimize the
    risks associated with operations around aircrafts</p>
  <h1>Literature review</h1>
  <p> Since the basic gate assignment problem
    (GAP) is a quadratic assignment problem and was shown to be
    NP-hard by Obata [1] as mentioned in [2], different solving
    approaches were developed and can be classified to three main
    categories. </p>
</body>
```

Introduction

an efficient flight to gate assignment must ensure the safety of the activities around the parked aircrafts. Knowing
operations around aircrafts

Literature review

Since the basic gate assignment problem (GAP) is a quadratic assignment problem and was shown to be NP-hard,
and can be classified to three main categories.

Contenu du Ficher style.css

```
p::first-line {
  color: red;
  font-style: italic;
}

/* toutes les premières lignes des
paragraphes auront une couleur rouge
et un style de police italique */
```

Sélecteurs CSS

Les Pseudo-éléments

Le pseudo-élément `::first-line`

Remarques

- Le pseudo-élément `::first-line` ne peut être appliqué qu'aux éléments de type bloc (`<div>`, `<h1>`... `<h6>`, `<p>`, ``, ``, `<dl>`, ``, `<dt>`, `<dd>` `<table>`, `<form>`..)
- Les propriétés suivantes s'appliquent au pseudo-élément `::first-line`:
 - `font properties`
 - `color properties`
 - `background properties`
 - `word-spacing`
 - `letter-spacing`
 - `text-decoration`
 - `vertical-align`
 - `text-transform`
 - `line-height`
 - `clear`

Sélecteurs CSS

Les Pseudo-éléments

Le pseudo-élément `::first-letter` est utilisé pour ajouter un style spécial à la première lettre d'un texte. L'exemple suivant met en forme la première lettre du texte dans tous les éléments `<p>`:

```
<body>
  <h1>Introduction</h1>
  <p>An efficient flight to gate assignment must ensure
    the safety of the activities around the parked aircrafts. Knowing
    that the use of aerobridges is the best solution to minimize the
    risks associated with operations around aircrafts</p>
  <h1>Literature review</h1>
  <p> Since the basic gate assignment problem
    (GAP) is a quadratic assignment problem and was shown to be
    NP-hard by Obata [1] as mentioned in [2], different solving
    approaches were developed and can be classified to three main
    categories. </p>
</body>
```

Contenu du Ficher style.css

```
p::first-letter {
  color:#ff0000;
  font-size: xx-large;
}

/* toutes les premières lettres des
paragraphes auront une couleur rouge
et un style de police extra-large */
```

Introduction

An efficient flight to gate assignment must ensure the safety of the activities around the parked with operations around aircrafts

Literature review

Since the basic gate assignment problem (GAP) is a quadratic assignment problem and was shc and can be classified to three main categories.

Sélecteurs CSS

Les Pseudo-éléments

Plusieurs pseudo-éléments peuvent être combinés:

```
<body>
<h1>Introduction</h1>
<p>An efficient flight to gate assignment must ensure
    the safety of the activities around the parked aircrafts. Knowing
    that the use of aerobridges is the best solution to minimize the
    risks associated with operations around aircrafts</p>
<h1>Literature review</h1>
<p> Since the basic gate assignment problem
    (GAP) is a quadratic assignment problem and was shown to be
    NP-hard by Obata [1] as mentioned in [2], different solving
    approaches were developed and can be classified to three main
    categories. </p>
</body>
```

Introduction

AN EFFICIENT FLIGHT TO GATE ASSIGNMENT MUST ENSURE THE SAFETY OF THE ACTIVITIES AROUND THE PARKED AIRCRAFTS associated with operations around aircrafts

Literature review

SINCE THE BASIC GATE ASSIGNMENT PROBLEM (GAP) IS A QUADRATIC ASSIGNMENT PROBLEM AND WAS SHOWN TO BE NP-HARD, DIFFERENT SOLVING APPROACHES WERE DEVELOPED AND CAN BE CLASSIFIED TO THREE MAIN CATEGORIES.

Contenu du Ficher style.css

```
p::first-letter {
    color: red;
    font-size: xx-large;
}
p::first-line {
    color: blue;
    font-variant: small-caps;
}
/* la première lettre des paragraphes sera rouge, dans une taille de police xx-large. Le reste de la première ligne sera bleu et en petites capitales. Le reste du paragraphe aura la taille et la couleur de la police par défaut: */
```

Sélecteurs CSS

Les Pseudo-éléments

Le Pseudo-élément `::before` peut être utilisé pour insérer du contenu avant le contenu d'un élément.

Le Pseudo-élément `::after` peut être utilisé pour insérer du contenu après le contenu d'un élément.

Exemple

```
<body>
  <h1>Introduction</h1>
  <p>An efficient flight to gate assignment must ensure
    the safety of the activities around the parked aircrafts. Knowing
    that the use of aerobridges is the best solution to minimize the
    risks associated with operations around aircrafts</p>
  <h1>Literature review</h1>
  <p> Since the basic gate assignment problem
    (GAP) is a quadratic assignment problem and was shown to be
    NP-hard by Obata [1] as mentioned in [2], different solving
    approaches were developed and can be classified to three main
    categories. </p>
</body>
```



Introduction

An efficient flight to gate assignment must ensure the safety of the activities around the parked aircrafts. Knowing that the use of aerobridges is the best solution to minimize the risks associated with operations around aircrafts



Literature review

Since the basic gate assignment problem (GAP) is a quadratic assignment problem and was shown to be NP-hard by Obata [1] as mentioned in [2], different solving approaches were developed and can be classified to three main categories



Contenu du Ficher style.css

```
h1::before {
  content: url(avion.jpg);
}

p::after {
  content: url(smiley.jpg);
}

/*insère une image d'un avion, avant le
contenu de chaque titre et insère un
smiley après chaque paragraphe */
```

Sélecteurs CSS

Les Pseudo-éléments

Le Pseudo-élément `::selection` cible la partie d'un élément qui est sélectionnée par un utilisateur.

Exemple

```
<body>
  <h1>Introduction</h1>
  <p>An efficient flight to gate assignment must ensure
    the safety of the activities around the parked aircrafts. Knowing
    that the use of aerobridges is the best solution to minimize the
    risks associated with operations around aircrafts</p>
  <h1>Literature review</h1>
  <p> Since the basic gate assignment problem
    (GAP) is a quadratic assignment problem and was shown to be
    NP-hard by Obata [1] as mentioned in [2], different solving
    approaches were developed and can be classified to three main
    categories. </p>
</body>
```

Introduction

An efficient flight to gate assignment must ensure the safety of the activities around the parked aircrafts.
with operations around aircrafts

Literature review

Since the basic gate assignment problem (GAP) is a quadratic assignment problem and was shown to be
can be classified to three main categories.

Contenu du Ficher style.css

```
::-moz-selection { /* Code for Firefox */
  color: red;
  background: yellow;
}

::selection {
  color: red;
  background: yellow;
}
```

/*la partie sélectionnée par l'utilisateur
aura une couleur rouge et un fond jaune*/

Sélecteurs CSS

Les Pseudo Classes

Une pseudo-classe est utilisée pour définir un état spécial d'un élément.

Par exemple, il peut être utilisé pour:

- Styliser un élément lorsqu'un utilisateur passe la souris dessus
- Styliser de manières différentes des liens visités et non visités
- Styliser un élément lorsqu'il obtient le focus
- ...

Syntaxe

La syntaxe d'une pseudo-classe:

```
selecteur:pseudo-class {  
    propriété: valeur;  
}
```

Sélecteurs CSS

Les Pseudo-classes

Le Pseudo-classe **a:link** permet de styler un lien non visité, **a:visited** cible un lien visité, **a:hover** cible un lien survolé et **a:active** cible un lien sélectionné.

Remarque: **a: hover** DOIT venir après **a: link** et **a: visited** dans la définition CSS pour être efficace. **a: active** DOIT venir après un **a:hover** dans la définition CSS pour être efficace.

Exemple

```
<body>
<h1>Foot Européen</h1>

<h2> Championnats </h2>
<ul>
    <li><a href="https://www.premierleague.com/"> Premier league </a></li>
    <li><a href="https://www.bundesliga.com/"> Bundesliga </a></li>
    <li><a href="https://www.laliga.com/en-GB"> Laliga </a></li>
    <li><a href="http://www.legaseriea.it/en"> Legaseriea </a></li>
</ul>
</body>
```

Foot Européen

Championnats

- [Premier league](https://www.premierleague.com/)
- [Bundesliga](https://www.bundesliga.com/)
- [Laliga](https://www.laliga.com/en-GB)
- [Legaseriea](http://www.legaseriea.it/en)

Foot Européen

Championnats

- [Premier league](https://www.premierleague.com/)
- [Bundesliga](https://www.bundesliga.com/)
- [Laliga](https://www.laliga.com/en-GB)
- [Legaseriea](http://www.legaseriea.it/en)

Contenu du Ficher style.css

```
/* lien non visité */
a:link { color: red; }

/* lien visité */
a:visited { color: green; }

/* souris survolant le lien */
a:hover { color: magenta; }

/* lien sélectionné */
a:active { color: cyan; }

/*les liens non visités auront la couleur
rouge ceux visités auront la couleur verte,
ceux survolés par la souris auront la
couleur magenta et ceux activés (en
cliquant dessus sans relâcher) auront la
couleur cyan*/
```

Sélecteurs CSS

toutes les Pseudo-Classes

Sélecteur	Exemple	Description de l'exemple
:active	a:active	cible le lien actif
:hover	a:hover	cible les liens survolés par la souris
:link	a:link	cible les lien non visités
:visited	a:visited	cible les lien visités

Sélecteur	Exemple	Description de l'exemple
:not(selector)	:not(p)	cible chaque élément qui n'est pas un élément <p>
:root	:root	Cible la racine du document
:target	#news:target	Cible l'élément actif ayant id=news (quand on clique sur un url contenant cet ancre)

Sélecteurs CSS toutes les Pseudo-Classes

Exemple :target

```
<body>
<h1>Ceci est un titre</h1>
<p><a href="#news1">aller à nouveau contenu 1</a></p>
<p><a href="#news2">aller à nouveau contenu 2</a></p>

<p id="news1"><b>nouveau contenu 1...</b></p>
<p id="news2"><b>nouveau contenu 2...</b></p>
</body>
```

```
/* contenu du fichier CSS */
:target {
    border: 2px solid #D4D4D4;
    background-color:#e5eecc;
}
```

Cible l'élément actif (quand on clique sur un url contenant son ancre)

Ceci est un titre

[aller à nouveau contenu 1](#)

[aller à nouveau contenu 2](#)

nouveau contenu 1...

nouveau contenu 2...

Sélecteurs CSS

toutes les Pseudo-Classes

Sélecteur	Exemple	Description de l'exemple
:checked	input:checked	Cible chaque checked <input>
:disabled	input:disabled	Cible chaque disabled <input>
:enabled	input:enabled	Cible chaque enabled <input>
:focus	input:focus	Cible chaque <input> qui a le focus
:in-range	input:in-range	Cible chaque <input> ayant une valeur comprise dans un intervalle spécifié.
:invalid	input:invalid	Cible chaque <input> avec une valeur non valide
:optional	input:optional	Cible chaque <input> qui n'a pas un attribut "required"
:out-of-range	input:out-of-range	Cible chaque <input> avec une valeur en dehors de l'intervalle spécifié
:read-only	input:read-only	Cible chaque <input> avec un attribut "readonly"
:read-write	input:read-write	Cible chaque <input> qui n'a pas un attribut "readonly"
:required	input:required	Cible chaque <input> ayant un attribut "required"
:valid	input:valid	Cible chaque <input> avec une valeur valide

Sélecteurs CSS

toutes les Pseudo-Classes

Sélecteur	Exemple	Description de l'exemple
:empty	p:empty	Cible chaque <p> vide
:first-child	p:first-child	Cible chaque <p> qui est le premier fils de son père
:first-of-type	p:first-of-type	Cible chaque <p> qui est le premier fils de type <p> de son père
:lang(language)	p:lang(it)	Cible chaque <p> avec un attribut lang ayant une valeur commençant par "it"
:last-child	p:last-child	Cible chaque <p> qui est le dernier fils de son père
:last-of-type	p:last-of-type	Cible chaque <p> qui est le dernier fils de type <p> de son père
:nth-child(n)	p:nth-child(2)	Cible chaque <p> qui est le deuxième fils de son père.
:nth-last-child(n)	p:nth-last-child(2)	Cible chaque <p> qui est le deuxième fils de son père , en commençant à compter à partir de la fin
:nth-last-of-type(n)	p:nth-last-of-type(2)	Cible chaque <p> qui est le second fils de type <p> de son père , en commençant à compter à partir de la fin
:nth-of-type(n)	p:nth-of-type(2)	Cible chaque <p> qui est le second fils de type<p> de son père
:only-of-type	p:only-of-type	Cible chaque <p> qui est le seul fils de type <p> de son père
:only-child	p:only-child	Cible chaque <p> qui est le seul fils de son père

Sélecteurs CSS

Les Pseudo-classes

:first-child vs **:first-of-type**

Exemple

```
<body>
  <p>Ce paragraphe est le premier fils de son père (body)</p>
  <p>Ce paragraphe est le deuxième fils de son père (body)</p>
  <h1>Titre 1</h1>
  <p>ce paragraphe est le troisième fils de type "p" de son père (body)</p>
  <div>
    <h1>titre 2 </h1>
    <a href="https://www.premierleague.com/"> Premier league </a>
    <p>Ce paragraphe est le premier fils de type "p" de son père (div).</p>
    <p>ce paragraphe est le dernier fils de son père</p>
  </div>
</body>
```

Ce paragraphe est le premier fils de son père (body)

Ce paragraphe est le deuxième fils de son père (body)

Titre 1

ce paragraphe est le troisième fils de type "p" de son père (body)

titre 2

[Premier league](https://www.premierleague.com/)

Ce paragraphe est le premier fils de type "p" de son père (div).

ce paragraphe est le dernier fils de son père

Contenu du Ficher style.css

```
p:first-child {
  background-color: yellow;
}
```

```
p:last-child {
  color:white;
  background-color: black;
}
```

```
p:nth-child(2)
{
  color:green;
  background-color: red;
}
```

Sélecteurs CSS

Les Pseudo-classes

:first-child vs :first-of-type

Exemple

```
<body>
  <h1>Titre 1</h1>
  <p>Ce paragraphe est le premier fils de type p de son père (body)</p>
  <p>ce paragraphe est le deuxième fils de type "p" de son père (body)</p>
  <div>
    <h1>titre3 </h1>
    <p> Ce paragraphe est le premier fils de type "p" de son père (div).</p>
    <a href="https://www.bundesliga.com/">Bundesliga</a>
    <p>Ce paragraphe est le deuxième fils de type "p" de son père (div) </p>
    <a href="https://www.premierleague.com/"> Premier league </a>
    <p>Ce paragraphe est le dernier fils de type "p" de son père (div) </p>
    <a href="https://www.laliga.com/en-GB">Laliga</a>
  </div>
  <p>ce paragraphe est le dernier fils de type "p" de son père (body)</p>
  <a href="http://www.legaseriea.it/en">legaseriea</a>
</body>
```

Titre 1

Ce paragraphe est le premier fils de type p de son père (body)

ce paragraphe est le deuxième fils de type "p" de son père (body)

titre3

Ce paragraphe est le premier fils de type "p" de son père (div).

Bundesliga

Ce paragraphe est le deuxième fils de type "p" de son père (div)

Premier league

Ce paragraphe est le dernier fils de type "p" de son père (div)

Laliga

ce paragraphe est le dernier fils de type "p" de son père (body)

legaseriea

Contenu du Ficher
style.css

```
p:first-of-type
{
  color:white;
  background-color: green;
}
p:nth-of-type(2)
{
  color:yellow;
  background-color:black;
}
p:last-of-type
{
  color:green;
  background-color: yellow;
}
```

Sélecteurs CSS

Un jeux pour maitriser les sélecteurs
<https://flukeout.github.io/>

Les propriétés basiques CSS

- **la propriété font-style:** spécifie le style de police d'un texte

Valeur	description
normal	Le navigateur affiche le style de police normal. (style par défaut)
italic	Le navigateur affiche le style de police italique.
oblique	Le navigateur affiche le style de police oblique.
initial	Réinitialise la propriété à sa valeur par défaut
inherit	Hérite cette propriété de son élément parent.

- **Remarque:** La forme **italique** est généralement une forme cursive qui utilise moins d'espace horizontal que les autres formes classiques. La forme **oblique** quant à elle est simplement une version penchée de la forme normale.

This should be the Regular face

This should be the Italic face

This should be the Oblique face

Les propriétés basiques CSS

- la propriété **font-style**: Exemple

```
<body>
<div>
    ceci est un <span>span element</span> avec une police italique comme c'est
    défini pour les éléments span.
</div>

<div class="extra">
    ceci est un <span>span element</span> avec une police normale, car il l'hérite
    de son père (div).
</div>

<div>
    ceci est un <span>span element</span> avec une police italique comme c'est
    défini pour les éléments span.
</div>
</body>
</head>
```

ceci est un span element avec une police italique comme c'est défini pour les éléments span.
ceci est un span element avec une police normale, car il l'hérite de son père (div).
ceci est un span element avec une police italique comme c'est défini pour les éléments span.

Contenu du Ficher style.css

```
div
{
    font-style: normal;
}

span {
    font-style: italic;
    border: 1px solid black;
}

.extra span {
    font-style: inherit;
    border: 1px solid black;
    color:red;
}
```

/* Le span ayant comme père un
element de classe "extra" hérite
le style de la police de son père.
*/

Les propriétés basiques CSS

- la propriété **font-weight**: définit l'épaisseur des caractères (gras ou fins) du texte

Valeur	description
normal	Définit des caractères normales (valeur par défaut)
bold	Définit des caractères gras
bolder	Définit des caractères plus gras
lighter	Définit des caractères plus fines
initial	Réinitialise la propriété à sa valeur par défaut
inherit	Hérite cette propriété de son élément parent.

Les propriétés basiques CSS

- **la propriété text-decoration:** spécifie la décoration ajoutée au texte et est une propriété abrégée pour:
 - text-decoration-line (obligatoire)
 - text-decoration-color
 - text-decoration-style

Valeur	description
text-decoration-line	Définit le type de décoration de texte à utiliser(underline, overline, line-through, none)
text-decoration-color	Définit la couleur de la décoration du texte
text-decoration-style	Définit le style de la décoration du texte (solid, wavy, dotted, dashed, double)

Les propriétés basiques CSS

- la propriété **text-decoration**: spécifie la décoration ajoutée au texte

```
<body>
    <h1>Ceci est un titre 1</h1>
    <h2>Ceci est un titre 2</h2>
    <h3>Ceci est un titre 3</h3>
</body>
</html>
```

Ceci est un titre 1

~~Ceci est un titre 2~~

Ceci est un titre 3

Contenu du Ficher style.css

```
h1 {
    text-decoration: underline overline dotted blue;
}

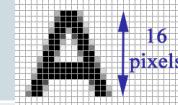
h2 {
    text-decoration: line-through wavy red;
}

h3 {
    text-decoration: underline dashed magenta;
}
```

Les propriétés basiques CSS

- **Les Unités CSS :** CSS a plusieurs unités pour exprimer une Taille. De nombreuses propriétés CSS prennent des valeurs de Taille, telles que **font-size**, **width**, **margin**, **padding**, etc..
- Il existe deux types d'unités de Taille: **absolue** et **relative**

Taille absolue

unité	description
px nombre	nombre de pixels 
cm	centimètres
mm	millimètres
in	inches
pt	point (1 pt = 1/72 in)
pc	Picas (1pc = 12 pt)

Les propriétés basiques CSS

- **Les Unités CSS :** CSS a plusieurs unités pour exprimer une Taille. De nombreuses propriétés CSS prennent des valeurs de Taille, telles que font-size, width, margin, padding, etc..
- Il existe deux types d'unités de Taille: **absolue** et **relative**

Taille relative

unité	description
em	Relative à la taille de la police de l'élément (2em signifie 2 fois la taille de la police actuelle)
rem	L'unité rem définit la taille de la police par rapport à la taille de la police de base du navigateur et n'héritera pas de ses parents.
ch	L'unité ch définit la taille de la police par rapport à la largeur du "0" (zéro)
% nombre	% définit la taille de la police par rapport à l'élément parent
D'autres unités relatives existent: vw,vh,vmin,vmax	

Les propriétés basiques CSS

- **la propriété font-size:** Définit la taille de la police pour différents éléments:

Valeur	description
longueur	Définit la taille de la police sur une taille fixe en px, cm, mm, pt, pc, em, rem, ch etc
%	Définit la taille de la police sur un pourcentage de la taille de la police de l'élément parent
Smaller	Définit la taille de la police sur une taille plus petite que l'élément parent
Larger	Définit la taille de la police sur une taille plus grande que l'élément parent
medium, xx-small, x-small, small, large, x-large xx-large	Définit la taille de la police sur une taille medium, xx-small, x-small, small, large, x-large xx-large size

Les propriétés basiques CSS

- **la propriété font-size:** Définit la taille de la police pour différents éléments:

```
</head>
<body>
    <h1> taille absolue </h1>
    <p id="p1"> ceci est du texte de taille 14px </p>
    <p id="p2"> ceci est du texte de taille 0.8cm </p>
    <p id="p3"> ceci est du texte de taille 10mm </p>
    <p id="p4"> ceci est du texte de taille 14pt </p>
    <p id="p5"> ceci est du texte de taille 2pc </p>

    <h1> taille relative </h1>
    <div id="div1">
        <p> ceci est du texte de taille courante
            <span id="sp1"> ceci est du texte de taille 2em
                (2 fois la taille courante)
            </span>
        </p>
        <p id="p6"> ceci est du texte de taille 2rem </p>
        <p id="p7"> ceci est du texte de taille 150%
            par rapport au parent (div) </p>
    </div>
    <p id="p8"> ceci est du texte de taille x-small </p>
    <p id="p9"> ceci est du texte de taille x-large </p>

</body>
</html>
```

Contenu du Ficher style.css

```
h1{color:red;}
#p1 {font-size:14px;}
#p2 {font-size:0.8cm;}
#p3 {font-size:10mm;}
#p4 {font-size:14pt;}
#p5 {font-size:2pc;}

#div1 {font-size:12px;}
#sp1{font-size:2em;}
#p6 {font-size:2rem;}
#p7 {font-size:150%;}
#p8 {font-size:x-small;}
#p9 {font-size:x-large;}
```

Les propriétés basiques CSS

- la propriété **font-size**: exemple

taille absolue

ceci est du texte de taille 14px

ceci est du texte de taille 0.8cm

ceci est du texte de taille 10mm

ceci est du texte de taille 14pt

ceci est du texte de taille 2pc

taille relative

ceci est du texte de taille courante ceci est du texte de taille 2em (2 fois la taille courante)

ceci est du texte de taille 2rem

ceci est du texte de taille 150% par rapport au parent (div)

ceci est du texte de taille x-small

ceci est du texte de taille x-large

Les propriétés basiques CSS

- **la propriété font-family:** spécifie la police de caractères d'un élément
- La propriété **font-family** peut contenir plusieurs noms de polices. Si le navigateur ne prend pas en charge la première police, il essaie la police suivante.

```
<body>
  <h1>Principes</h1>
  <p >La clé dans tout ce que l'on fait, c'est de prioriser.  
C'est-à-dire de cerner les points importants à traiter pour  
nous, et de s'y tenir
  </p>
  <p >Une règle simple pour vraiment changer les choses,  
c'est de commencer toujours par le plus simple,  
et non par le plus difficile
  </p>
</body>
```

Contenu du Ficher style.css

```
p
{
  font-family: "Times New Roman", Times, serif;
  font-size: 16px;
}

h1
{
  font-family: "Book Antiqua", Helvetica, sans-serif;
}
```

Principes

La clé dans tout ce que l'on fait, c'est de prioriser. C'est-à-dire de cerner les point

Une règle simple pour vraiment changer les choses, c'est de commencer toujours

Les propriétés basiques CSS

- la propriété **@font-face**: permet de définir les polices de caractères à utiliser pour afficher le texte. Cette police peut être chargée depuis un serveur distant ou depuis l'ordinateur de l'utilisateur. Ce qui permet d'être indépendant des polices existantes sur le poste de l'utilisateur
- il existe *plusieurs formats* de fichiers de polices, et ceux-ci ne fonctionnent pas sur tous les navigateurs :
 - .ttf : *TrueType Font*. Fonctionne sur IE9 et tous les autres navigateurs ;
 - .eot : *Embedded OpenType*. Fonctionne sur Internet Explorer uniquement, toutes versions. Ce format est propriétaire, produit par Microsoft ;
 - .otf : *OpenType Font*. Ne fonctionne pas sur Internet Explorer ;
 - .svg : *SVG Font*. format reconnu sur les iPhone et iPad ;
 - .woff : *Web Open Font Format*. Nouveau format conçu pour le Web, qui fonctionne sur IE9 et tous les autres navigateurs.
- En CSS, pour définir une nouvelle police, vous devez la déclarer comme ceci :

```
@font-face {  
    font-family: 'MaNouvellePolice';  
    src: url(sansation_light.woff);  
}
```

Le fichier de police (ici sansation_light.woff) doit être téléchargé et placé dans le même dossier que le fichier CSS (ou dans un sous-dossier, si vous utilisez un chemin relatif).

Les propriétés basiques CSS

- la propriété @font-face:

Exemple:

1. Télécharger les fichiers de police (parfois présente sous différents formats) à partir du site <https://fonts.google.com/> ou <https://www.fontsquirrel.com/>
2. Extraire le fichier compressé dans le dossier /fonts que vous créez au dessous du dossier contenant le fichier CSS

 OFL	25/08/2020 00:00	Document texte	5 Ko
 Syne_Mono	31/10/2020 16:03	Archive WinRAR ZIP	36 Ko
 SyneMono-Regular	25/08/2020 00:00	Fichier de police TrueType	69 Ko

3. On ajoute ce code au fichier style.css

```
@font-face {  
    font-family: 'MaNouvellePolice';  
    src: url(fonts/SyneMono-regular.ttf);  
}
```

4. On pourra donc utiliser cette police en la désignant par son nom dans le fichier CSS

```
P  
{  
    font-family: 'MaNouvellePolice';  
    font-size: 20px;  
}
```

Principes

La clé dans tout ce que l'on fait, c'est de prioriser. C nous, et de s'y tenir

Une règle simple pour vraiment changer les choses, c'est plus difficile

Les propriétés basiques CSS

- la propriété **text-align**: spécifie l'alignement horizontal du texte dans un élément.

Valeur	description
left	Aline le texte à gauche (c'est la valeur par défaut)
right	Aline le texte à droite
center	Centre le texte
justify	Étire les lignes de sorte que chaque ligne ait la même largeur

- Utilisation

```
#p1 {text-align:left;}  
#p2 { text-align:right; }  
#p3{ text-align:center; }  
#p4 { text-align:justify; }
```

alignement left Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

alignement right Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

alignement center Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

alignement justify Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Les propriétés basiques CSS

Ajouter de la couleur et un fond

- Couleur du texte : propriété **color:valeur;**
 - La valeur peut être le nom de la couleur (limité à 16 couleurs)

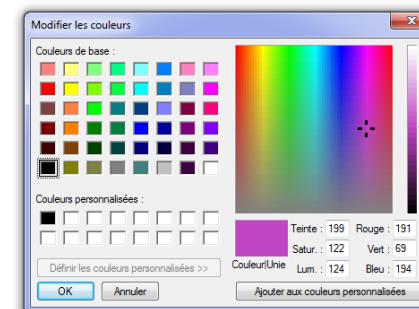
`h1 { color: green; }`

- La valeur peut être en notation hexadécimale

`h1 {color: #FF5A28}` les deux premières lettres désignent la quantité de rouge, le deux secondes la quantité de vert et les deux dernières la quantité de bleu. Voir <https://www.webfx.com/web-design/color-picker/>

- La valeur peut être en **RGB** pour choisir une couleur, on doit définir une quantité de rouge, de vert et de bleu.

`p { color: rgb(240,96,204); }` on peut utiliser le logiciel Paint et la section modifier la couleur pour trouver les coordonnées rgb de la couleur qu'on désire utiliser



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

Les propriétés basiques CSS

Ajouter de la couleur et un fond

- Couleur du fond : propriété **background-color:valeur;** il prend les mêmes valeurs que la propriété color

Exemple

```
/* dans cet exemple on travaille sur la balise body, donc sur toute la page */  
Body { background-color:blue; /* Le fond de la page sera bleu */  
       color:white; /* Le texte de la page sera blanc */ }
```

Remarque toutes les balises contenues dans body vont hériter ces valeurs de couleur de fond et de texte

- On peut changer le fond de n'importe quel élément : titres, paragraphes, certains mots en les marquant avec la balise <mark>

```
<body>  
  <h1>Principes</h1>  
  <p>La clé dans tout ce que l'on fait,<mark> c'est de prioriser</mark>. C'est-à-dire...</p>  
  <p>Une règle simple pour vraiment changer les choses, c'est de commencer ...</p>  
</body>
```

Principes

La clé dans tout ce que l'on fait, **c'est de prioriser**. C'est-à-dire ...

Une règle simple pour vraiment changer les choses, c'est de commencer ...

Contenu du fichier CSS
mark
{ background-color:green;
color:yellow; }

h1 {background-color:magenta;}

Les propriétés basiques CSS

Ajouter de la couleur et un fond

- **Image de fond : propriété background-image:valeur;**
- Comme valeur, on doit renseigner url("nom_de_l_image.png") L'adresse indiquant où se trouve l'image de fond peut être écrite en absolu (http://...) ou en relatif (fond.png)

Exemple

Dans le fichier CSS

body

```
{ background-image: url("nuages.png"); }
```

Attention lorsque vous écrivez une adresse en relatif dans le fichier CSS !

L'adresse de l'image doit être indiquée *par rapport au fichier .css* et non pas par rapport au fichier .html



Les propriétés basiques CSS

Ajouter de la couleur et un fond

- **Image de fond : propriété `background-image:valeur;`**
- **Options disponibles pour l'image de fond**

`background-attachment:fixed;` l'image de fond reste fixe .

`background-attachment:scroll;` l'image de fond défile avec le texte (par défaut)

`background-repeat: repeat-x;` le fond sera répété uniquement sur la première ligne, horizontalement ;

`background-repeat: repeat-y;` le fond sera répété uniquement sur la première colonne, verticalement

`background-repeat: repeat;` le fond sera répété en mosaïque (par défaut)

`background-repeat: no-repeat;` le fond ne sera pas répété. L'image sera donc unique sur la page ;

Les propriétés basiques CSS

Ajouter de la couleur et un fond

- **Image de fond : propriété `background-image:valeur;`**
- **Options disponibles pour l'image de fond**

```
body
{
    background-image: url("ballons.jpg");
    background-repeat: repeat-x;
}
```



Fichier | C:/MOEZ/cours%20WEB1/coursMoez%20CSS/selecteurfont-family.html

Applications Soft Computing : C... Search Nawaat de Tunisie ... icsi2013 Lexical Analysis Wit... Bison fo

Principes

La clé dans tout ce que l'on fait,c'est de prioriser. C'est-à-dire ...

Une règle simple pour vraiment changer les choses, c'est de commencer ...

Fichier | C:/MOEZ/cou

Applications Ouvrir la page d'accueil Search

Principes

La clé dans tout ce que l'on fait,c'est de prioriser

Une règle simple pour vraiment changer les choses

```
body
{
    background-image: url("ballons.jpg");
    background-repeat: repeat-y;
}
```

Applications Soft Computing : C... Search Nawaat de Tunisie ...

Principes

La clé dans tout ce que l'on fait,c'est de prioriser. C'est-à-dire ...

Une règle simple pour vraiment changer les choses, c'est de commencer ...

```
body
{
    background-image: url("ballons.jpg");
    background-repeat: no-repeat;
}
```

Les propriétés basiques CSS

Ajouter de la couleur et un fond

- Positionner l'image de fond : propriété **background-position:valeur;**

Valeur	description
left top left center left bottom right top right center right bottom center top center center center bottom	La première valeur définit la position horizontale (left, right, center) et le deuxième valeur définit la position verticale (top, center, bottom). Si on omet une valeur Si on ne spécifie qu'un seul mot-clé, l'autre valeur sera "center"
x% y%	La première valeur est la position horizontale et la deuxième valeur est la verticale. Le coin supérieur gauche est 0% 0%. Le coin inférieur droit est à 100% 100%. Si on ne spécifie qu'une seule valeur, l'autre valeur sera de 50%. La valeur par défaut est: 0% 0%
xpos ypos	La première valeur est la position horizontale et la deuxième valeur est la verticale. Le coin supérieur gauche est 0 0. Les unités peuvent être des pixels (0px 0px) ou toute autre unité CSS. Si on ne spécifie qu'une seule valeur, l'autre valeur sera de 50%.

Les propriétés basiques CSS

Ajouter de la couleur et un fond

- un raccourci de toutes les autres propriétés sur le background

background: *bg-color bg-image position bg-repeat bg-attachment*

Peu importe si l'une des valeurs ci-dessus est manquante

Contenu du fichier CSS

```
body{  
background:lightblue url("bird.png") center top no-repeat fixed;  
} /* équivalent à  
background-color:lightblue;  
background-image: url("bird.png");  
background-position: center top;  
background-repeat: no-repeat;  
background-attachment: fixed; */
```



Les propriétés basiques CSS

Transparence : Le CSS nous permet de jouer avec les niveaux de transparence des éléments ! Pour cela, nous allons utiliser des fonctionnalités de CSS3 : la propriété **opacity** et la notation **RGBa**.

- **Transparence avec la propriété opacity :** permet d'indiquer le niveau d'opacité (c'est l'inverse de la transparence).
 - Avec une valeur de 1, l'élément sera totalement opaque : c'est le comportement par défaut.
 - Avec une valeur de 0, l'élément sera totalement transparent.
 - Il faut donc choisir une valeur comprise entre 0 et 1. Ainsi, avec une valeur de 0.6, l'élément sera opaque à 60 %... et on verra donc à travers !

Contenu du fichier CSS

```
body
{
    background: url('texture.jpg');
}

div
{
    background-color: red;
    color: white;
    opacity: 0.3;
}
```

Les écoliers

Le maître, s'il le veut, peut transformer la leçon de lecture en une leçon de langage. Il fera lire les élèves, s'attachera à perfectionner le ton, la prononciation et puis, expliquera la grammaire. La lecture faite, le livre fermé, le maître dictera aux élèves une partie du texte. Il peut aussi composer des phrases nouvelles en servant des mots de la leçon. Les vers peuvent être appris par cœur



Les propriétés basiques CSS

- **Transparence avec la propriété RGBa :** Si on applique la propriété **opacity** à un élément de la page, tout le contenu de cet élément sera rendu transparent (même les images, les autres blocs à l'intérieur, etc.). Si on veut juste rendre la couleur de fond transparente, il faut utiliser plutôt la notation RGBa.
- Il s'agit en fait de la notation RGB que nous avons vue précédemment, mais avec un quatrième paramètre : le niveau de transparence.
- De la même façon que précédemment, avec une valeur de 1, le fond est complètement opaque. Avec une valeur inférieure à 1, il est transparent.

Contenu du fichier CSS

```
body
{
    background: url('texture.jpg');
}

div
{
    background-color: rgba(255, 0, 0, 0.5);
    /* Fond rouge à moitié transparent */
    color: white;
}
```

Les écoliers

Le maître, s'il le veut, peut transformer la leçon de lecture en une leçon de langage. Il fera lire les élèves, s'attachera à perfectionner le ton, la prononciation et puis, expliquera la grammaire. La lecture faite, le livre fermé, le maître dictera aux élèves une partie du texte. Il peut aussi composer des phrases nouvelles en servant des mots de la leçon. Les vers peuvent être appris par cœur



Les propriétés basiques CSS

- **La propriété border: border-width border-style border-color**
- Définit le style des bordures pour différents éléments
- **Valeurs de border-width** : medium, thin, thick, lenght(en px)
- **Valeurs de border-style**: none, dotted, dashed, solid, double, groove, ridge, inset, outset
- **Valeurs de border-color** : nom de la couleur, notation hexadécimale ou RGB .

Exemple 1

```
<body>
<h1>un titre avec une bordure solid rouge </h1>
<h2>un titre avec une bordure pointillée bleu</h2>
<div>un élément div avec une bordure double</div>
</body>
```

```
h1 {
    border: 5px solid red;
}
h2 {
    border: 4px dotted blue;
}
div {
    border: double;
}
```

The screenshot shows a web browser window with a toolbar at the top containing various icons and links. Below the toolbar, there are three distinct sections separated by horizontal lines. The first section, enclosed in a red border, contains the text "un titre avec une bordure solid rouge". The second section, enclosed in a blue dotted border, contains the text "un titre avec une bordure pointillée bleu". The third section, enclosed in a double black border, contains the text "un élément div avec une bordure double". This visual representation demonstrates how the CSS properties defined in the code block above are applied to the corresponding HTML elements.

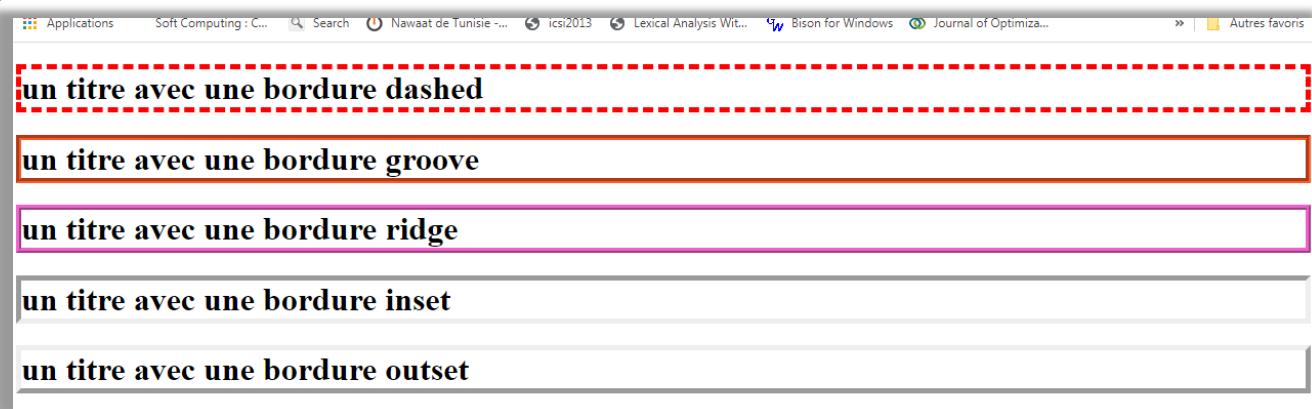
Les propriétés basiques CSS

- **La propriété border: border-width border-style border-color**
- Définit le style des bordures pour différents éléments
- **Valeurs de border-width** : medium, thin, thick, lenght(en px)
- **Valeurs de border-style**: none, dotted, dashed, solid, double, groove, ridge, inset, outset
- **Valeurs de border-color** : nom de la couleur, notation hexadécimale ou RGB .

Exemple 2

```
<body>
<h1 id="titre1">un titre avec une bordure dashed
</h1>
<h1 id="titre2">un titre avec une bordure groove </h2>
<h1 id="titre3">un titre avec une bordure ridge </h2>
<h1 id="titre4">un titre avec une bordure inset </h2>
<h1 id="titre5">un titre avec une bordure outset </h2>
</body>
```

```
#titre1 { border: 5px dashed red;}
#titre2 { border: 5px groove #FF5A28;}
#titre3 { border: 5px ridge rgb(240,96,204);}
#titre4 { border: 5px inset; }
#titre5 { border: 5px outset; }
```



Les propriétés basiques CSS

- **Autre propriétés de bordure**
- border-top : haut
- border-bottom : bas
- border-left : gauche
- border-right : droite
- border-radius : pour arrondir une bordure, peut prendre de 1 à 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 (en px), le décalage vertical de l'ombre (en px), l'adoucissement du dégradé (en px) et la couleur de l'ombre.

Exemple 2

```
<body>
<h1 id="titre1">un titre avec différentes options de bordure </h1>
<h1 id="titre2">un autre titre avec différentes options de bordure </h1>
</body>
```

un titre avec différentes options de bordure

un autre titre avec différentes options de bordure

```
#titre1
{border-bottom: 3px red dotted;
border-top: 3px red dashed;
border-right: 3px blue solid;
border-left: 3px blue double;
border-radius: 25px; /*pour les quatre coins */
box-shadow: 3px 3px 1px gray;
}

#titre2{
border:3px solid blue;
border-radius: 50px 20px; /* 50px pour haut à gauche et bas à droite 20px pour haut à droite bas à gauche*/
box-shadow: 6px 6px 2px gray;
}
```

Les propriétés basiques CSS

- **Propriété border-collapse : définit si les bordures de tableau doivent se réduire en une seule bordure ou être séparées comme dans le HTML standard.**
- **Valeurs de border-collapse**
 - **separate:** les bordures des cellules sont séparées; chaque cellule affichera ses propres bordures. (c'est la valeur par défaut)
 - **Collapse:** Les bordures sont réduites en une seule bordure

```
<body>
  <h2>border-collapse: separate (default):</h2>
  <table id="table1">
    <tr>
      <th>Prénom</th>
      <th>Nom</th>
    </tr>
    <tr>
      <td>Peter</td>
      <td>Griffin</td>
    </tr>
    <tr>
      <td>Lois</td>
      <td>Griffin</td>
    </tr>
    </table>
```

```
<h2>border-collapse: collapse:</h2>
<table id="table2">
  <tr>
    <th>Prénom</th>
    <th>Nom</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>
</body>
```

```
table, td, th {
  border: 1px solid black;
}
#table1 {
  border-collapse: separate;
}

#table2 {
  border-collapse: collapse;
}
```

border-collapse: separate (default):

Prénom	Nom
Peter	Griffin
Lois	Griffin

border-collapse: collapse:

Prénom	Nom
Peter	Griffin
Lois	Griffin

Les propriétés basiques CSS

- **Propriété text-shadow :** ajoute une ombre au texte
- prend quatre valeurs, le décalage horizontal de l'ombre (en px), le décalage vertical de l'ombre (en px), l'adoucissement du dégradé (en px) et la couleur de l'ombre.

```
<body>  
  
  <div>  
    <h1 id="titre1"> premier exemple de Text-shadow </h1>  
    <h1 id="titre2"> deuxième exemple de Text-shadow</h1>  
  </div>  
</body>
```

```
#titre1  
{text-shadow: 2px 2px 8px #FF0000;}  
  
#titre2  
{color: white;  
 text-shadow: 2px 2px 4px #000000;}
```



The screenshot shows a web page with two main headings. The first heading, "premier exemple de Text-shadow", is displayed in a dark red color with a black text shadow. The second heading, "deuxième exemple de Text-shadow", is displayed in a light beige color with a black text shadow. Both headings are contained within a single

element.

premier exemple de Text-shadow

deuxième exemple de Text-shadow

Les propriétés basiques CSS

Comment changer les puces ou les numéros?

- **La propriété: list-style-type** spécifie le type de puce dans une liste.
- Valeurs possibles pour les listes non ordonnées : circle, disc, square,
- Valeurs possibles pour les listes ordonnées decimal, lower-alpha , lower-greek, lower-latin, lower-roman, upper-alpha, upper-greek, upper-latin, upper-roman,
- Valeur none (pas de puces)

The list-style-type Property

- Circle type
 - Tunisie
 - Algérie
- Disc type
 - Tunisie
 - Algérie
- Square type
 - Tunisie
 - Algérie
- 1. Decimal type
 - 2. Tunisie
 - 3. Algérie
- 01. Decimal-leading-zero type
 - 02. Tunisie
 - 03. Algérie
- a. Lower-greek type
 - β. Tunisie
 - γ. Algérie
- a. Lower-latin type
 - b. Tunisie
 - c. Algérie

```
/*Contenu du fichier style.css*/
li:first-child{ color:red; font-style:bold; font-size:18px; }
ul.a {list-style-type: circle;}
ul.b {list-style-type: disc;}
ul.c {list-style-type: square;}
ol.f {list-style-type: decimal;}
ol.g {list-style-type: decimal-leading-zero;}
ol.o {list-style-type: lower-greek;}
ol.p {list-style-type: lower-latin;}
ol.q {list-style-type: lower-roman;}
ol.t {list-style-type: upper-latin;}
ol.u {list-style-type: upper-roman;}
ol.v {list-style-type: none;}
```

Les propriétés basiques CSS

On peut aussi utiliser une image comme puce pour les *listes non ordonnées*

La propriété: list-style-image: url ; remplace la puce dans une liste non ordonnée par une image

```
<body>
<h1>Taches aujourd'hui</h1>
<ul>
  <li>Travail</li>
  <li>courses</li>
  <li>cinéma</li>
</ul>
</body>
```

```
/*Contenu du fichier style.css*/
ul {
  list-style-image: url('to-do.png');
}
```

Taches aujourd'hui

-  Travail
-  courses
-  cinéma

Les propriétés basiques CSS

- La propriété **list-style-position**: Spécifie la position des puces
- Valeurs:
 - **inside** Les puces seront à l'intérieur de l'élément de liste
 - **outside** Les puces seront à l'extérieur de l'élément de liste

```
/*Contenu du fichier style.css*/
ul.a {
    list-style-position: inside;
}
ul.b {
    list-style-position: outside;
}
```

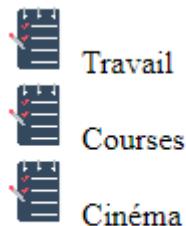
- La clé dans tout ce que l'on fait, c'est de prioriser. C'est-à-dire de cerner les points importants à traiter pour nous, et de s'y tenir
- Une règle simple pour vraiment changer les choses, c'est de commencer toujours par le plus simple, et non par le plus difficile
- La clé dans tout ce que l'on fait, c'est de prioriser. C'est-à-dire de cerner les points importants à traiter pour nous, et de s'y tenir
- Une règle simple pour vraiment changer les choses, c'est de commencer toujours par le plus simple, et non par le plus difficile

Les propriétés basiques CSS

- Un raccourci **list-style**: est un raccourci pour les propriétés suivantes:
- **list-style-type** : si un style-image est spécifié, ceci sera affiché quand l'image ne peut être affichée
- **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

```
<body>
<ul>
  <li>Travail</li>
  <li>courses</li>
  <li>cinéma</li>
</ul>
</body>
```

```
/*Contenu du fichier style.css*/
ul {
  list-style:square inside url('to-do.png');
}
```



Les positions CSS

- La propriété **position** spécifie le type de méthode de positionnement utilisé pour un élément(static, relative, absolute, fixed, ou sticky);
- **position: static;** Valeur par défaut. Les éléments sont rendus dans l'ordre, tels qu'ils apparaissent dans le flux de documents. Les propriétés *top* et *left* n'ont aucun effet.
- **position: absolute;** L'élément est positionné par rapport à son élément parent.

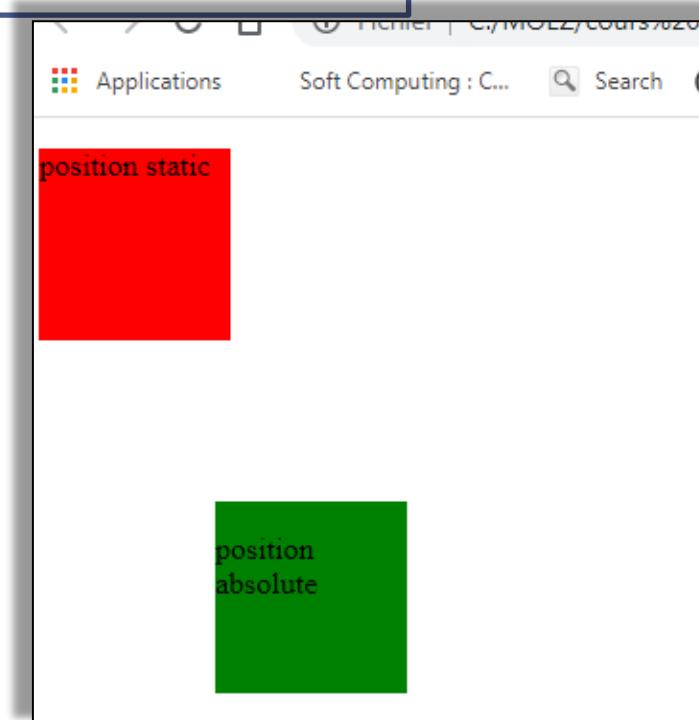
```
<body>
  <div id="myDivStatic">
    <p> position static </p>
  </div>
  <div id="myDivAbsolute">
    <p> position absolute </p>
  </div>
</body>
```

```
/*Contenu du fichier style.css*/
#myDivStatic {
  position:static;
  width:100px;
  height:100px;
  background:red;
  left:100px;
  top:200px;
}
#myDivAbsolute {
  position:absolute;
  width:100px;
  height:100px;
  background:green;
  left:100px;
  top:200px;
}
```

Les positions CSS

- **position: static; Vs position: absolute;**

```
<body>
  <div id="myDivStatic">
    <p> position static </p>
  </div>
  <div id="myDivAbsolute">
    <p> position absolute </p>
  </div>
</body>
```



```
/*Contenu du fichier style.css*/
#myDivStatic {
  position:static;
  width:100px;
  height:100px;
  background:red;
  left:100px;
  top:200px;
}

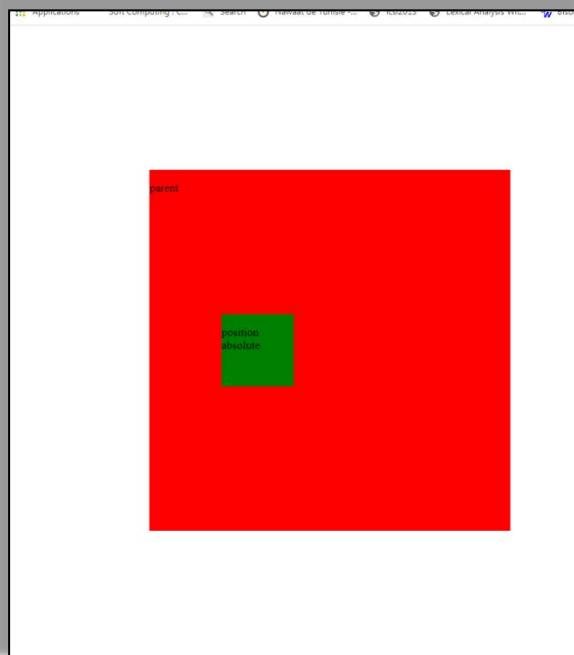
#myDivAbsolute {
  position:absolute;
  width:100px;
  height:100px;
  background:green;
  left:100px;
  top:200px;
}

/* les propriétés left et top
n'ont eu aucun effet lorsque la
position est static
```

Les positions CSS

- **position: absolute;** (position par rapport au parent)

```
<body>
<div id="myDivParent">
  <p> parent </p>
  <div id="myDivAbsolute">
    <p> position absolute </p>
  </div>
</div>
</body>
```



```
#myDivParent {
  position: absolute;
  width: 500px;
  height: 500px;
  background: red;
  left: 200px;
  top: 200px;
}
```

```
#myDivAbsolute {
  position: absolute;
  width: 100px;
  height: 100px;
  background: green;
  left: 100px;
  top: 200px;
}
/* myDivAbsolute est
positionée par rapport au coin
supérieur gauche de son père
myDivParent
```

Les positions CSS

- **position: relative;** L'élément est positionné par rapport à sa position normale
- Exemple : Dans le deuxième écran div3 est décalée par rapport à sa position normale dans le flot (écran de gauche).

```
<body>
  <div id="div1">
    <p> première Division </p>
  </div>
  <div id="div2">
    <p> deuxième division </p>
  </div>
  <div id="div3">
    <p> troisième division </p>
  </div>
</body>
```



/* Position normale des boîtes*/

```
#div1 {
  position: relative;
  height: 100px;
  width: 300px;
  background: red;
  font-size: 30px;
  color: white;
}
```

```
#div2 {
  position: relative;
  height: 100px;
  width: 300px;
  background: green;
  font-size: 30px;
  color: white;
}
```

```
#div3 {
  position: relative;
  height: 100px;
  width: 300px;
  background: blue;
  font-size: 30px;
  color: white;
}
```

/* décalage de div3 par rapport à sa position normale*/

```
#div1 {
  position: relative;
  height: 100px;
  width: 300px;
  background: red;
  font-size: 30px;
  color: white;
}
```

```
#div2 {
  position: relative;
  height: 100px;
  width: 300px;
  background: green;
  font-size: 30px;
  color: white;
}
```

```
#div3 {
  position: relative;
  height: 100px;
  width: 300px;
  background: blue;
  font-size: 30px;
  color: white;
}
top: 50px;
left: 80px; }
```

Les positions CSS

- **position: sticky;** L'élément est positionné en fonction de la position de défilement de l'utilisateur. Un élément sticky **bascule entre relative et fixed**, en fonction de la position de défilement. Il est positionné « **relative** » jusqu'à ce qu'il atteigne une position donnée dans la fenêtre - puis il devient fixe (comme **position: fixed**).
- Voir exemple https://www.w3schools.com/cssref/tryit.asp?filename=trycss_position_sticky

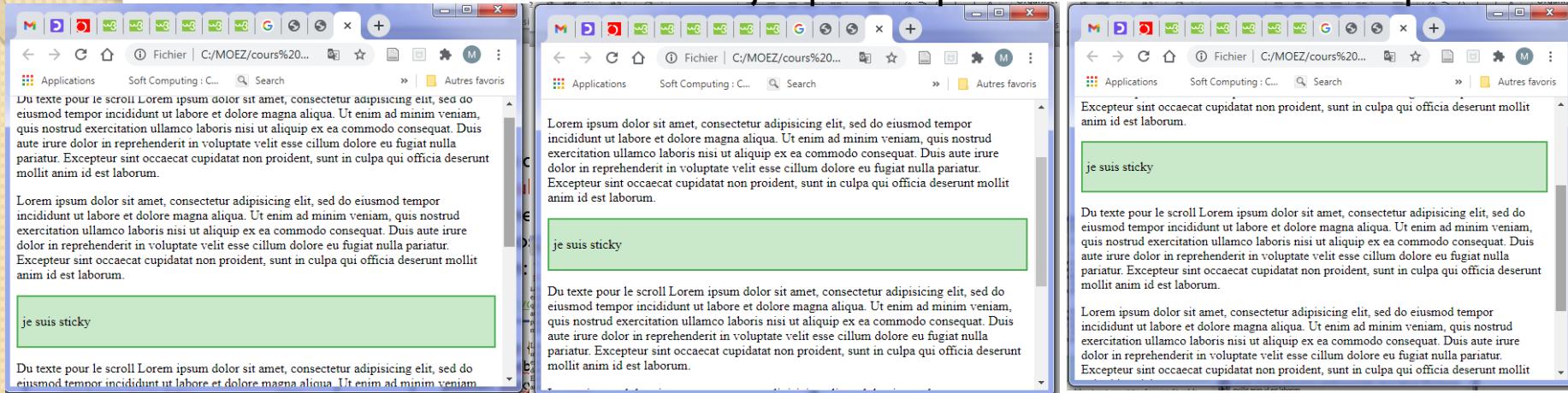
```
<body>
  <div id="div1">
    <p> Du texte pour le scroll Lorem ipsum .... </p>
    <p> Lorem ipsum dolor ... </p>
  </div>
  <div id="myDIVSticky">
    <p> je suis sticky</p>
  </div>
  <div id="div2">
    <p> Du texte pour le scroll Lorem ... </p>
    <p> Lorem ipsum dolor sit amet,</p>
    <p> Lorem ipsum dolor sit amet, </p>
  </div>
```

```
#myDIVSticky {
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
```

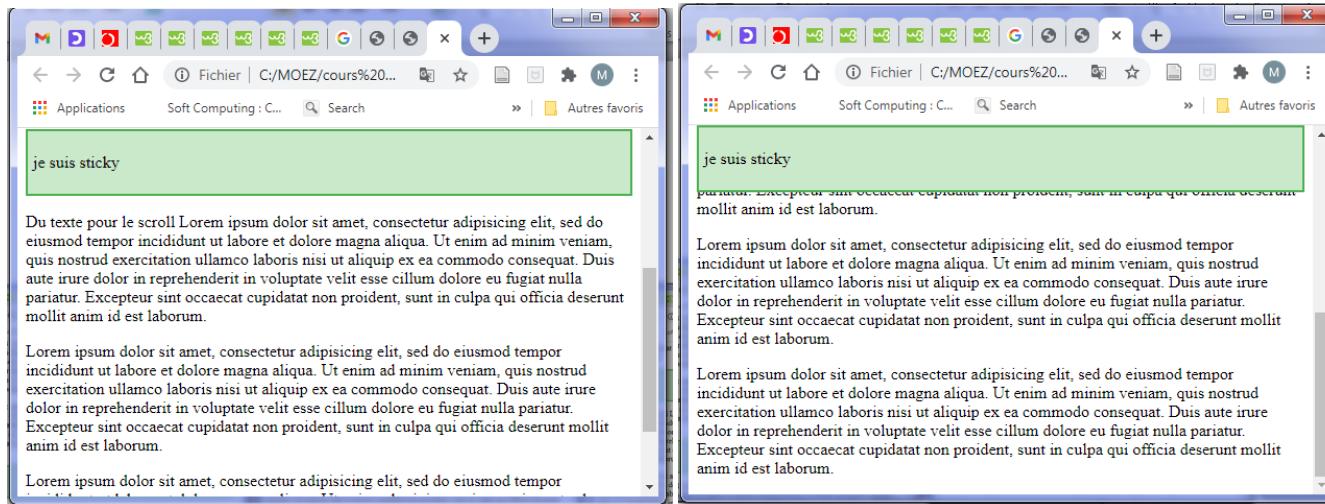
Les positions CSS

- **position: sticky**

Le bloc en vert défile avec le texte jusqu'à ce que le bloc en vert atteint top=0



Ensuite le bloc en vert se fixe à la position top 0 et le texte continue à défiler



Les positions CSS

- La propriété **z-index** : spécifie l'ordre d'empilement d'un élément. Il 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*.
- *Valeur :auto* : Définit l'ordre d'empilement égal à ses parents. (par défaut).
- *Valeur:nombre*: Définit l'ordre d'empilement de l'élément. Les nombres négatifs sont autorisés.

```
<body>
<div class="d1">
<h1>Le monde des oiseaux</h1>
</div>

<div class="d2">
  <p>Titre z-index: 0, image z-index: 1 paragraphe z-index:2</p>
</div>
</body>
```

Le monde des oiseaux

Titre z-index: 0, image z-index: 1 paragraphe z-index:2



```
img { position: absolute;
left: 0px; top: 0px;
width:150px; height:150px;
z-index:1; }
```

```
div.d1
{ position:absolute; top:10px;
z-index:0;
}
div.d2
{position:absolute;
font-size:20px;
top:60px;
z-index:2;}
```

Les boites: dimensions et marges

- Une page web peut être vue comme une succession et un empilement de boîtes, qu'on appelle « blocs » « boxes » en anglais . La plupart des éléments vus au chapitre HTML sont des blocs `<div>`, `<header>` , `<article>` , `<nav>` ...et aussi les paragraphes `<p>` , les titres `<h1>`,
- Dans ce chapitre, nous allons apprendre à manipuler ces blocs comme des boîtes. Nous allons leur donner des dimensions, les agencer en jouant sur leurs marges, mais aussi apprendre à gérer leur contenu... pour éviter que le texte ne dépasse de ces blocs

Les balises de type block et inline

En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :

- **block** : une balise de type block sur une page web crée automatiquement un retour à la ligne avant et après. La page web sera en fait constituée d'une série de blocs les uns à la suite des autres. Et il est possible de mettre un bloc à l'intérieur d'un autre. Exemple: `<p>`, `<footer>`, `<h1>`, `<h2>`, `<div>`, `<article>`
- **inline** : une balise de type inline se trouve 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. Exemple `<a>` `` `<mark>` ``, `` ``

Les boites: dimensions et marges

- **Les dimensions.** Nous allons ici travailler uniquement sur des balises de type block.
- **Propriété width** : c'est la largeur du contenu bloc. À exprimer en pixels (px) ou en pourcentage (%) ;
- **Propriété height** : c'est la hauteur du contenu bloc. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).

Remarques :

- Si le bloc a des marges, celles-ci s'ajouteront à la largeur et à la hauteur.
- Par défaut, un bloc prend 100 % de la largeur disponible. On peut le vérifier en appliquant à nos blocs une couleur de fond.



Les boites: dimensions et marges

- **Les dimensions**
- Les pourcentages seront utiles pour créer un design qui s'adapte automatiquement à la résolution d'écran du visiteur.
Toutefois, il se peut qu'on ait besoin de créer des blocs ayant une dimension précise en pixels :

```
h1  
{width:250px;  
height:60px;  
background-color: yellow;  
}  
  
P  
{background-color: yellow;  
font-size:20px;  
width:50%; }/*tous mes paragraphes auront une largeur de 50 % et mes  
titres h1 auront une largeur de 250 pixels et une hauteur de 60  
pixels*/
```



Les boites: dimensions et marges

- **Les dimensions**
- **Minimum et maximum** On peut définir des dimensions minimales et maximales. cela nous permet de définir des dimensions « limites » pour que notre site s'adapte aux différentes résolutions d'écran de nos visiteurs :
 - **min-width** : largeur minimale
 - **min-height** : hauteur minimale
 - **max-width** : largeur maximale
 - **max-height** : hauteur maximale.

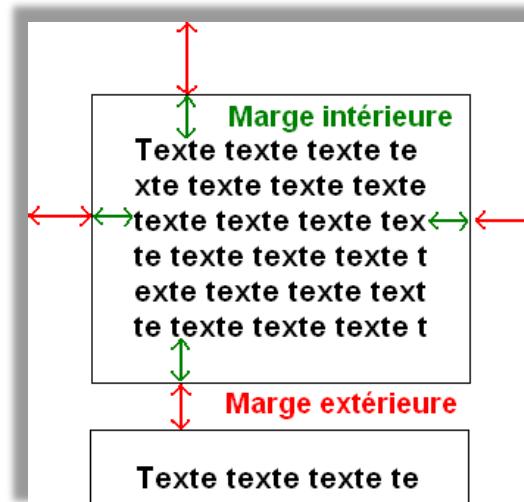
```
P  
{ width: 50%;  
  min-width: 400px; }
```

/*les paragraphes occupent 50 % de la largeur courante de la fenêtre et exiger qu'il fassent au moins 400 pixels de large dans tous les cas
Observez le résultat en modifiant la largeur de la fenêtre de votre navigateur.
Vous allez voir que, si celle-ci est trop petite, le paragraphe se force à occuper au moins 400 pixels de largeur.*/

Les boites: dimensions et marges

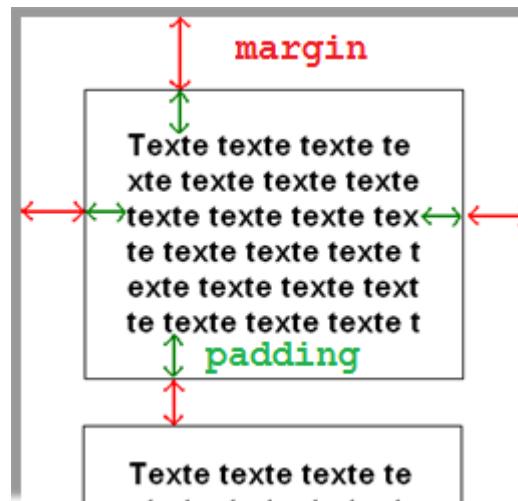
Les marges tous les blocs possèdent des marges. Il existe deux types de marges :

- les marges intérieures ; L'espace entre le texte et la bordure
- les marges extérieures; L'espace entre la bordure et le bloc suivant



Les boites: dimensions et marges

- **Les marges** En CSS, on peut modifier la taille des marges avec les deux propriétés suivantes :
- **padding** : indique la taille de la marge intérieure. À exprimer en général en pixels (px) ;
- **margin** : indique la taille de la marge extérieure. Là encore, on utilise le plus souvent des pixels.



Les boites: dimensions et marges

- **Les marges**

```
p  
{  
width: 350px;  
border: 1px solid black;  
text-align: justify;  
}
```

sans marges définies

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
p  
{  
width: 350px;  
border: 1px solid black;  
text-align: justify;  
padding: 12px; /* Marge  
intérieure de 12px */  
}
```

En ajoutant un padding de 12px

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
p  
{  
width: 350px;  
border: 1px solid black;  
text-align: justify;  
padding: 12px;  
margin: 50px; /* Marge  
extérieure de 50px */  
}
```

Padding et margin

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Les boites: dimensions et marges

Les marges Si on veut spécifier des marges différentes en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises...

- **margin-top** : marge extérieure en haut ;
 - **margin-bottom** : marge extérieure en bas ;
 - **margin-left** : marge extérieure à gauche ;
 - **margin-right** : marge extérieure à droite.
-
- **padding-top** : marge intérieure en haut ;
 - **padding-bottom** : marge intérieure en bas ;
 - **padding-left** : marge intérieure à gauche ;
 - **padding-right** : marge intérieure à droite.

Remarque: Il y a d'autres façons de spécifier les marges avec les propriétés **margin** et **padding**. Par exemple: **margin: 2px 0 3px 1px;** signifie « 2 px de marge en haut, 0 px à droite (le px est facultatif dans ce cas), 3 px en bas, 1 px à gauche ». Autre notation raccourcie : **margin: 2px 1px;** signifie « 2 px de marge en haut et en bas, 1 px de marge à gauche et à droite ».

Les boites: dimensions et marges

Les marges: comment centrer les blocs sur la fenêtre du navigateur ?

- Il faut donner une largeur au bloc (avec la propriété width) ;
- Ensuite indiquer qu'on veut 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;  
}
```

Centrage

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis

Les boites: dimensions et marges

- **Quand ça dépasse... Propriété overflow**
- Lorsqu'on définit des dimensions précises pour nos blocs, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent. La propriété **overflow** permet de contrôler les dépassements... et décider quoi faire si jamais cela devait arriver.
- Supposons qu'on a un paragraphe avec des dimensions fixés à 250 px de large et 110 px de haut. Ajoutons-lui une bordure et remplissons-le de texte... le texte ne tient pas à l'intérieur d'un si petit bloc.

Texte qui dépasse

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

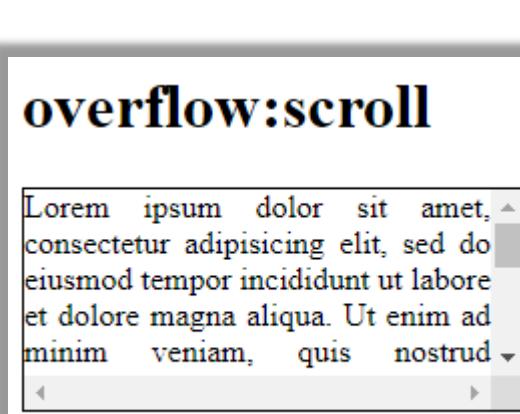
- Si on veut que le texte ne dépasse pas des limites du paragraphe, il va falloir utiliser la propriété **overflow**

Les boites: dimensions et marges

• Quand ça dépasse... 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.
- **auto** : c'est le navigateur qui décide de mettre ou non des barres de défilement (si c'est nécessaire).



overflow:visible

Lore ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Les boites: dimensions et marges

- **La propriété word-wrap : couper les textes trop larges**
- Si on veut placer un mot très long (sans espace ni tiret) dans un bloc, qui ne tient pas dans la largeur, il est nécessaire d'utiliser **word-wrap**. Qui permet de forcer la césure des très longs mots (généralement des adresses un peu longues).
- La figure suivante représente ce que l'on peut avoir quand on écrit une URL un peu longue dans un bloc.

dépassemement en largeur

<file:///C:/COURS/cours%20WEB1/coursPPT%20CSS/Boxes.html>

- Voici ce qui s'affiche lorsqu'on utilise **p { word-wrap: break-word; }**

avec word-wrap: break-word

<file:///C:/COURS/cours%20WEB1/coursPPT%20CSS/Boxes.html>

La propriété float :

- La propriété **float** est utilisée pour le positionnement et la mise en forme du contenu.
- Dans son utilisation la plus simple, la propriété float peut être utilisée pour envelopper du texte autour des images.
- La propriété float peut avoir l'une des valeurs suivantes:
 - **left** L'élément flotte à gauche de son conteneur
 - **right** L'élément flotte à droite de son conteneur
 - **none** (par défaut) L'élément ne flotte pas (sera affiché juste là où il se trouve dans le texte).

img { float: right; }

img { float: left; }

Float Right

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.



Float left

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

img { float: none; }

Float none



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

La propriété **clear** :

- la propriété **clear** : pour annuler **float**
- *Elle prend les valeurs suivantes*
 - **none** (par défaut) Autorise les éléments flottants des deux côtés.
 - **left** Aucun élément flottant autorisé sur le côté gauche.
 - **right** Aucun élément flottant autorisé sur le côté droit.
 - **both** Aucun élément flottant autorisé sur le côté gauche ou droit.

Exemples sur les propriété clear et float :

```
<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>
```

Code CSS

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



Les boites sont affichées selon leur ordre d'apparition

Et si on voulait placer le vert à gauche et le bleu à droite ?

Exemples sur les propriété clear et float :

```
<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>
```

Code CSS

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



Et si on veut que le jaune, reste à la ligne ?

Exemples sur les propriété clear et float :

```
<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>
```

Code CSS

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



redgreenblue

yellow

Exemples sur les propriété **clear** et **float** :

```
<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>
```

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

.container > div
{ text-align: center; }
```

redgreenblue

yellow

redgreenblue

yellow

Exemples sur les propriété **clear** et **float** :

```
<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>
```

```
.component1 { background-color: red; float:left; }
.component2 {background-color: green; float:left; }
.component3 {background-color: blue; float:left; }
.component4 {background-color: yellow;clear:left; }

.container > div
{ text-align: center; }
```

redgreenblue

yellow

Exemples sur les propriété clear et float :

```
<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>
```

```
.component1 { background-color: red; float:left; height:90px; width: 100px}
.component2 {background-color: green; height:30px; }
.component3 {background-color: blue; height:30px; }
.component4 {background-color: yellow; height:30px; }

.container > div
{ text-align: center; }
```

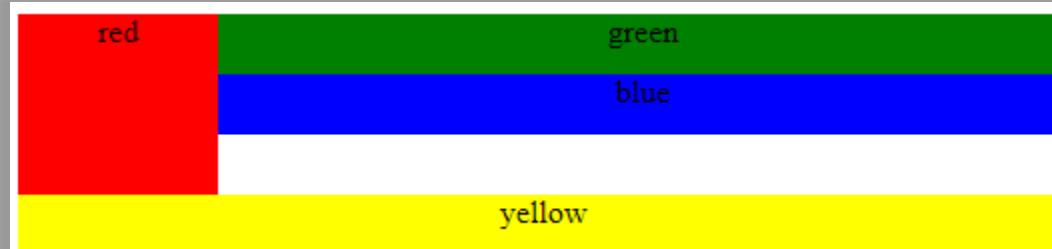


Exemples sur les propriété clear et float :

```
<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>
```

```
.component1 { background-color: red; float:left; height:90px; width: 100px}
.component2 {background-color: green; height:30px; }
.component3 {background-color: blue; height:30px; }
.component4 {background-color: yellow; clear: left; height:30px; }

.container > div
{ text-align: center; }
```



Exemples sur les propriété clear et float :

```
<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>
```

```
.component1 { background-color: red; float:left; height:90px; width: 100px}
.component2 {background-color: green; height:30px; }
.component3 {background-color: blue; clear: left; height:30px; }
.component4 {background-color: yellow; height:30px;}

.container > div
```

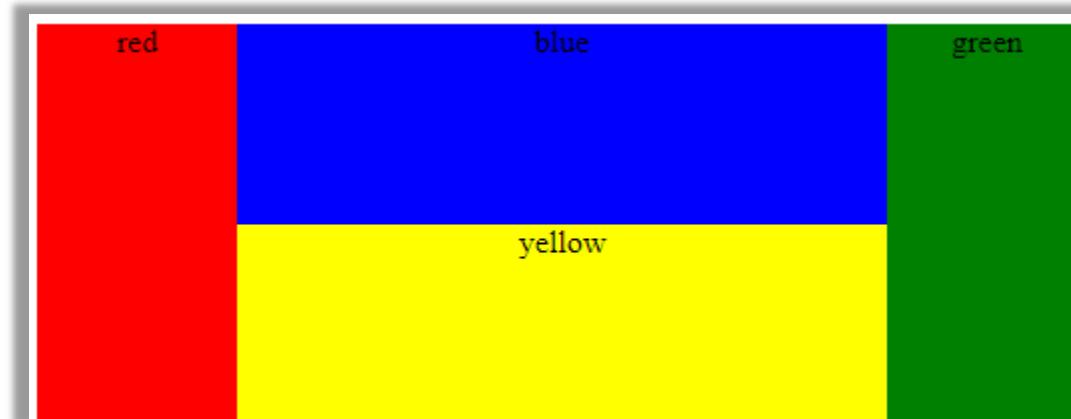


Exemples sur les propriété clear et float :

```
<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>
```

```
.component1 { background-color: red; float:left; height:200px; width: 100px}
.component2 {background-color: green; float:right; height:200px; width: 100px}
.component3 {background-color: blue; height:100px; }
.component4 {background-color: yellow; height:100px; }

.container > div
{ text-align: center; }
```

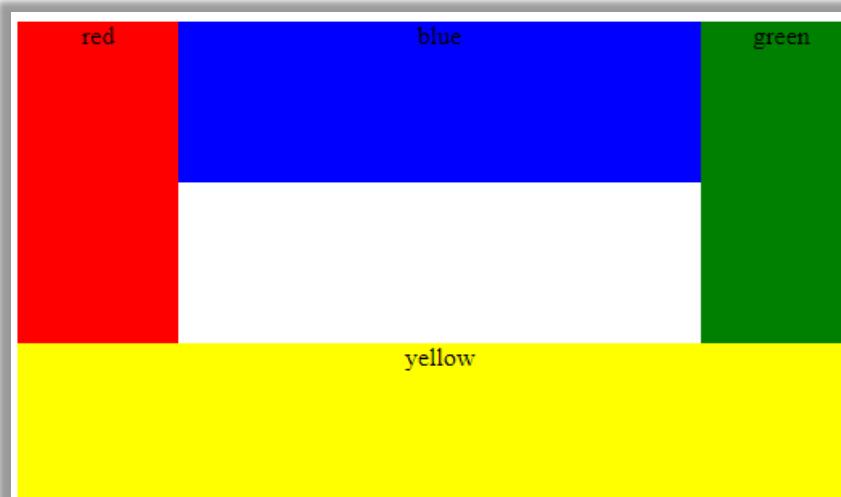


Exemples sur les propriété clear et float :

```
<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>
```

```
.component1 { background-color: red; float:left; height:200px; width: 100px}
.component2 {background-color: green; float:right; height:200px; width: 100px }
.component3 {background-color: blue; height:100px; }
.component4 {background-color: yellow; clear: both; height:100px; }

.container > div
{ text-align: center; }
```

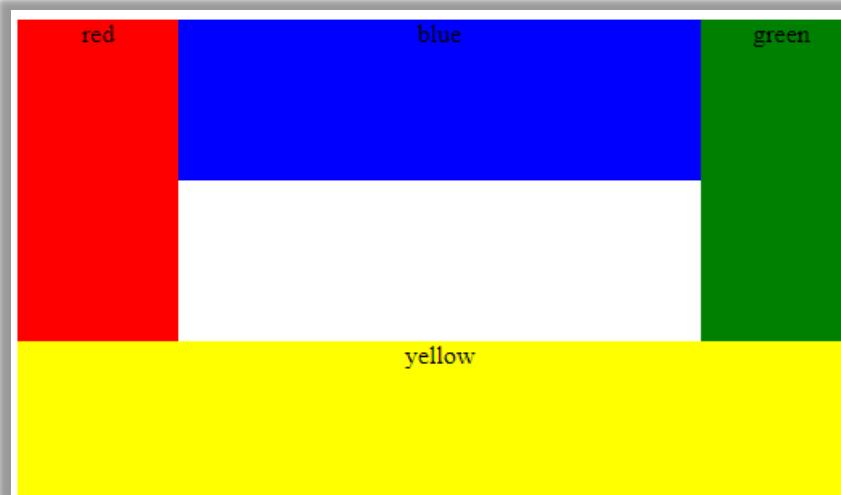


Exemples sur les propriété clear et float :

```
<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>
```

```
.component1 { background-color: red; float:left; height:200px; width: 100px}
.component2 {background-color: green; float:right; height:200px; width: 100px }
.component3 {background-color: blue; height:100px; }
.component4 {background-color: yellow; clear: left; height:100px; }

.container > div
{ text-align: center; }
```



La propriété visibility

Spécifie si un élément est visible ou non.

Elle prend les valeurs suivantes:

- Visible : valeur par défaut, l'élément est visible
- Hidden: l'élément est masqué mais prend toujours de la place.
- Collapse: Uniquement pour les lignes de table (`<tr>`), les groupes de lignes (`<tbody>`), les colonnes (`<col>`), les groupes de colonnes (`<colgroup>`). Cette valeur masque une ligne ou une colonne, mais n'affecte pas la disposition du tableau. L'espace occupé par la ligne ou la colonne sera disponible pour d'autres contenus.
- Si « collapse » est utilisée sur d'autres éléments, il se comporte comme "hidden"

La propriété visibility

Exemple 1:

```
<body>
  <h1>Propriété visibility </h1>
  <h2 class="a">Ce titre est visible</h2>
  <h2 class="b">ce titre est caché</h2>
  <p>le titre masqué occupe toujours de l'espace sur la page.</p>
</body>
```

```
h2.a {
  visibility: visible;
}

h2.b {
  visibility: hidden;
}
```

Propriété visibility

Ce titre est visible

le titre masqué occupe toujours de l'espace sur la page.

La propriété visibility

Exemple 2: dans les tableaux

```
<body>  
  
<h1>The visibility Property</h1>  
  
<table>  
  <tr>  
    <td>cours</td>  
    <td>HTML</td>  
  </tr>  
  <tr class="masquer">  
    <td>cours</td>  
    <td>CSS</td>  
  </tr>  
  <tr>  
    <td>cours</td>  
    <td>JavaScript</td>  
  </tr>  
</table>  
  
</body>
```

```
table, td {  
  border: 1px solid black;  
}  
  
tr.masquer {  
  visibility: visible;  
}
```

visibility: visible

cours	HTML
cours	CSS
cours	JavaScript

La propriété visibility

Exemple 2: dans les tableaux

```
<body>  
  
<h1>The visibility Property</h1>  
  
<table>  
  <tr>  
    <td>cours</td>  
    <td>HTML</td>  
  </tr>  
  <tr class="masquer">  
    <td>cours</td>  
    <td>CSS</td>  
  </tr>  
  <tr>  
    <td>cours</td>  
    <td>JavaScript</td>  
  </tr>  
</table>  
  
</body>
```

```
table, td {  
  border: 1px solid black;  
}  
  
tr.masquer {  
  visibility: collapse;  
}
```

visibility: collapse

cours	HTML
cours	JavaScript

L'espace occupé par la ligne ou la colonne sera disponible pour d'autres contenus.

La propriété visibility

Exemple 2: dans les tableaux

```
<body>  
  
<h1>The visibility Property</h1>  
  
<table>  
  <tr>  
    <td>cours</td>  
    <td>HTML</td>  
  </tr>  
  <tr class="masquer">  
    <td>cours</td>  
    <td>CSS</td>  
  </tr>  
  <tr>  
    <td>cours</td>  
    <td>JavaScript</td>  
  </tr>  
</table>  
  
</body>
```

```
table, td {  
  border: 1px solid black;  
}  
  
tr.masquer {  
  visibility: hidden;  
}
```

visibility: hidden

cours	HTML
cours	JavaScript

l'élément est masqué mais prend toujours de la place.

La propriété visibility

Exemple 3: cacher une colonne

```
<body>
  <table border="1">
    <colgroup>
      <col class="premiereColonne">
      <col class="deuxiemeColonne">
      <col class="troisiemeColonne">
      <col class="quatriemeColonne">
    </colgroup>

    <thead>
      <tr>
        <th>Première Colonne</th>
        <th>Deuxième Colonne</th>
        <th>Troisième Colonne</th>
        <th>Quatrième Colonne</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>cahier</td>
        <td>1,200</td>
        <td>5</td>
        <td>6,000</td>
      </tr>
      <tr>
        <td>stylo</td>
        <td>0.7</td>
        <td>10</td>
        <td>4,000</td>
      </tr>
    </tbody>
  </table>
```

```
<tr>
  <td>stylo</td>
  <td>0.7</td>
  <td>10</td>
  <td>4,000</td>
</tr>
</tbody>
</table>
```

Code CSS
.troisiemeColonne{
 visibility: collapse;
}

sans visibility: collapse;

Première Colonne	Deuxième Colonne	Troisième Colonne	Quatrième Colonne
cahier	1,200	5	6,000
stylo	0.7	10	4,000

Avec visibility: collapse;

Première Colonne	Deuxième Colonne	Quatrième Colonne
cahier	1,200	6,000
stylo	0.7	4,000

La propriété display

- La propriété display spécifie le mode d'affichage d'un élément. En HTML, la valeur de la propriété d'affichage par défaut est tirée des spécifications HTML.
- Certains éléments ont un display *inline* par défaut comme `` ``.. D'autres ont un display *block* par défaut comme `<p>` `<h1>``<h2>`.
- Grace à la propriété display on peut changer le mode d'affichage par défaut.
- Valeurs les plus usuelles
 - **inline** Affiche un élément en tant qu'élément inline(comme ``).Toutes les propriétés de hauteur et de largeur n'auront aucun effet
 - **block** Affiche un élément en tant qu'élément block(comme `<p>`). Il commence sur une nouvelle ligne et occupe toute la largeur
 - **inline-block** L'élément est formaté en tant qu'élément 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,...
 - **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).

La propriété display

- Exemple sur `display: none, inline, block, inline-block`

```
<body>
<h1>The display Property</h1>
```

```
<h2>display: none:</h2>
```

```
<div>


  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. <p class="ex1">JE NE  
M'AFFICHE PAS!</p> Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.


</div>
```

```
<h2>display: inline:</h2>
```

```
<div>


  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. <p class="ex2">JE M'AFFICHE  
INLINE</p> Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.


</div>
```

Contenu du fichier css

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



```
p.ex1 {display: none;}
```

```
p.ex2 {display: inline;}
```

```
p.ex3 {display: block;}
```

```
p.ex4 {display: inline-block;}
```

```
<h2>display: block:</h2>
```

```
<div>


  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. <p class="ex3">JE M'AFFICHE  
BLOCK!</p> Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.


</div>
```

```
<h2>display: inline-block:</h2>
```

```
<div>


  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. <p class="ex4">Je M'AFFICHE INLINE-  
BLOCK!</p> Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.


</div>
```

The display Property

display: none:

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

display: inline:

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. **JE M'AFFICHE INLINE** Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

display: block:

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

JE M'AFFICHE BLOCK!

 Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

display: inline-block:

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. **Je M'AFFICHE INLINE-BLOCK!** Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

La propriété display

- **display:list-item;** permet à l'élément de se comporter comme un élément et s'afficher sous forme de liste après retour à la ligne après chaque élément

```
<body>
<h1>display List-item</h1>
<p> <span> ceci est un premier span </span>
<span id="monSpan"> ceci est un span qui s'affiche en
list-item</span>
<span> ceci est un troisième span </span>
<span> ceci est un quatrième span </span></p>
</body>
</html>
```

```
#monSpan {
background-color:red;
display:list-item;
}
/*ce span va effectuer un
retour à la ligne avant et
après
```

display List-item

ceci est un premier span
ceci est un span qui s'affiche en list-item
ceci est un troisième span ceci est un quatrième span

La propriété display

- **Exercice:** étant donné le code HTML permettant de créer un menu
- écrire un premier code CSS permettant de créer un menu vertical
- écrire un deuxième code CSS permettant de créer un menu horizontal

```
<nav>
<ul id="menu">
  <li><a href="page1.html">menu1 </a>
    <ul class="sous-menu">
      <li><a href="page11.html">menu11 </a></li>
      <li><a href="page12.html">menu12 </a></li>
    </ul>
  </li>
  <li><a href="page2.html">menu2 </a></li>
  <li><a href="page3.html">menu3 </a></li>
  <li><a href="page4.html">menu4 </a>
    <ul class="sous-menu">
      <li><a href="page41.html">menu41 </a></li>
      <li><a href="page42.html">menu42 </a></li>
    </ul>
  </li>
</ul>
</nav>
```

La propriété display

- **Exercice:** étant donné le code HTML permettant de créer un menu écrire un premier code CSS permettant de créer un menu vertical
- Code CSS permettant de créer un menu vertical

```
<nav>
<ul id="menu">
  <li><a href="page1.html">menu1 </a>
    <ul class="sous-menu">
      <li><a href="page11.html">menu11 </a></li>
      <li><a href="page12.html">menu12 </a></li>
    </ul>
  </li>
  <li><a href="page2.html">menu2 </a></li>
  <li><a href="page3.html">menu3 </a></li>
  <li><a href="page4.html">menu4 </a>
    <ul class="sous-menu">
      <li><a href="page41.html">menu41 </a></li>
      <li><a href="page42.html">menu42 </a></li>
    </ul>
  </li>
</ul>
</nav>
```

```
ul {
  list-style-type: none;
}
.sous-menu {
  display: none;
}
li:hover > ul {
  display: list-item;
}
```

menu1
 menu11
 menu12
menu2
menu3
menu4

La propriété display

- **Exercice:** étant donné le code HTML permettant de créer un menu écrire un premier code CSS permettant de créer un menu vertical
- Code CSS permettant de créer un menu vertical

```
<nav>
<ul id="menu">
  <li><a href="page1.html">menu1 </a>
    <ul class="sous-menu">
      <li><a href="page11.html">menu11 </a></li>
      <li><a href="page12.html">menu12 </a></li>
    </ul>
  </li>
  <li><a href="page2.html">menu2 </a></li>
  <li><a href="page3.html">menu3 </a></li>
  <li><a href="page4.html">menu4 </a>
    <ul class="sous-menu">
      <li><a href="page41.html">menu41 </a></li>
      <li><a href="page42.html">menu42 </a></li>
    </ul>
  </li>
</ul>
</nav>
```

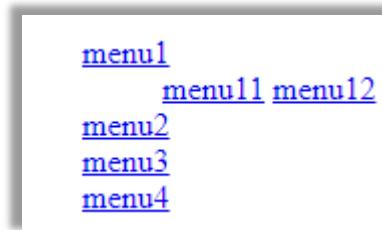
```
ul {
  list-style-type: none;
}
.sous-menu {
  display: none;
}
li:hover > ul {
  display: inline;
  /* inline par rapport au flux du parent */
}
```

menu1
menu11
menu12
menu2
menu3
menu4

La propriété display

- **Exercice:** étant donné le code HTML permettant de créer un menu écrire un premier code CSS permettant de créer un menu vertical
- Code CSS permettant de créer un menu vertical et un sous menu horizontal

```
<nav>
<ul id="menu">
  <li><a href="page1.html">menu1 </a>
    <ul class="sous-menu">
      <li><a href="page11.html">menu11 </a></li>
      <li><a href="page12.html">menu12 </a></li>
    </ul>
  </li>
  <li><a href="page2.html">menu2 </a></li>
  <li><a href="page3.html">menu3 </a></li>
  <li><a href="page4.html">menu4 </a>
    <ul class="sous-menu">
      <li><a href="page41.html">menu41 </a></li>
      <li><a href="page42.html">menu42 </a></li>
    </ul>
  </li>
</ul>
</nav>
```



```
ul {
  list-style-type: none;
}
.sous-menu {
  display: none;
}
li:hover > ul {
  display: list-item;
}

li:hover > ul > li {
  display: inline;
}
```

La propriété display

- **Exercice:** étant donné le code HTML permettant de créer un menu écrire un code CSS permettant de créer un menu vertical
- Une deuxième solution pour le menu vertical consistant à afficher les sous-menu à coté du menu principal

```
<nav>
<ul id="menu">
  <li><a href="page1.html">menu1 </a>
    <ul class="sous-menu">
      <li><a href="page11.html">menu11 </a></li>
      <li><a href="page12.html">menu12 </a></li>
    </ul>
  </li>
  <li><a href="page2.html">menu2 </a></li>
  <li><a href="page3.html">menu3 </a></li>
  <li><a href="page4.html">menu4 </a>
    <ul class="sous-menu">
      <li><a href="page41.html">menu41 </a></li>
      <li><a href="page42.html">menu42 </a></li>
    </ul>
  </li>
</ul>
</nav>
```

Avec padding-left:0px;

[menu1](#) [menu11](#)
[menu2](#) [menu12](#)
[menu3](#)
[menu4](#)

sans padding-left:0px;

[menu1](#) [menu11](#)
[menu2](#) [menu12](#)
[menu3](#)
[menu4](#)

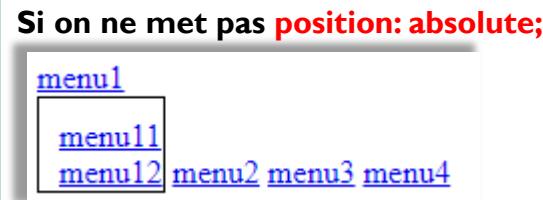
```
ul {
  list-style-type: none;
}
.sous-menu {
  display: none;
}
li:hover > ul {
  display: inline-block;
/*pour pouvoir annuler les
marges*/
position: absolute;
/*position par rapport au
parent*/
padding-left:0px;
/* sinon il y'aura une marge à
gauche */
}
```

La propriété display

- **Exercice:** étant donné le code HTML permettant de créer un menu écrire un premier code CSS permettant de créer un menu horizontal
- Code CSS permettant de créer un menu horizontal

```
<nav>
<ul id="menu">
  <li><a href="page1.html">menu1 </a>
    <ul class="sous-menu">
      <li><a href="page11.html">menu11 </a></li>
      <li><a href="page12.html">menu12 </a></li>
    </ul>
  </li>
  <li><a href="page2.html">menu2 </a></li>
  <li><a href="page3.html">menu3 </a></li>
  <li><a href="page4.html">menu4 </a>
    <ul class="sous-menu">
      <li><a href="page41.html">menu41 </a></li>
      <li><a href="page42.html">menu42 </a></li>
    </ul>
  </li>
</ul>
</nav>
```

menu1 menu2 menu3 menu4
menu11
menu12

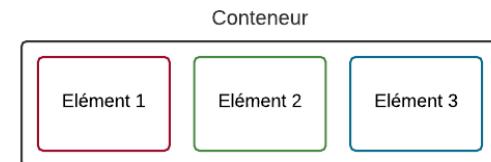
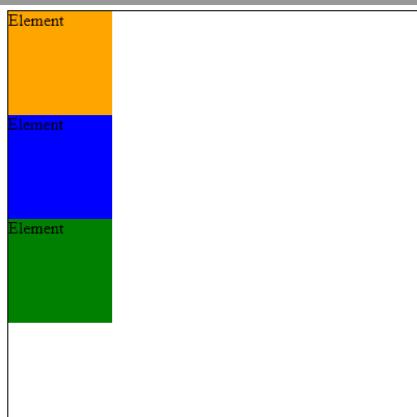


```
li, ul {
  display: inline-block;
  list-style-type: none;
  /* ne pas afficher les puces */
  padding: 0;
}
.sous-menu {
  display: none;
}
li:hover .sous-menu {
  border: 1px solid black;
  display: block;
  /* pour qu'il retourne à la ligne*/
  position: absolute;
  /*position par rapport au parent li
  Sinon menu2 menu3 menu4 se
  positionneront après lui*/
  padding-left: 2px;
  padding-top: 0px;
}
.sous-menu li {
  display: block;
  /* retour à la ligne après chaque li du
  sous menu*/
}
```

Placement de boites avec Flexbox

- Avec **flexbox** on va pouvoir considérer les éléments de la page comme des boîtes flexibles qu'on peut agencer les unes par rapport aux autres de multiples manières.
- On doit tout d'abord définir un conteneur, et à l'intérieur on place plusieurs éléments qu'on pourra agencer à notre guise.
- Le conteneur est une balise HTML, et les éléments sont d'autres balises HTML à l'intérieur
- **Exemple sans Flex box**

```
<div id="conteneur">
    <div class="element 1">Element </div>
    <div class="element 2">Element </div>
    <div class="element 3">Element </div>
</div>
```



Code CSS

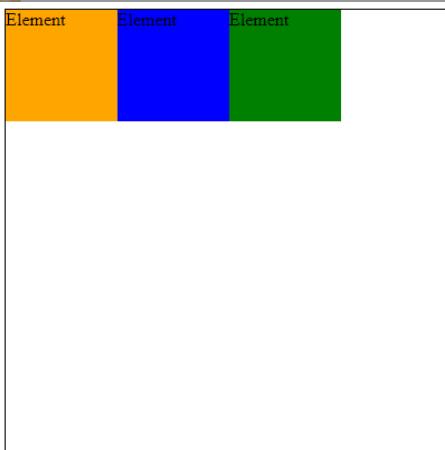
```
#conteneur { height:400px; width:400px;
              border: 1px solid black; }
.element1 { background-color: orange;
            height:100px; width:100px; }
.element2 { background-color: blue;
            height:100px; width:100px; }
.element3 { background-color: green;
            height:100px; width:100px; }
```

Par défaut, les blocs se placent les uns en dessous des autres

Placement de boites avec Flexbox

- Avec **flexbox** on va pouvoir considérer les éléments de la page comme des boîtes flexibles qu'on peut agencer les unes par rapport aux autres de multiples manières.
- On doit tout d'abord définir un conteneur, et à l'intérieur on place plusieurs éléments qu'on pourra agencer à notre guise.
- Le conteneur est une balise HTML, et les éléments sont d'autres balises HTML à l'intérieur
- **Exemple avec Flex box**

```
<div id="conteneur">
    <div class="element1">Element </div>
    <div class="element2">Element </div>
    <div class="element3">Element </div>
</div>
```



```
/* on ajoute display:flex; au conteneur */
#conteneur { display:flex;
              height:400px; width:400px;
              border: 1px solid black;}
.element1 { background-color: orange;
            height:100px; width:100px; }
.element2 { background-color: blue;
            height:100px; width:100px; }
.element3 { background-color: green;
            height:100px; width:100px; }
```

les blocs se placent par défaut côté à côté.

Placement de boites avec Flexbox

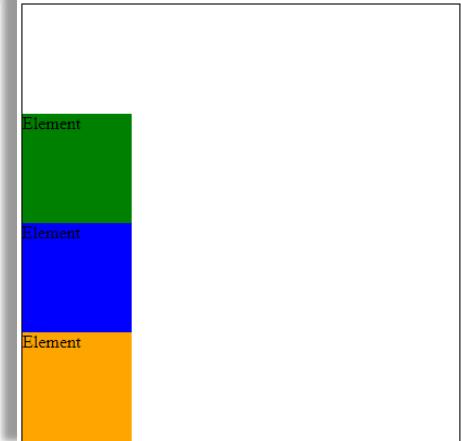
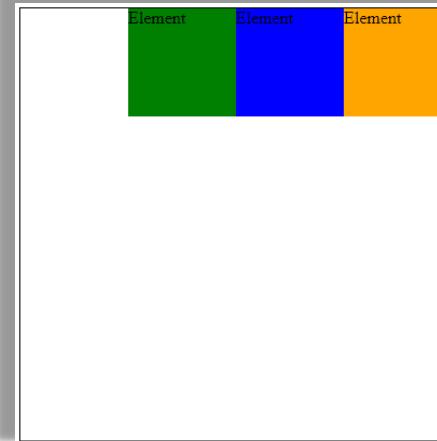
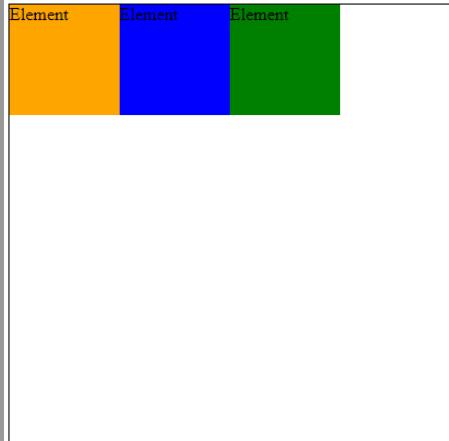
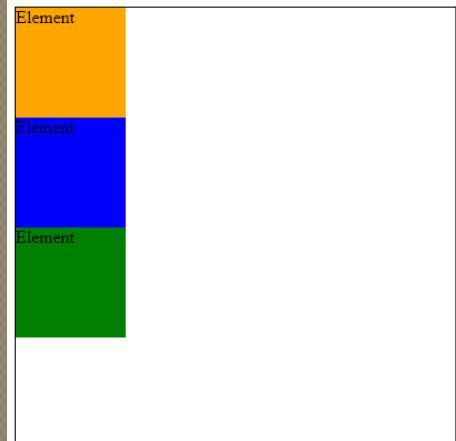
- La direction avec **flex-direction** on peut positionner les boites verticalement ou horizontalement ou encore les inverser. Il peut prendre les valeurs suivantes :
 - **row** : organisés sur une ligne (par défaut) ;
 - **column** : organisés sur une colonne ;
 - **row-reverse** : organisés sur une ligne, mais en ordre inversé ;
 - **column-reverse** : organisés sur une colonne, mais en ordre inversé.

```
/* flex-direction */
#conteneur
{
  display: flex;
  flex-direction: column;
  height:400px; width:400px;
  border: 1px solid black;}
```

```
/* flex-direction */
#conteneur
{
  display: flex;
  flex-direction: row;
  height:400px; width:400px;
  border: 1px solid black;}
```

```
/* flex-direction */
#conteneur
{
  display:flex;
  flex-direction: row-reverse;
  height:400px; width:400px;
  border: 1px solid black;}
```

```
/* flex-direction */
#conteneur
{
  display: flex;
  flex-direction: column-reverse;
  height:400px; width:400px;
  border: 1px solid black;}
```



Placement de boites avec Flexbox

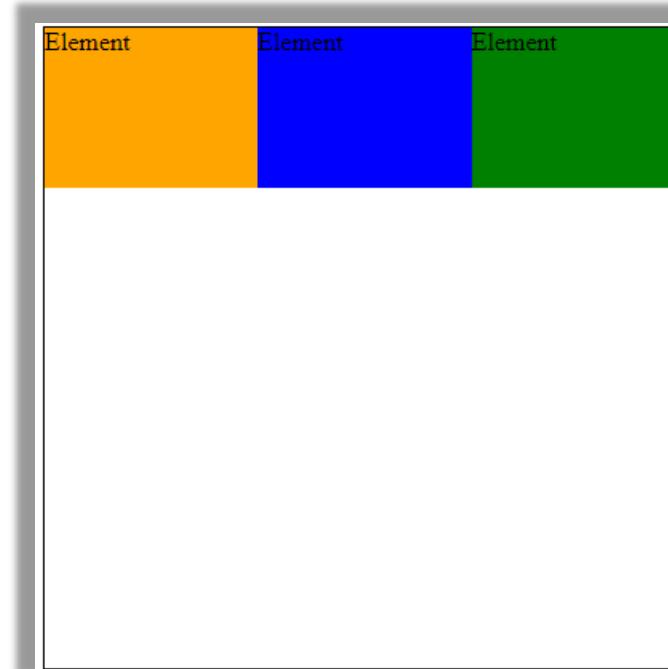
- **Le retour à la ligne avec `flex-wrap`**
- Par défaut, les blocs essaient de rester sur la même ligne s'ils n'ont pas la place (par exemple si la somme de leur largeurs dépasse la largeur du conteneur le navigateur ajuste automatiquement leur largeur par rapport à la largeur du conteneur).
- `flex-wrap` permet aux blocs de revenir à la ligne lorsqu'ils n'ont plus la place,
- `flex-wrap` qui peut prendre les valeurs suivantes :
 - `nowrap` : pas de retour à la ligne (par défaut) ;
 - `wrap` : les éléments vont à la ligne lorsqu'il n'y a plus la place ;
 - `wrap-reverse` : les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

Placement de boites avec Flexbox

- **Le retour à la ligne avec `flex-wrap`**
- `flex-wrap` qui peut prendre les valeurs suivantes :
 - `nowrap` : pas de retour à la ligne (par défaut) ;
 - `wrap` : les éléments vont à la ligne lorsqu'il n'y a plus la place ;
 - `wrap-reverse` : les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

```
/* avec des éléments qui dépassent
Sa largeur 40% + 40% + 40% */
#conteneur
{
  display: flex;
  flex-direction: row;
  flex-wrap: nowrap;
  height:400px; width:400px;
  border: 1px solid black;

.element1 { background-color: orange;
            height:100px; width:40%;}
.element2{ background-color: blue;
            height:100px; width:40%;}
.element3{ background-color: green;
            height:100px; width:40%;}
```



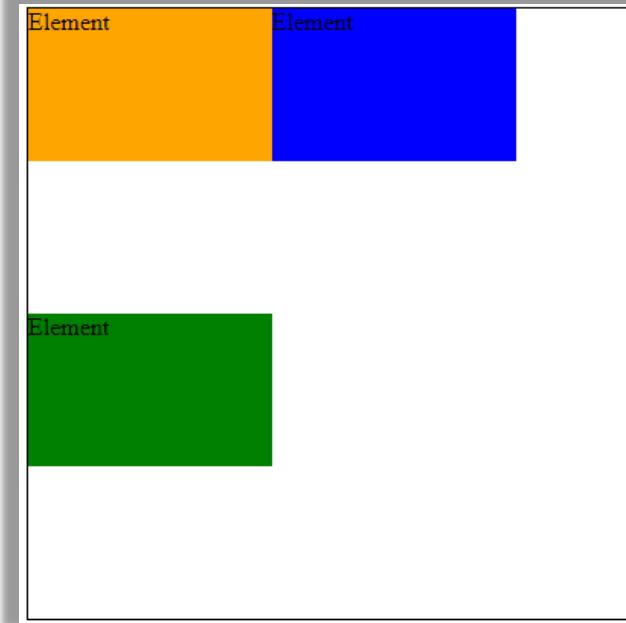
`flex-wrap: nowrap;`

Les boites s'ajustent automatiquement à la largeur du conteneur

Placement de boites avec Flexbox

- **Le retour à la ligne avec `flex-wrap`**
- `flex-wrap` qui peut prendre les valeurs suivantes :
 - `nowrap` : pas de retour à la ligne (par défaut) ;
 - `wrap` : les éléments vont à la ligne lorsqu'il n'y a plus la place ;
 - `wrap-reverse` : les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

```
/*avec des éléments qui dépassent  
Sa largeur 40% + 40% + 40% */  
#conteneur  
{ display: flex;  
flex-direction: row;  
flex-wrap: wrap;  
height:400px; width:400px;  
border: 1px solid black;}  
  
.element1 { background-color: orange;  
height:100px; width:40%;}  
.element2{ background-color: blue;  
height:100px; width:40%;}  
.element3{ background-color: green;  
height:100px; width:40%;}
```

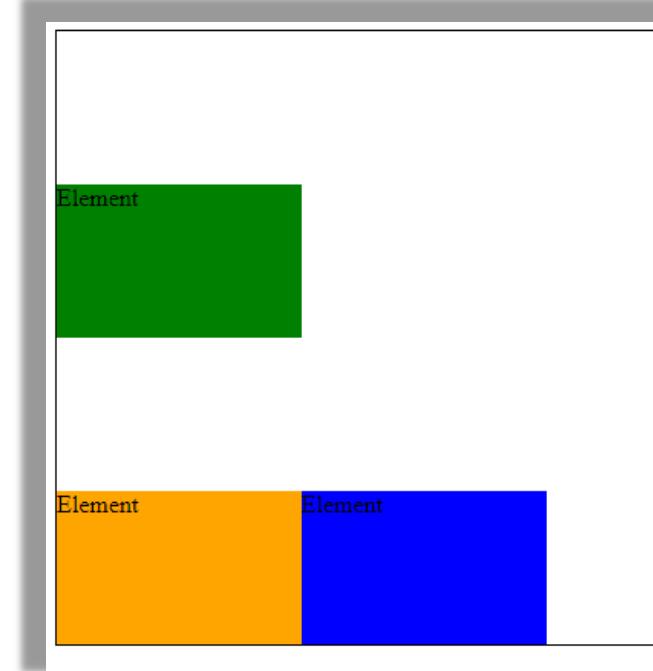


flex-wrap: wrap;
les éléments vont à la ligne lorsqu'il n'y a plus la place

Placement de boites avec Flexbox

- **Le retour à la ligne avec `flex-wrap`**
- `flex-wrap` qui peut prendre les valeurs suivantes :
 - `nowrap` : pas de retour à la ligne (par défaut) ;
 - `wrap` : les éléments vont à la ligne lorsqu'il n'y a plus la place ;
 - `wrap-reverse` : les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

```
/*avec des éléments qui dépassent  
Sa largeur 40% + 40% + 40% */  
#conteneur  
{ display: flex;  
flex-direction: row;  
flex-wrap: wrap-reverse;  
height:400px; width:400px;  
border: 1px solid black;}  
  
.element1 { background-color: orange;  
height:100px; width:40%;  
.element2{ background-color: blue;  
height:100px; width:40%;  
.element3{ background-color: green;  
height:100px; width:40%;
```



`flex-wrap: wrap-reverse;`

les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

Placement de boites avec Flexbox

- **Axe Principal, Axe Secondaire**
 - Avec **flex-direction** Les éléments sont organisés soit horizontalement (par défaut), soit verticalement. Cela définit ce qu'on appelle **l'axe principal**. Il y a aussi un axe secondaire (*cross axis*) :
 - Si les éléments sont organisés horizontalement, l'axe secondaire est l'axe vertical .
 - Si les éléments sont organisés verticalement, l'axe secondaire est l'axe horizontal.
- comment aligner nos éléments sur l'axe principal et sur l'axe secondaire?**
- alignement sur l'axe principal **justify-content**
 - et alignement sur l'axe secondaire **align-items**

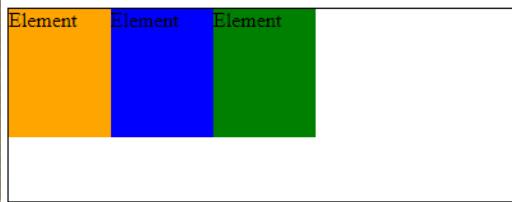
Placement de boites avec Flexbox

- **Alignment sur l'axe principal avec justify-content :** aligne les éléments du conteneur flexible lorsque les éléments n'utilisent pas tout l'espace disponible sur l'axe principal
- **Justify-content prend les valeurs suivantes:**
 - **flex-start** : alignés au début (par défaut) ;
 - **flex-end** : alignés à la fin ;
 - **center** : alignés au centre ;
 - **space-between** : les éléments sont étirés sur tout l'axe (il y a de l'espace entre eux) ;
 - **space-around** : idem, les éléments sont étirés sur tout l'axe, mais ils laissent aussi de l'espace sur les extrémités.

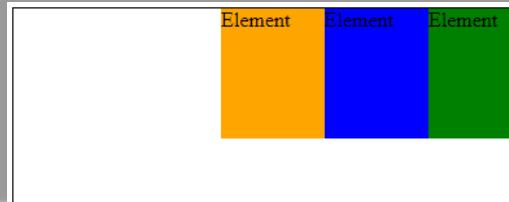
Placement de boites avec Flexbox

- **Alignment sur l'axe principal avec justify-content :**
- **Dans les exemples nous supposons que l'axe principal est l'axe horizontal (valeur par défaut qu'on peut changer avec flex-direction)**

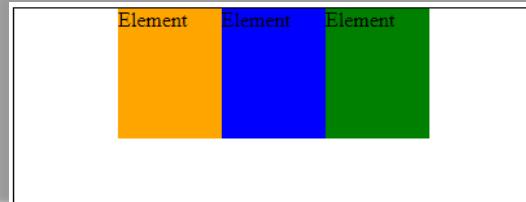
```
#conteneur
{
  display: flex;
  justify-content: flex-start;
  height: 150px; width:400px;
  border: 1px solid black;
}
.element1 {background-color: orange;
            height:100px; width:20%; }
.element2 { background-color: blue;
            height:100px; width:20%; }
.element3{ background-color: green;
            height:100px; width:20%; }
```



```
#conteneur
{
  display: flex;
  justify-content: flex-end;
  height:150px; width:400px;
  border: 1px solid black;
}
.element1 {background-color: orange;
            height:100px; width:20%; }
.element2 { background-color: blue;
            height:100px; width:20%; }
.element3{ background-color: green;
            height:100px; width:20%; }
```



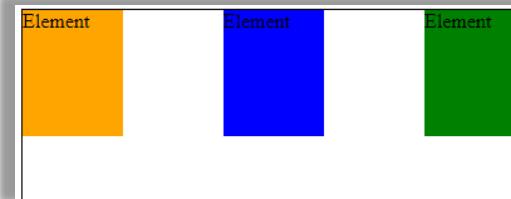
```
#conteneur
{
  display: flex;
  justify-content: center;
  height:150px; width:400px;
  border: 1px solid black;
}
.element1 {background-color: orange;
            height:100px; width:20%; }
.element2 { background-color: blue;
            height:100px; width:20%; }
.element3{ background-color: green;
            height:100px; width:20%; }
```



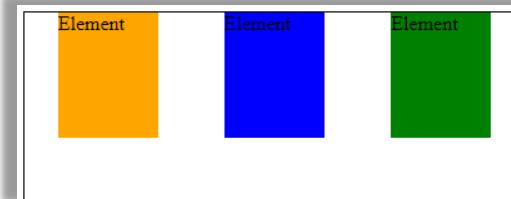
Placement de boites avec Flexbox

- **Alignment sur l'axe principal avec justify-content :**
- **Dans les exemples nous supposons que l'axe principal est l'axe horizontal (valeur par défaut qu'on peut changer avec flex-direction)**

```
#conteneur
{ display: flex;
justify-content: space-between;
height: 150px; width:400px;
border: 1px solid black;}
.element1 {background-color: orange;
height:100px; width:20%; }
.element2 { background-color: blue;
height:100px; width:20%; }
.element3{ background-color: green;
height:100px; width:20%; }
```



```
#conteneur
{ display: flex;
justify-content: space-around;
height: 150px; width:400px;
border: 1px solid black;}
.element1 {background-color: orange;
height:100px; width:20%; }
.element2 { background-color: blue;
height:100px; width:20%; }
.element3{ background-color: green;
height:100px; width:20%; }
```



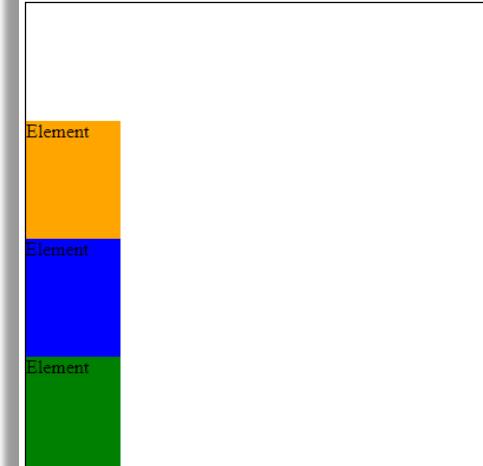
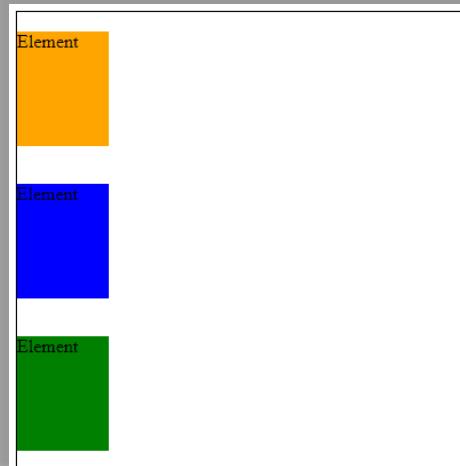
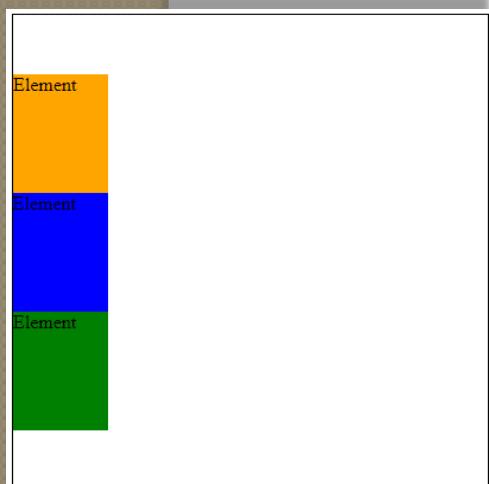
Placement de boîtes avec Flexbox

- **Alignment sur l'axe principal avec justify-content :**
- **ça marche aussi si les éléments sont dans une direction verticale.** Dans ce cas, l'axe vertical devient l'axe principal, et **justify-content** s'applique aussi :

```
#conteneur
{ display: flex;
  flex-direction: column;
  justify-content: center;
  height:400px; width:400px;
  border: 1px solid black;}
.element1 {background-color: orange;
           height:100px; width:20%; }
.element2 { background-color: blue;
            height:100px; width:20%; }
.element3{ background-color: green;
           height:100px; width:20%; }
```

```
#conteneur
{ display: flex;
  flex-direction: column;
  justify-content: space-around;
  height:400px; width:400px;
  border: 1px solid black;}
.element1 {background-color: orange;
           height:100px; width:20%; }
.element2 { background-color: blue;
            height:100px; width:20%; }
.element3{ background-color: green;
           height:100px; width:20%; }
```

```
#conteneur
{ display: flex;
  flex-direction: column;
  justify-content: flex-end;
  height:400px; width:400px;
  border: 1px solid black;}
.element1 {background-color: orange;
           height:100px; width:20%; }
.element2 { background-color: blue;
            height:100px; width:20%; }
.element3{ background-color: green;
           height:100px; width:20%; }
```



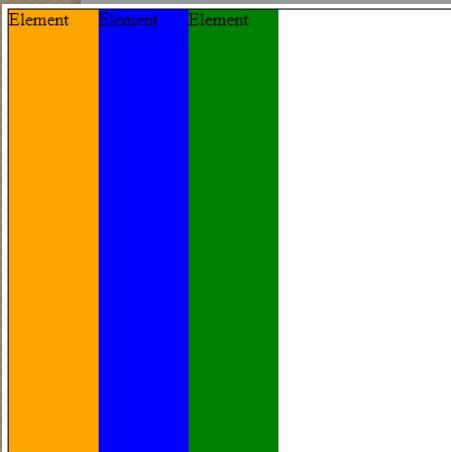
Placement de boites avec Flexbox

- **Alignement sur l'axe secondaire avec align-items:** aligne les éléments du conteneur flexible sur l'axe secondaire.
- peut prendre les valeurs suivantes:
 - **stretch** : les éléments sont étirés sur tout l'axe (valeur par défaut) ;
 - **flex-start** : alignés au début ;
 - **flex-end** : alignés à la fin ;
 - **center** : alignés au centre ;
 - **baseline** : alignés sur la ligne de base

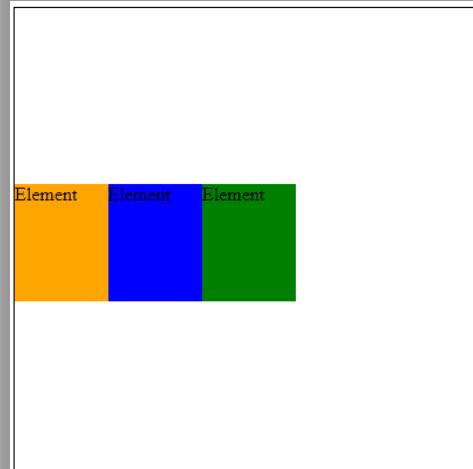
Placement de boites avec Flexbox

- **Alignement sur l'axe secondaire avec align-items:** aligne les éléments du conteneur flexible sur l'axe secondaire.
- **Dans les exemples nous supposons que l'axe secondaire est l'axe vertical (valeur par défaut qu'on peut changer avec flex-direction)**

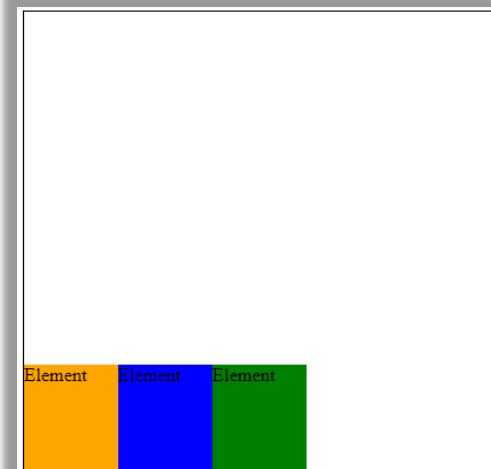
```
#conteneur
{
  display: flex;
  align-items: stretch;
  height:400px; width:400px;
  border: 1px solid black;}
.element1{ background-color: orange;
  width:20%;}
.element2{ background-color: blue;
  width:20%; }
.element3 { background-color: green;
  width:20%; }
/*ne pas préciser l'hauteur des elements*/
```



```
#conteneur
{
  display: flex;
  align-items: center;
  height:400px; width:400px;
  border: 1px solid black;}
.element1 { background-color: orange;
  height:100px; width:20%; }
.element2 { background-color: blue;
  height:100px; width:20%; }
.element3 {background-color: green;
  height:100px; width:20%;}
```



```
#conteneur
{
  display: flex;
  align-items: flex-end;
  height:400px; width:400px;
  border: 1px solid black;}
.element1 { background-color: orange;
  height:100px; width:20%; }
.element2 { background-color: blue;
  height:100px; width:20%; }
.element3 {background-color: green;
  height:100px; width:20%;}
```

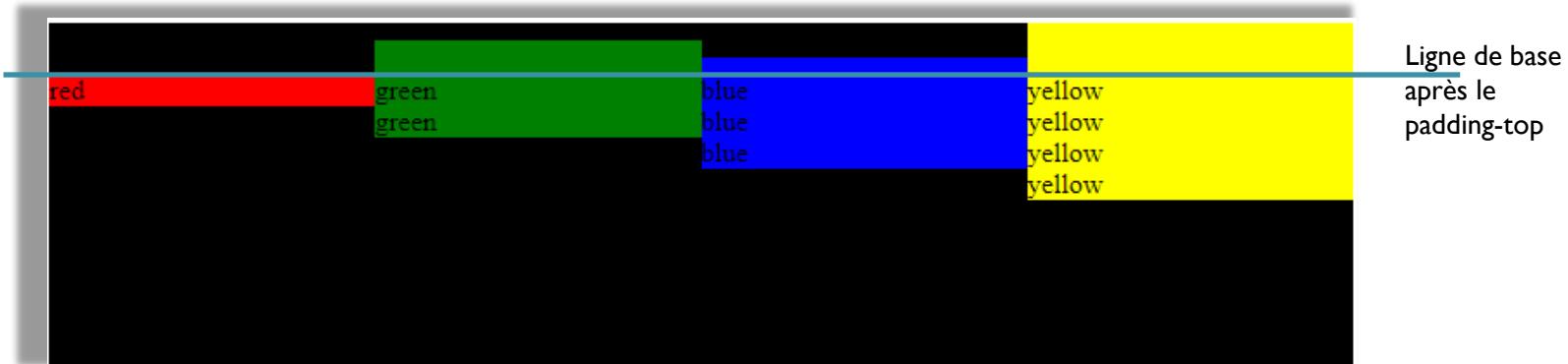


Placement de boîtes avec Flexbox

- **Alignement sur l'axe secondaire avec align-items:** aligne les éléments du conteneur flexible sur l'axe secondaire.
- **align-items: baseline;** aligne les éléments sur une ligne de base
- **Pour pouvoir tester la valeur baseline, on ajoute un padding-top pour quelques éléments**

```
<body>
<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>
</body>
```

```
.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%; }
```

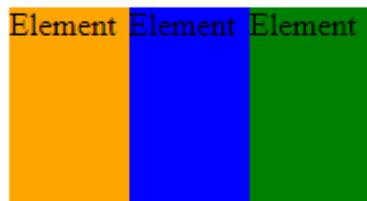


Placement de boites avec Flexbox

- Exemple d'alignement sur les deux axes

```
<div id="conteneur">  
  <div class="element 1">Element </div>  
  <div class="element 2">Element </div>  
  <div class="element 3">Element </div>  
</div>
```

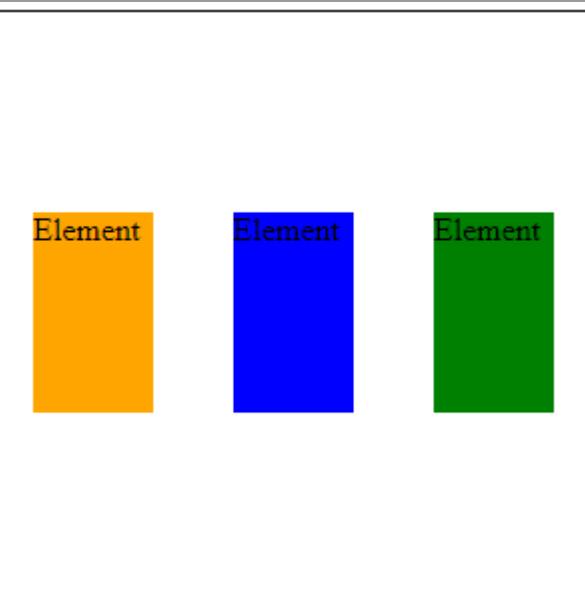
```
#conteneur  
{  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height:300px; width:300px;  
  border:solid 1px black;  
}  
.element1 { background-color: orange;  
            height:100px; width:20%; }  
.element2 { background-color: blue;  
            height:100px; width:20%; }  
.element3 {background-color: green;  
           height:100px; width:20%; }
```



Placement de boites avec Flexbox

- **Alignement sur les deux axes avec margin:auto**
- le centrage vertical et horizontal peut aussi être obtenu encore plus facilement. Il faut préciser pour le conteneur qu'il s'affiche **display:flex;** ensuite établir des marges automatiques avec **margin:auto;** sur les éléments à l'intérieur.

```
<div id="conteneur">  
  <div class="element 1">Element </div>  
  <div class="element 2">Element </div>  
  <div class="element 3">Element </div>  
</div>
```

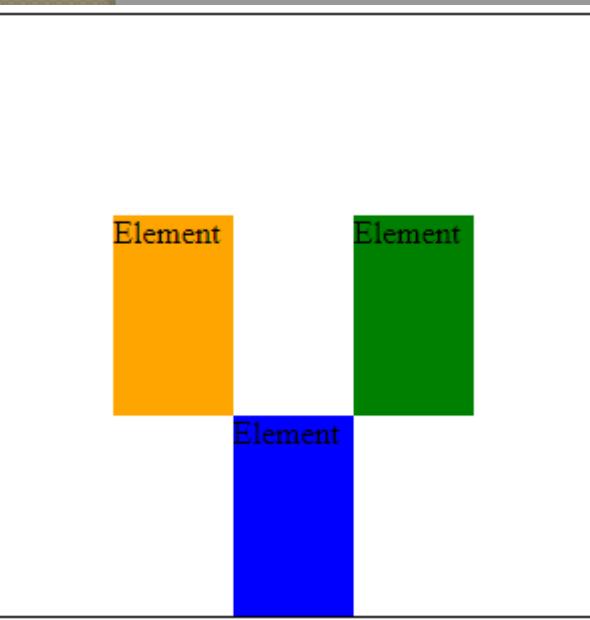


```
#conteneur  
{  
  display: flex;  
  height:300px; width:300px;  
  border:solid 1px black;  
}  
#conteneur>div { margin:auto; }  
  
.element1 { background-color: orange;  
            height:100px; width:20%; }  
.element2 { background-color: blue;  
            height:100px; width:20%; }  
.element3 { background-color: green;  
            height:100px; width:20%; }
```

Placement de boites avec Flexbox

- **Aligner un seul élément :** Il est possible de faire une exception pour un seul des éléments sur l'axe secondaire avec align-self :

```
<div id="conteneur">  
  <div class="element 1">Element </div>  
  <div class="element 2">Element </div>  
  <div class="element 3">Element </div>  
</div>
```

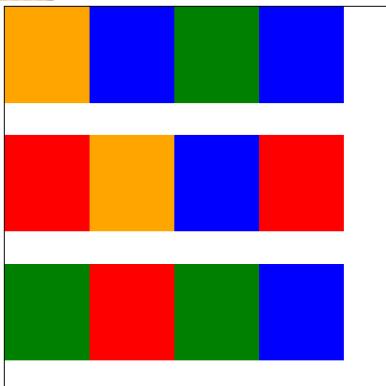


```
#conteneur  
{  
  display: flex;  
  flex-direction: row;  
  justify-content: center;  
  align-items: center;  
  height:300px; width:300px;  
  border:solid 1px black;  
}  
  
.element1 { background-color: orange;  
            height:100px; width:20%; }  
.element2 { background-color: blue;  
            align-self: flex-end;  
            height:100px; width:20%; }  
.element3 {background-color: green;  
           height:100px; width:20%;}  
/* L'élément 2 a fait l'exception de faire l'alignement  
flex-end */
```

Placement de boites avec Flexbox

- **Aligner plusieurs lignes :** Si on a plusieurs lignes dans le Flexbox, on peut choisir comment celles-ci seront réparties avec align-content

```
<div id="conteneur">
  <div class="element1"></div>
  <div class="element2"></div>
  <div class="element3"></div>
  <div class="element4"></div>
  <div class="element5"></div>
  <div class="element6"></div>
  <div class="element7"></div>
  <div class="element8"></div>
  <div class="element9"></div>
  <div class="element10"></div>
  <div class="element11"></div>
  <div class="element12"></div>
</div>
```



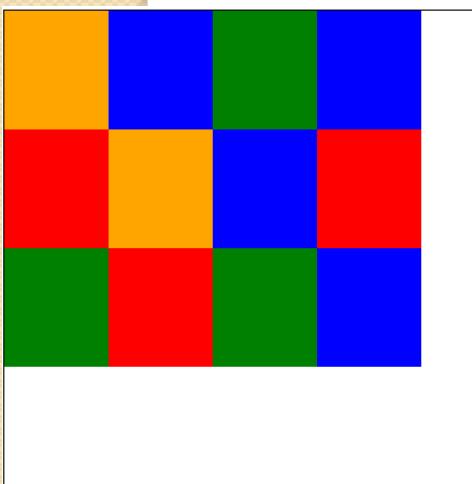
```
#conteneur
{ display: flex;
  flex-wrap: wrap;
  height:400px; width:400px;
  border:solid 1px black; }

.element1 { background-color: orange; height:100px; width:22%; }
.element2 { background-color: blue;   height:100px; width:22%; }
.element3 {background-color: green; height:100px; width:22%; }
.element4 { background-color: blue; height:100px; width:22%; }
.element5 { background-color: red;  height:100px; width:22%; }
.element6 {background-color: orange; height:100px; width:22%; }
.element7 { background-color: blue; height:100px; width:22%; }
.element8 { background-color: red; height:100px; width:22%; }
.element9 {background-color: green ; height:100px; width:22%; }
.element10 { background-color: orange; height:100px; width:22%; }
.element11 { background-color: green; height:100px; width:22%; }
.element12 {background-color: blue; height:100px; width:22%; }
```

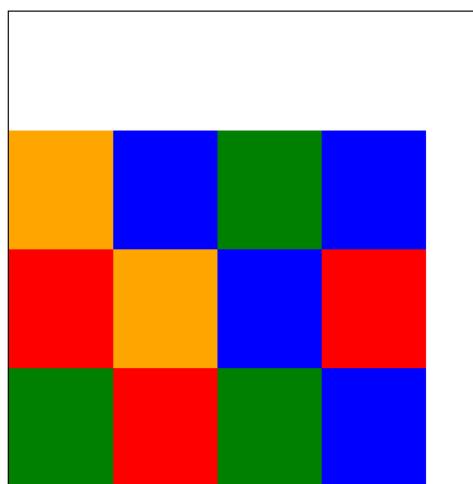
Placement de boîtes avec Flexbox

- **Aligner plusieurs lignes :** Si on a plusieurs lignes dans le Flexbox, on peut choisir comment celles-ci seront réparties avec align-content

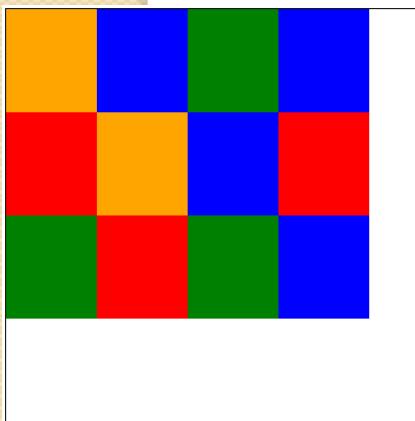
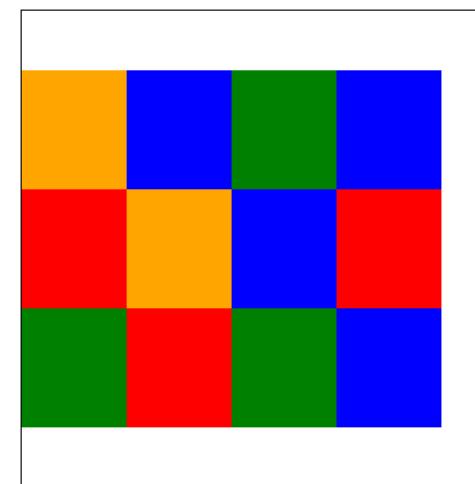
align-content: flex-start;



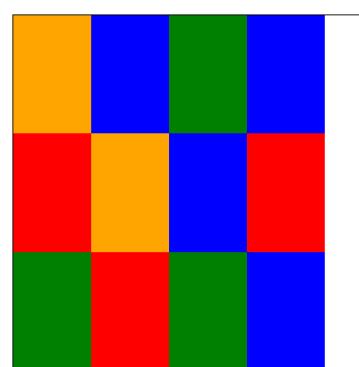
align-content: flex-end;



align-content: center;



align-content: baseline;

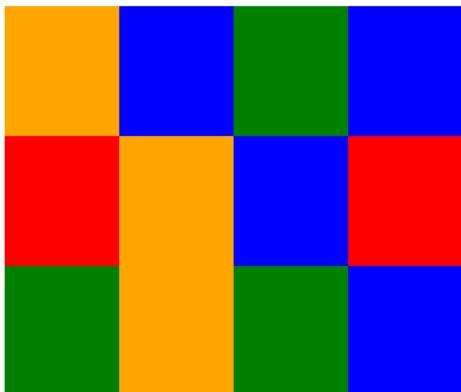


align-content: stretch; après avoir supprimé l'attribut height des éléments du conteneur

Placement de boîtes avec Flexbox

- Exemple de centrage sur les deux axes

```
<div id="conteneur">
  <div class="element1"></div>
  <div class="element2"></div>
  <div class="element3"></div>
  <div class="element4"></div>
  <div class="element5"></div>
  <div class="element6"></div>
  <div class="element7"></div>
  <div class="element8"></div>
  <div class="element9"></div>
  <div class="element10"></div>
  <div class="element11"></div>
  <div class="element12"></div>
</div>
```



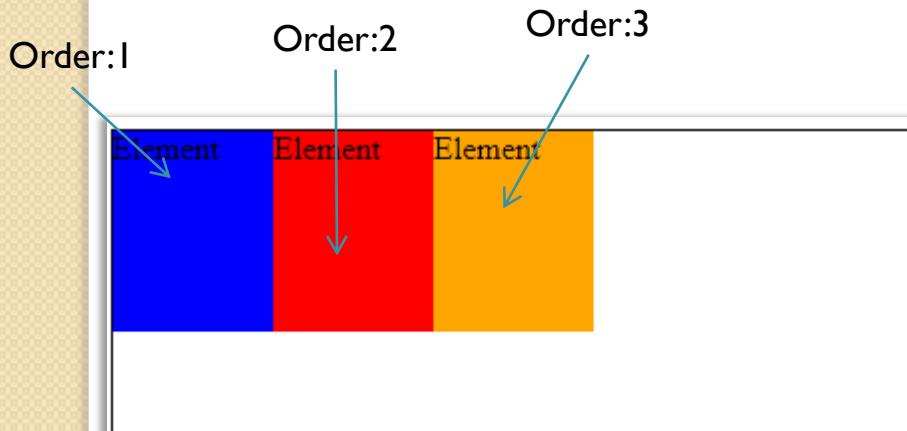
```
#conteneur
{
  display: flex;
  flex-wrap: wrap;
  height:400px; width:400px;
  border:solid 1px black;
  align-content: center;
  justify-content: center;
}

.element1 { background-color: orange; height:100px; width:22%; }
.element2 { background-color: blue; height:100px; width:22%; }
.element3 { background-color: green; height:100px; width:22%; }
.element4 { background-color: blue; height:100px; width:22%; }
.element5 { background-color: red; height:100px; width:22%; }
.element6 { background-color: orange; height:100px; width:22%; }
.element7 { background-color: blue; height:100px; width:22%; }
.element8 { background-color: red; height:100px; width:22%; }
.element9 { background-color: green; height:100px; width:22%; }
.element10 { background-color: orange; height:100px; width:22%; }
.element11 { background-color: green; height:100px; width:22%; }
.element12 { background-color: blue; height:100px; width:22%; }
```

Placement de boites avec Flexbox

- **La propriété order:** Sans changer le code HTML, nous pouvons modifier l'ordre des éléments en CSS grâce à la propriété order . Indiquer simplement un nombre, et les éléments seront triés du plus petit au plus grand nombre.
- Exemple

```
<body>  
  
    <div id="conteneur">  
        <div class="element1">Element </div>  
        <div class="element2">Element </div>  
        <div class="element3">Element </div>  
    </div>  
  
</body>
```

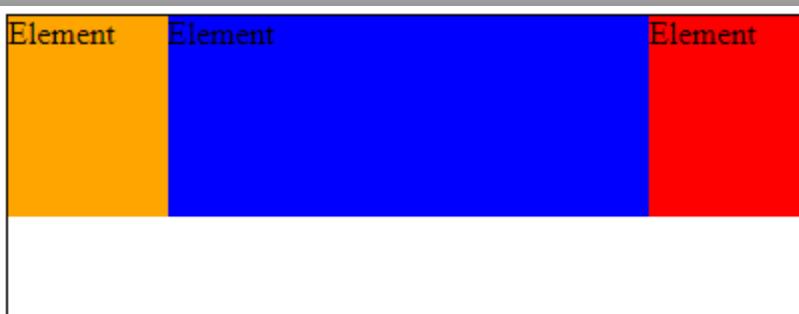


```
#conteneur  
{  
    display: flex;  
    height: 150px; width: 400px;  
    border: solid 1px black;  
}  
  
.element1  
{  
    background-color: orange;  
    height: 100px; width: 20%;  
    order: 3;  
}  
.element2  
{  
    background-color: blue;  
    height: 100px; width: 20%;  
    order: 1;  
}  
.element3  
{  
    background-color: red;  
    height: 100px; width: 20%;  
    order: 2;  
}
```

Placement de boîtes avec Flexbox

- **La propriété flex:** définit la longueur flexible sur les éléments flexibles.
- Exemple I

```
<body>  
  
    <div id="conteneur">  
        <div class="element1">Element </div>  
        <div class="element2">Element </div>  
        <div class="element3">Element </div>  
    </div>  
  
</body>
```

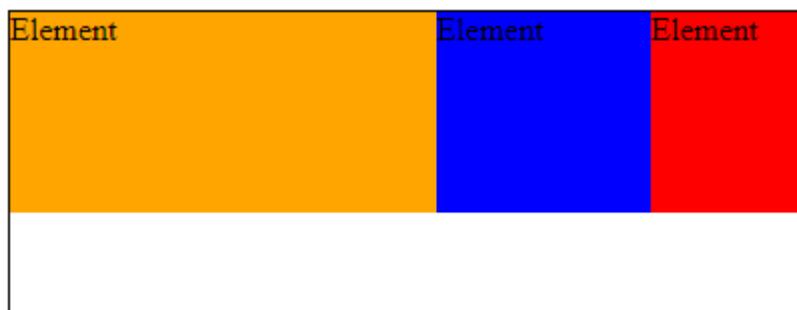


```
#conteneur  
{  
    display: flex;  
    height:150px; width:400px;  
    border:solid 1px black;  
}  
  
.element1  
{  
    background-color: orange;  
    height:100px; width:20%;  
}  
  
.element2  
{  
    background-color: blue;  
    width:100px; height:100px;  
    flex:1;  
/* L'élément2 s'étire pour occuper tout  
l'espace restant */  
}  
  
.element3  
{  
    background-color: red;  
    height:100px; width:20%;  
}
```

Placement de boites avec Flexbox

- **La propriété flex:** définit la longueur flexible sur les éléments flexibles.
- Exemple 2

```
<body>  
  
    <div id="conteneur">  
        <div class="element1">Element </div>  
        <div class="element2">Element </div>  
        <div class="element3">Element </div>  
    </div>  
  
</body>
```



Le premier élément peut grossir deux fois plus que le deuxième élément

```
#conteneur  
{  
    display: flex;  
    height:150px; width:400px;  
    border:solid 1px black;  
}  
  
.element1  
{  
    background-color: orange;  
    height:100px;  
flex:2;  
}  
  
.element2  
{  
    background-color: blue;  
    height:100px;  
flex:1;  
}  
  
.element3  
{  
    background-color: red;  
    height:100px; width:20%;}
```

Placement de boites avec Flexbox

Pour mieux comprendre Flexbox

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

Placement de boites avec grid

- Le module [CSS Grid Layout](#) ajoute à CSS une grille à deux dimensions. Les grilles peuvent être utilisées pour agencer des pages entières ou de petits éléments d'interface.
- Une grille est un ensemble de lignes horizontales et verticales qui se croisent – les premières définissant les rangées, et les secondes les colonnes. Les éléments sont placés sur la grille en fonction de ces rangées et colonnes.
- La grille est une spécification puissante qui peut être combinée avec d'autres modules CSS tels que [flexbox](#). Le point de départ est le **conteneur**.

Placement de boîtes avec grid

```
<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>
```



```
.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;
}
```

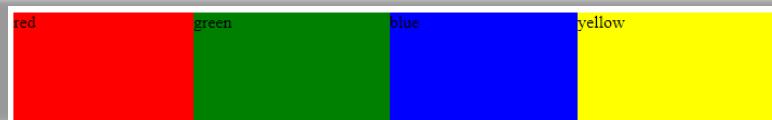
Tous les enfants directs sont maintenant des éléments de grille. On ne voit pas la différence dans un navigateur, car la grille n'a qu'une seule colonne.

Placement de boîtes avec grid

Les pistes: `grid-template`, `grid-template-columns`, `grid-template-rows`

Les propriétés `grid-template-columns` et `grid-template-rows` permettent de définir des colonnes et des rangées. Celles-ci définissent les pistes. Une *piste* est l'espace entre deux lignes d'une grille.

```
<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>
```



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

Placement de boîtes avec grid

Les pistes: `grid-template`, `grid-template-columns`, `grid-template-rows`

Exemple avec une grille composée de 3 colonnes

```
<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>
```



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

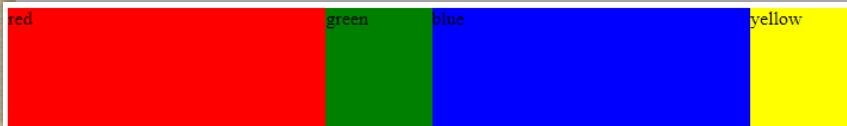
```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: grid;
  grid-template-columns: auto auto auto;
}
.container > div {
  height: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

Placement de boîtes avec grid

Les pistes: `grid-template`, `grid-template-columns`, `grid-template-rows`

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

```
<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>
```



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

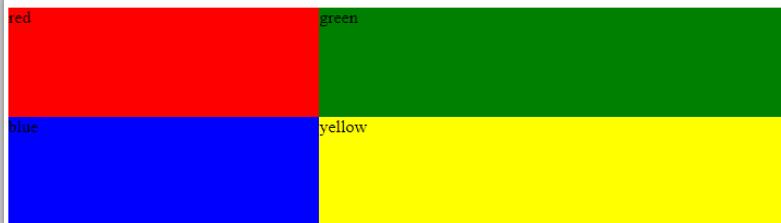
```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: grid;
  grid-template-columns: 1.5fr 0.5fr 1.5fr 0.5fr;
}
.container > div{
  height: 100px;
}
```

Placement de boîtes avec grid

Les pistes: grid-template, grid-template-columns, grid-template-rows

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

```
<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>
```



```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: grid;
  grid-template-columns: 40% 60%; // on peut aussi utiliser px : pixel ou fr : fraction
}
.container > div {
  height: 100px;
}
```

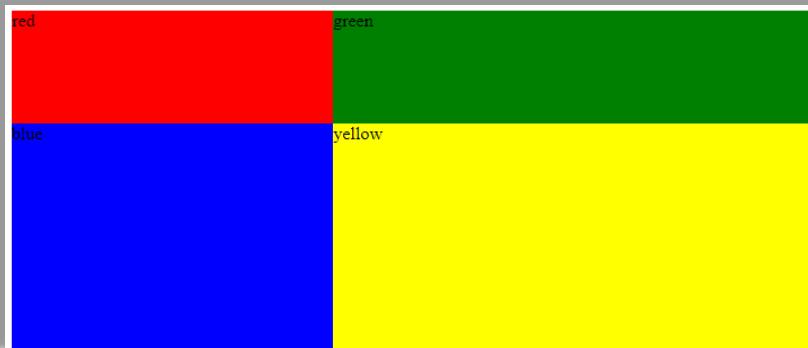
grid-template-columns: 40% 60%; // on peut aussi utiliser px : pixel ou fr : fraction

Placement de boîtes avec grid

Les pistes: `grid-template`, `grid-template-columns`, `grid-template-rows`

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

```
<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>
```



```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: grid;
  grid-template: 100px 200px / 40% 60%;
/*
  grid-template-columns: 40% 60%;
  grid-template-rows: 100px 200px;
*/
}
```

Placement de boites avec grid

Autres Propriétés: `grid-column-gap` , `grid-row-gap`, `justify-content`, `align-content`

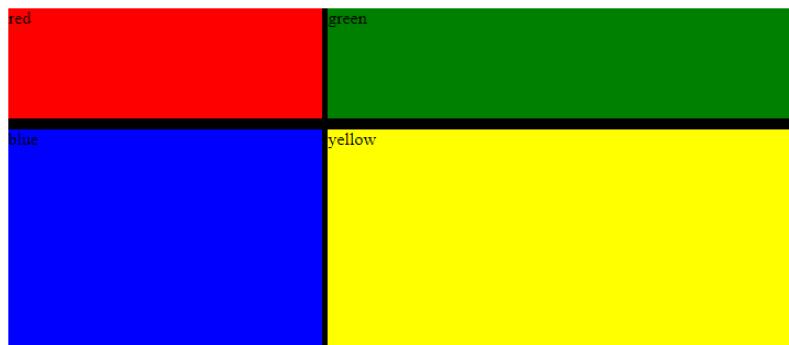
- **column-gap** : permet de définir un espace entre les colonnes
- **row-gap** : permet de définir un espace entre les lignes
- **justify-content** : permet de définir la façon dont l'espace doit être réparti entre et autour des composants par rapport à un axe horizontal
- **align-content** : permet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

Placement de boites avec grid

Autres Propriétés: `grid-column-gap`, `grid-row-gap`, `justify-content`, `align-content`

- `column-gap` : permet de définir un espace entre les colonnes
- `row-gap` : permet de définir un espace entre les lignes

```
<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>
```



```
.component1 {
  background-color: red;
}
.component2 {
  background-color: green;
}
.component3 {
  background-color: blue;
}
.component4 {
  background-color: yellow;
}
.container {
  background-color: black;
  display: grid;
  grid-template: 100px 200px / 40% 60%;
  column-gap: 5px;
  row-gap: 10px;
  /* ou bien      gap: 10px 5px
   * old grid-column-gap, grid-row-gap
}
```

Placement de boîtes avec grid

Autres Propriétés: justify-content, align-content

- Parfois, la taille totale de votre grille peut être inférieure à la taille de son conteneur de grille. Cela peut se produire si tous vos éléments de grille sont dimensionnés avec des unités non flexibles telles que px. Dans ce cas, vous pouvez définir l'alignement de la grille dans le conteneur de grille. La propriété justify-content aligne la grille le long de l'axe horizontal row par opposition à align-content qui aligne la grille le long de l'axe vertical (column)).
- **justify-content** : permet de définir la façon dont l'espace doit être réparti entre et autour des composants **par rapport à un axe horizontal**
- **align-content** : permet de définir la façon dont l'espace doit être réparti entre et autour des composants **par rapport à un axe vertical**

Placement de boîtes avec grid

Autres Propriétés: justify-content, align-content

```
<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 class=component5
      >magenta</div>
    <div class=component6 >purple</div>
    <div class=component7 >lime</div>
    <div class=component8 >navy</div>
    <div class=component9 >aqua</div>
  </div>
</body>
```

```
.component1 { background-color: red;}
.component2 {background-color: green;}
.component3 {background-color: blue;}
.component4 {background-color: yellow;}
.component5 {background-color: magenta;}
.component6 {background-color: purple;}
.component7 {background-color: lime;}
.component8 {background-color: navy;}
.component9 {background-color: aqua;}
```



```
.container {
  background-color: black;
  display: grid;
  width: 500px;
  height: 500px;
  grid-template: 100px 200px 100px /
    100px 100px 100px;
  justify-content :start; }
```

Placement de boites avec grid

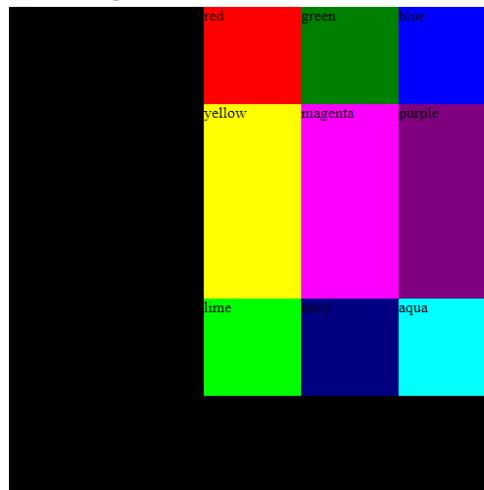
Propriétés pour le conteneur (Grid properties)

Autres Propriétés: justify-content, align-content

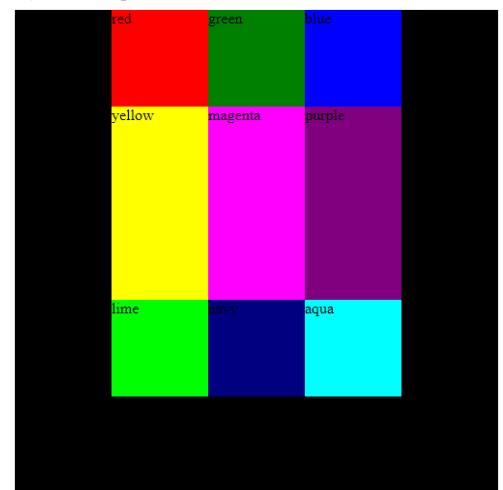
justify-content :start;



justify-content :end;

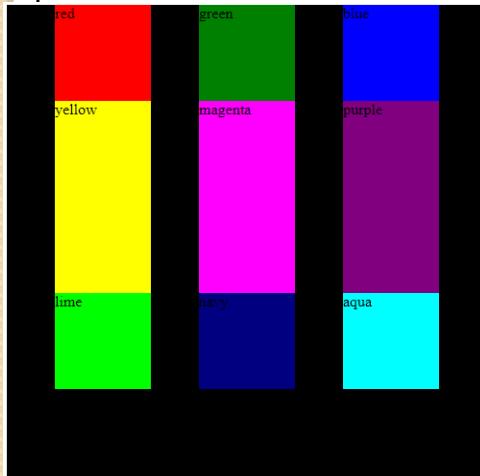


justify-content :center;



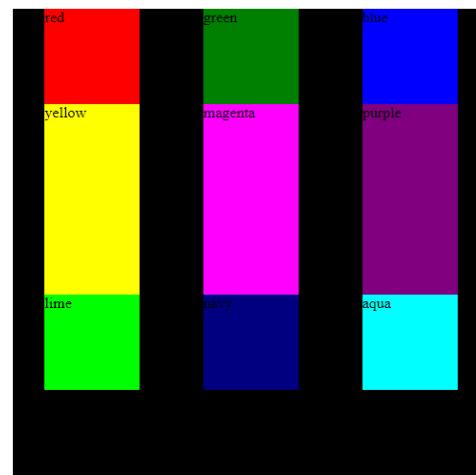
justify-content :space-around;

distribue l'espace disponible uniformément
autour de tous les éléments de la grille y
compris sur les cotés



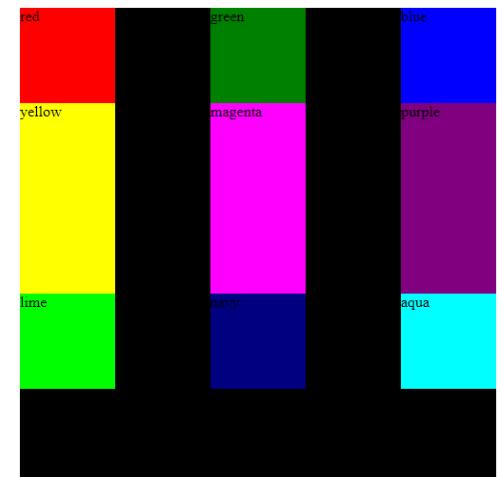
justify-content :space-around;

distribue l'espace disponible uniformément
autour de tous les éléments de la grille y
compris sur les cotés



justify-content:space-between;

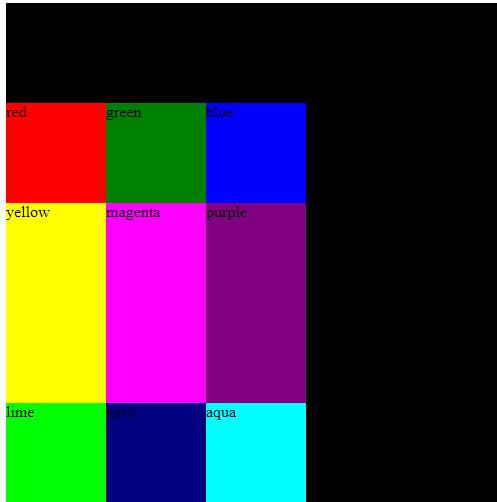
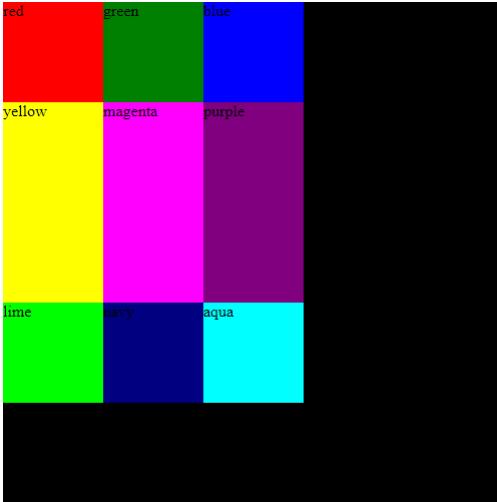
distribue l'espace disponible uniformément
entre les éléments de la grille y
compris sur les cotés



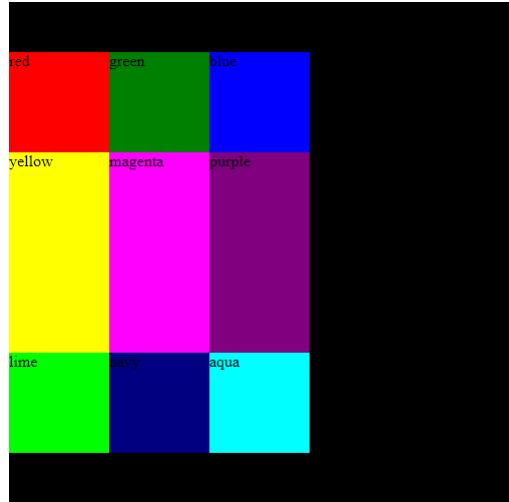
Placement de boites avec grid

Propriétés pour le conteneur (Grid properties)

Autres Propriétés: justify-content, align-content
align-content :start;

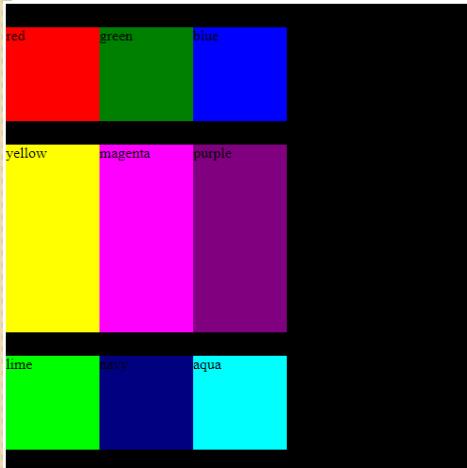


align-content :center;

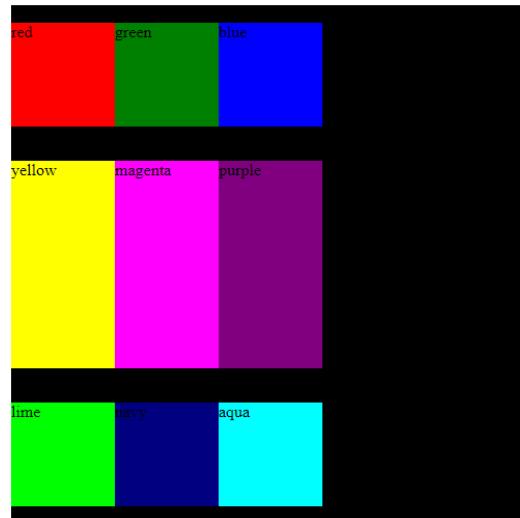


align-content :space-evenly;

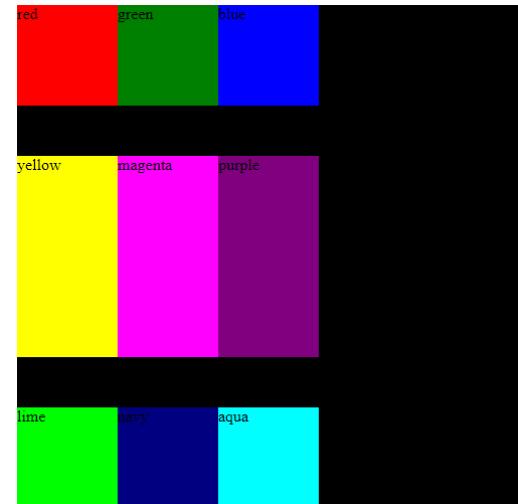
distribue l'espace disponible uniformément
autour de tous les éléments de la grille y
compris sur les cotés



align-content :space-around;

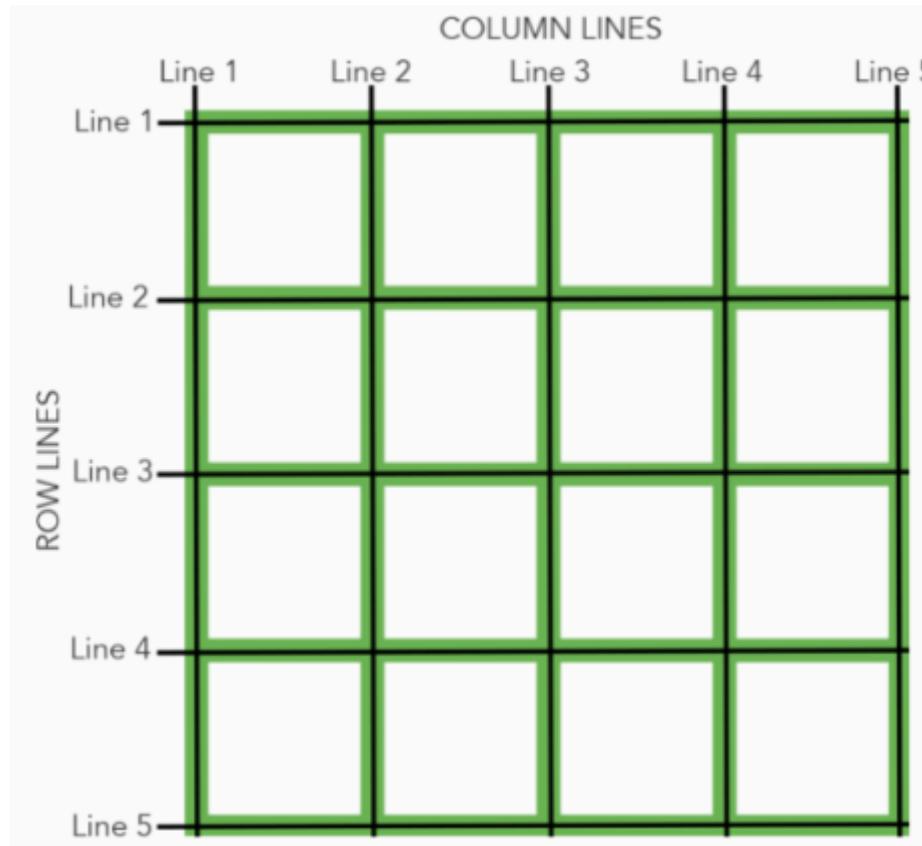


align-content:space-between;



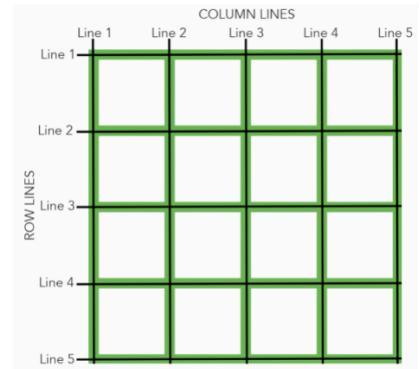
Placement de boîtes avec grid

- On peut aussi placer les composants en précisant pour chacun entre quelles ligne verticale et ligne horizontale il faut le placer



Placement de boites avec grid

- On peut aussi placer les composants en précisant pour chacun entre quelles ligne verticale et ligne horizontale il faut le placer
- **grid-column-start**
- **grid-column-end**
- **grid-row-start**
- **grid-row-end**



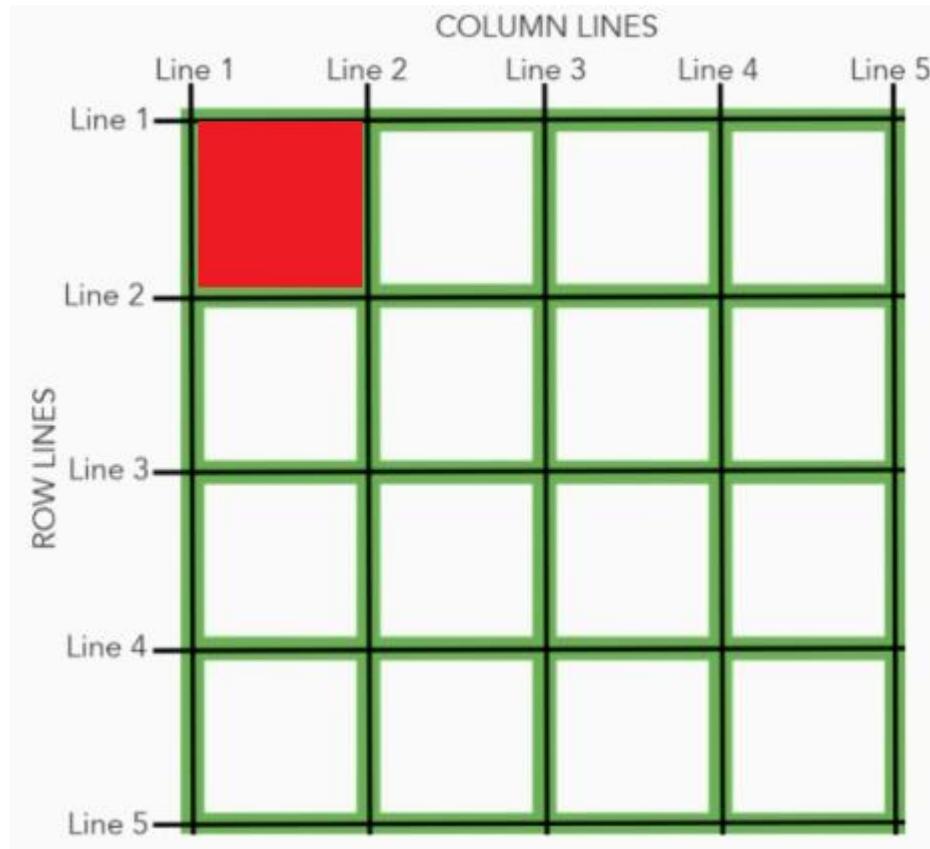
Détermine l'emplacement d'un élément de grille dans la grille en se référant à des lignes de grille spécifiques.

- **grid-column-start/grid-row-start** est la ligne où commence l'élément, et
- **grid-column-end/grid-row-end** est la ligne où se termine l'élément

Placement de boîtes avec grid

- **Exemple 1**

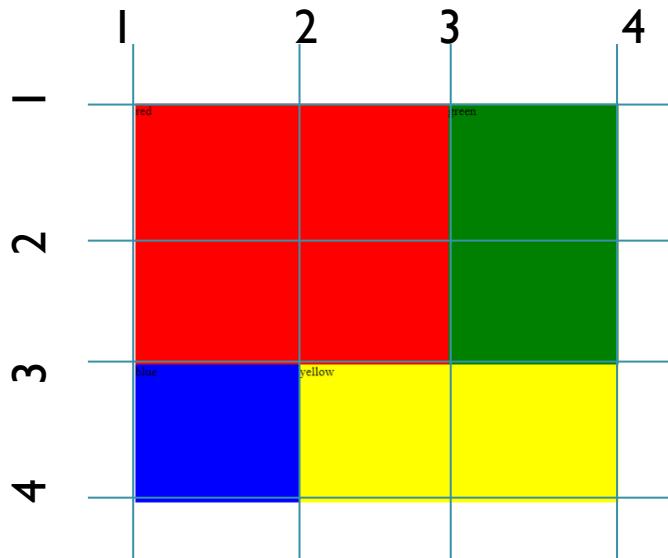
```
.component1 {  
    background-color: red;  
    grid-column-start: 1;  
    grid-column-end: 2;  
    grid-row-start: 1;  
    grid-row-end: 2;  
}
```



Placement de boîtes avec grid

- Exemple 1

```
<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>
```



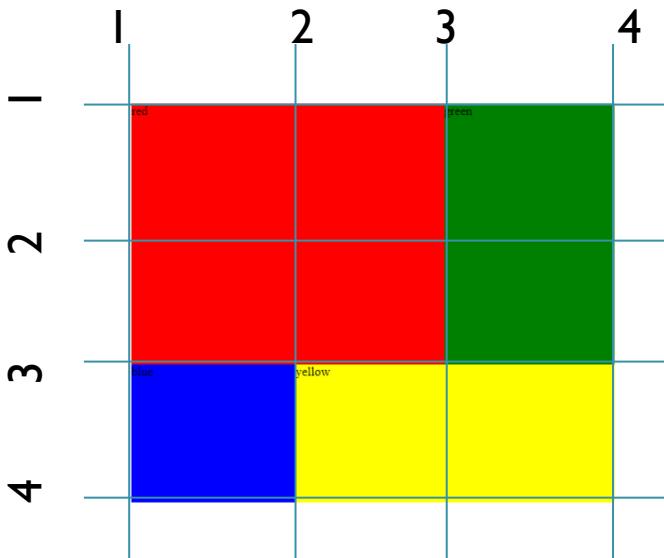
```
.component1 {  
background-color: red;  
grid-column-start: 1;  
grid-column-end: 3;  
grid-row-start: 1;  
grid-row-end: 3;  
}  
.component2 {  
background-color: green;  
grid-column-start: 3;  
grid-column-end: 4;  
grid-row-start: 1;  
grid-row-end: 3;  
}  
.component3 {  
background-color: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

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

Placement de boîtes avec grid

- On peut encore le simplifier

```
<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>
```



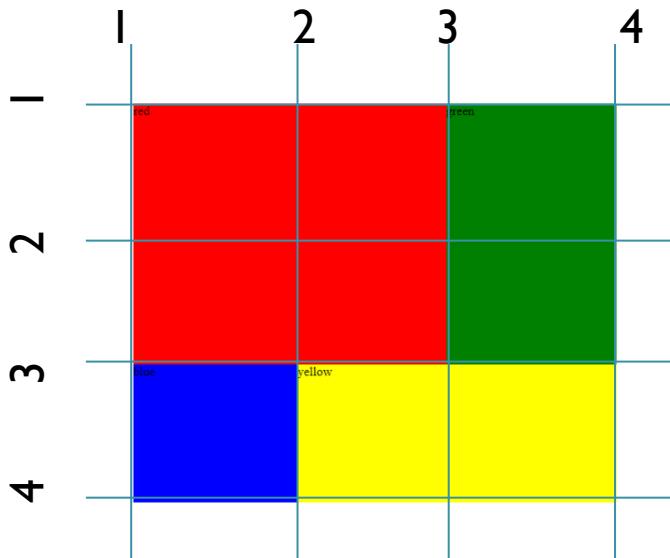
```
.component1 {  
background-color: red;  
grid-column: 1 / span 2;  
grid-row: 1 / span 2;  
}  
.component2 {  
background-color: green;  
grid-column: 3 / span 1;  
grid-row: 1 / span 2;  
}  
.component3 {  
background-color: blue;  
grid-column: 1 / span 1;  
grid-row: 3 / span 1;  
}  
.component4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

```
.container {  
background-color: black;  
display : grid;  
height: 500px;  
grid-template: repeat(3, 1fr) ;  
}
```

Placement de boîtes avec grid

- On peut aussi utiliser `grid-area: row-start / col-start / row-number / col-number`

```
<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>
```



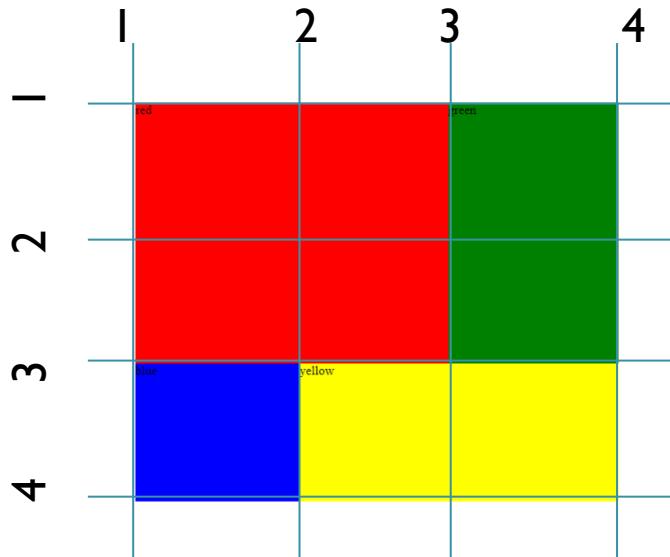
```
/*grid-area: row-start / col-start / row-number /
col-number*/
.component1 {
background-color: red;
grid-area: 1 / 1 / span 2 / span 2;
}
.component2 {
background-color: green;
grid-area: 1 / 3 / span 2 / span 1;
}
.component3 {
background-color: blue;
grid-area: 3 / 1 / span 1 / span 1;
}
.component4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

```
.container {
background-color: black;
display : grid;
height: 500px;
grid-template: repeat(3, 1fr);
}
```

Placement de boîtes avec grid

- On peut aussi utiliser `grid-area: row-start / col-start / row-number /col-number`

```
<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>
```



```
.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:'rouge rouge vert'
'rouge rouge vert'
'bleu  jaune jaune';
}
```

Placement de boites avec grid

Pour mieux comprendre Grid

- <http://cssgridgarden.com/#fr>
- <https://css-tricks.com/snippets/css/complete-guide-grid/>

Les Media Queries

- **Les media queries** permettent de modifier l'apparence d'un site ou d'une application en fonction du type d'appareil (impression ou écran par exemple) et de ses caractéristiques (la résolution d'écran ou la largeur de la zone d'affichage (*viewport*) par exemple)
- Grâce à cette technique, nous pourrons créer un design qui s'adapte automatiquement à l'écran de chaque visiteur ! (écran d'ordinateur , tablette, smart-phone,..)
- Concrètement, on va pouvoir dire « Si la résolution de l'écran du visiteur est inférieure à tant, alors applique les propriétés CSS suivantes sinon on applique d'autres propriétés ».
- Cela permet de changer l'apparence du site dans certaines conditions : on peut donc augmenter la taille du texte, changer la couleur de fond, positionner différemment votre contenu dans certaines résolutions, etc.
- On peut changer l'apparence de notre site en fonction d'autres critères que la résolution comme le type d'écran (smartphone, télévision, projecteur...), le nombre de couleurs, l'orientation de l'écran (portrait ou paysage), etc.

Les Media Queries

Mise en place des media queries

Les media queries sont donc des règles qui indiquent quand on doit appliquer des propriétés CSS. Il y a deux façons de les utiliser :

1. **en chargeant une feuille de style .css différente en fonction de la règle** (ex. : « Si la résolution est inférieure à 1 280 px de large, charge le fichier petite_resolution.css ») ;
2. **en écrivant la règle directement dans le fichier .css habituel** (ex. : « Si la résolution est inférieure à 1 280 px de large, charge les propriétés CSS ci-dessous »).

Les Media Queries

Mise en place des media queries

I. en chargeant une feuille de style .css différente en fonction de la règle

(ex. : « Si la résolution est inférieure à 1 280 px de large, charge le fichier petite_resolution.css ») ;

Le code HTML pourrait proposer plusieurs fichiers CSS : un par défaut (qui est chargé dans tous les cas) et un ou deux autres qui seront chargés en supplément, uniquement si la règle correspondante s'applique.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" /> <!-- Pour tout le monde --&gt;
    &lt;link rel="stylesheet" media="screen and (max-width: 1280px)" href="petite_resolution.css" /&gt;
    <!-- Pour ceux qui ont une résolution inférieure à 1280px --&gt;
  &lt;title&gt;Media queries&lt;/title&gt;
&lt;/head&gt;
&lt;/html&gt;</pre>
```

Note : Lorsqu'une feuille de style est attachée à un élément `<link>` comportant une requête média, cette feuille de style sera toujours téléchargée, même si la requête renvoie false. Toutefois, le contenu de cette feuille n'est pas appliquée tant que le résultat de la requête ne devient pas true.

Les Media Queries

Mise en place des media queries

2. **Chargement des règles directement dans la feuille de style** une deuxième technique plus pratique consiste à écrire ces règles dans le même fichier CSS que d'habitude. Dans ce cas, on écrit la règle dans le fichier .css comme ceci :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="styleMedia1.css" />
  <!-- Pour tout le monde -->
  <title>Media queries</title>
</head>
<body>
  <p> La taille de ce paragraphe se réajuste avec
  la résolution de l'écran </p>
</body>

</html>
```

```
/* fichier styleMedia1.css*/
/* style pour tous les écrans */
p {
  font-size: 22px;
  color: red;
}

/* style pour les écrans ne dépassant pas 800 pixels
de large */
@media screen and (max-width: 800px)
{
  p {
    font-size: 14px;
    color: blue;
  }
}
```

Les Media Queries

Mise en place des media queries

2. **Chargement des règles directement dans la feuille de style** une deuxième technique plus pratique consiste à écrire ces règles dans le même fichier CSS que d'habitude. Dans ce cas, on écrit la règle dans le fichier .css comme ceci :



Dès qu'on rétrécit la largeur de la fenêtre au dessous de 800 px le nouveau style du paragraphe s'applique



```
/* fichier styleMedia1.css*/
/* style pour tous les écrans */
p {
    font-size: 22px;
    color: red;
}

/* style pour les écrans ne dépassant pas 800 pixels
de large */
@media screen and (max-width: 800px)
{
    p {
        font-size: 14px;
        color: blue;
    }
}
```

Les Media Queries

Syntaxe

Syntaxe Une requête média se compose d'un **type de média** optionnel et **d'une ou plusieurs expressions de caractéristiques de média**.

Plusieurs requêtes peuvent être combinées entre elles grâce à des opérateurs logiques. Les requêtes média ne sont pas sensibles à la casse.

Types de média: Un type de média définit une catégorie générale d'appareil. Le type de média est optionnel dans une requête média, sauf si celle-ci utilise les opérateurs logiques `not` ou `only`. Par défaut, le type de média utilisé est `all`.

Voici les principaux types de média

- **all** Correspond pour tous les appareils.
- **print** Correspond aux documents consultés en aperçu avant impression.
- **screen** Correspond aux appareils dotés d'un écran.
- **handheld** : périphérique mobile, (Plusieurs navigateurs mobiles ignorent le media handheld (par exemple Safari Mobile) et se considèrent comme un média screen.)
- **projection** : projecteur,

Les Media Queries

Syntaxe

Syntaxe Une requête média se compose d'un **type de média** optionnel et **d'une ou plusieurs expressions de caractéristiques de média**.

Plusieurs requêtes peuvent être combinées entre elles grâce à des opérateurs logiques.
Les requêtes média ne sont pas sensibles à la casse.

Caractéristiques média (media features): décrivent certaines caractéristiques spécifiques de l'appareil d'affichage ou de l'environnement. Les expressions de caractéristique média testent leur présence ou leur valeur. **Chaque expression de caractéristique doit être entourée de parenthèses**

Les Media Queries

Syntaxe

Principales Caractéristiques média (media features):

media feature	description	exemple
height	hauteur de la zone d'affichage (viewport)	@media screen and (height: 600px) { }
width	largeur de la zone d'affichage	@media screen and (width: 600px) { }
color (max,min)	Le nombre de bits par composante de couleur pour le périphérique de sortie	@media (color:32) {} /* Tout appareil qui gère des couleurs avec 32 bits par composante */
resolution (max, min)	résolution du périphérique	@media (resolution: 150dpi) { p { color: red; } }
max-height	hauteur maximale de la zone d'affichage	@media screen and (max-height: 600px) { }
max-width	largeur maximale de la zone d'affichage	@media screen and (max-width: 600px) { }
min-height	hauteur minimale de la zone d'affichag	@media screen and (min-height: 600px) { }
min-width	largeur minimale de la zone d'affichage	@media screen and (min-width 600px) { }
orientation	orientation du périphérique(portrait ou paysage)	@media (orientation: landscape) {body {flex-direction: row;}} @media (orientation: portrait) { body { flex-direction: column; } }

Les Media Queries

Syntaxe

Les règles peuvent être combinées à l'aide des mots suivants :

- only : uniquement
- and : et
- not : non

```
/* Sur les écrans, quand la largeur de la fenêtre fait au maximum 1280px */  
@media screen and (max-width: 1280px)  
/* Sur tous types d'écran, quand la largeur de la fenêtre est comprise entre 1024px et  
1280px */  
@media all and (min-width: 1024px) and (max-width: 1280px)  
/* Sur écran Tv ou projecteur */  
@media projection and tv {}  
/* Sur tous types d'écrans orientés verticalement */  
@media all and (orientation: portrait)
```

Les animations CSS

- **La Propriété transform:**
- La propriété **transform** applique une transformation 2D ou 3D à un élément. Cette propriété vous permet de faire pivoter, redimensionner, déplacer, incliner, etc. des éléments.

Syntaxe:

`transform: none|transform-functions;`

Les animations CSS

- **La Propriété transform:**

fonction	
translate(x,y)	Defines a 2D translation
translate3d(x,y,z)	Defines a 3D translation
translateX(x)	Defines a translation, using only the value for the X-axis
translateY(y)	Defines a translation, using only the value for the Y-axis
translateZ(z)	Defines a 3D translation, using only the value for the Z-axis
scale(x,y)	Defines a 2D scale transformation
scale3d(x,y,z)	Defines a 3D scale transformation
scaleX(x)	Defines a scale transformation by giving a value for the X-axis
scaleY(y)	Defines a scale transformation by giving a value for the Y-axis
scaleZ(z)	Defines a 3D scale transformation by giving a value for the Z-axis
rotate(angle)	Defines a 2D rotation, the angle is specified in the parameter
rotate3d(x,y,z,angle)	Defines a 3D rotation

Les animations CSS

- **La Propriété transform:**

rotateX(<i>angle</i>)	Defines a 3D rotation along the X-axis
rotateY(<i>angle</i>)	Defines a 3D rotation along the Y-axis
rotateZ(<i>angle</i>)	Defines a 3D rotation along the Z-axis
skew(<i>x-angle,y-angle</i>)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(<i>angle</i>)	Defines a 2D skew transformation along the X-axis
skewY(<i>angle</i>)	Defines a 2D skew transformation along the Y-axis

Les animations CSS

- **La Propriété Transition:**
- Les transitions CSS vous permettent de modifier les valeurs des propriétés en douceur, sur une durée donnée.

Comment utiliser les transitions CSS ?

Pour créer un effet de transition, vous devez spécifier deux choses :

1. la propriété CSS à laquelle vous souhaitez ajouter un effet
2. la durée de l'effet

Les animations CSS

- **Exemple sur la Propriété Transition:**

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
  <h1>Transition + Transform</h1>
  <p>Hover over the div element below:</p>
  <div></div>
</body>
</html>
```

```
<style>

div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s, height 2s, transform 2s;
}

div:hover {
  width: 300px;
  height: 300px;
  transform: rotate(180deg);
}

</style>
```

La transition de la largeur (de 100 à 300px) prendra 2 secondes
La transition de la hauteur(de 100 à 300px) prendra 2 secondes
La transformation de rotation prendra 2 secondes

Les animations CSS

- **On peut aussi Spécifier la courbe de vitesse de la transition**

La propriété **transition-timing-function** peut avoir les valeurs suivantes

- **ease** : spécifie un effet de transition avec un démarrage lent, puis rapide, puis une fin lente (c'est la valeur par défaut)
- **linear** : spécifie un effet de transition avec la même vitesse du début à la fin
- **À voir aussi ease-in, ease-out, ease-in-out..**

- On peut aussi Spécifier un délais avant l'exécution de la transition
- La propriété **transition-delay** spécifie un délai (en secondes) pour l'effet de transition.

- **On peut combiner ces 4 propriétés en une seule**

Exemple

- Transition height 2s ease-in-out 1s; la transition sur la propriété height s'effectue en 2 secondes en utilisant la fonction de transition ease-in-out et elle effectue un délai d'attente de 1s avant de commencer l'animation

Les animations CSS

- **Animations avec @keyframes**
- La règle @keyframes spécifie le code d'animation.
- L'animation est créée en passant progressivement d'un ensemble de styles CSS à un autre, pendant l'animation,
- vous pouvez modifier plusieurs fois l'ensemble des styles CSS.
- Spécifiez quand le changement de style se produira en pourcentage, ou avec les mots-clés « from » et « to », qui sont identiques à 0 % et 100 %. 0 % correspond au début de l'animation, 100 % correspond à la fin de l'animation.

- **Exemple 1**

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Animation keyframes </h1>
<div>
  
</div>
</body>
</html>
```

```
<style>

@keyframes anim1 {
  from{
    transform:translateX(300px);
    opacity:0;
  }
  To{
    transform:translateX(0px);
    opacity:1;
  }
}
.image{
  animation:anim1 5s;
}

</style>
```

Les animations CSS

- **Animations avec @keyframes**
- La règle @keyframes spécifie le code d'animation.
- L'animation est créée en passant progressivement d'un ensemble de styles CSS à un autre, pendant l'animation,
- vous pouvez modifier plusieurs fois l'ensemble des styles CSS.
- Spécifiez quand le changement de style se produira en pourcentage, ou avec les mots-clés « from » et « to », qui sont identiques à 0 % et 100 %. 0 % correspond au début de l'animation, 100 % correspond à la fin de l'animation
- **Exemple 2 pourcentage %**

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Animation keyframes </h1>
<div>
  
</div>
</body>
</html>
```

```
<style>
@keyframes anim1 {
  0%{
    transform:translateX(300px);
    opacity:0;
  }
  50%{
    transform:translateX(0px);
    opacity:1;
  }
  100%{
    transform:translateX(300px);
    opacity:0;
  }
}

.image{
  animation:anim1 5s;
} </style>
```

Les animations CSS

- Animations avec `@keyframes`
- Si on veut que l'animation boucle à l'infini!
- `animation-iteration-count: infinite;`
- On pourra ajouter infinite à la propriété `animation`

- Exemple 2 pourcentage %

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Animation keyframes </h1>
<div>
  
</div>
</body>
</html>
```

```
<style>
@keyframes anim1 {
  0%{
    transform:translateX(300px);
    opacity:0;
  }
  50%{
    transform:translateX(0px);
    opacity:1;
  }
  100%{
    transform:translateX(300px);
    opacity:0;
  }
}
.image{
  animation:anim1 5s infinite;
}
</style>
```