

# III- LE LANGUAGE CSS (Cascading Style Sheets)



# SITE WEB STATIQUE

## **LE LANGAGE CSS**

1. **INTRODUCTION**
2. **APPLIQUER UN STYLE:SELECTIONNER UNE BALISE**
3. **LE SELECTEUR : CLASS ET ID**
4. **LES BALISES UNIVERSELLES**
5. **APPLIQUER UN STYLE/ SELECTEURS AVANCES**
6. **QUELQUES PROPRIETES POUR COMMENCER**

# **1- INTRODUCTION**

## **1- Introduction**

**Les feuilles de styles sont un langage qui permet de gérer la présentation d'une page Web.**

**Les styles permettent de définir des règles appliquées à un ou plusieurs documents HTML. Ces règles portent sur le positionnement des éléments, l'alignement, les polices de caractères, les couleurs, les marges et espacements, les bordures, les images de fond, etc.**

**L'utilisation de feuilles de styles permet de séparer la forme du fond ce qui présente de nombreux avantages. En particulier la conception et la maintenance des sites sont simplifiées.**

**A quoi sert CSS?**

**Où écrit-on le CSS ? (exemple)**

**Les commentaires en CSS**

## **A quoi sert CSS ?**

**C'est lui qui vous permet de choisir la couleur de votre texte, sélectionner la police utilisée sur votre site, et encore définir la taille du texte, les bordures, le fond. Et aussi, c'est lui qui permet de faire la mise en page de votre site. Vous pourrez dire :**

**je veux que mon menu soit à gauche et occupe telle largeur, que l'en-tête de mon site soit calé en haut et qu'il soit toujours visible, etc.**

## Où écrit-on le CSS ?

**Vous avez le choix car on peut écrire du code en langage CSS à trois endroits différents :**

- **directement dans les balises du fichier HTML via l'attribut style.**

**<p style =" color : blue ;">Bonjour et bienvenue sur mon site !</p>**

- **dans l'en-tête <head> du fichier HTML, via la balise <style>:**

```
<head >  
  <style >  
    p  
    {  
      color : blue ;  
    }  
  </ style >  
</head>
```

## Où écrit-on le CSS ?

- dans un fichier .css (méthode la plus recommandée)

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

La balise **<LINK>** avertit le navigateur qu'il doit chercher un document situé à l'extérieur de la page HTML.

L'attribut **rel="stylesheet"** précise que le document en question est une feuille de style externe.

L'attribut **type="text/css"** précise le type du fichier.

L'attribut **href=" URL "** donne l'URL de la feuille de style, c'est-à-dire son emplacement.

## Les commentaires en CSS

**Comme en HTML, il est possible de mettre des commentaires. Les commentaires ne seront pas affichés, ils servent simplement à indiquer des informations pour vous, par exemple pour vous y retrouver dans un looong fichier CSS.**

**Pour faire un commentaire, c'est facile ! Tapez /\*, suivi de votre commentaire, puis \*/ pour terminer votre commentaire. Vos commentaires peuvent être écrits sur une ou plusieurs lignes.**



## **2- APPLIQUER UN STYLE : SÉLECTIONNER UNE BALISE**

**Dans un code CSS on trouve trois éléments différents :**

```
Selecteur{  
    propriete :valeur ;  
}
```

**Des noms de balises(sélecteur)** : il peut être une balise dont on veut modifier l'apparence, Par exemple, si je veux modifier l'apparence de tous les paragraphes <p>, je dois écrire p.

**Des propriétés CSS** : les « effets de style » de la page sont rangés dans des propriétés. Il y a par exemple la propriété color qui permet d'indiquer la couleur du texte, font-size qui permet d'indiquer la taille du texte, etc.

**Les valeurs** : pour chaque propriété CSS, on doit indiquer une valeur. Par exemple, pour la propriété color, il faut indiquer le nom de la couleur. Pour font-size, il faut indiquer quelle taille on veut, etc.

**Voici un exemple de code CSS permettant d'afficher les titres principaux (H1) en bleu :**

```
H1 {  
    color : blue ;  
}
```

## Remarque:

**Les sélecteurs sont les éléments HTML, on peut donc indiquer pour chaque élément un ensemble de déclarations (lorsqu'il y en a plusieurs, on les sépare avec des points virgules). De même si des déclarations s'appliquent à plusieurs sélecteurs, on peut les regrouper. Dans l'exemple suivant, on affiche les titres et paragraphes en rouge et on place une marge gauche de 1cm dans tous les paragraphes, que l'on affiche en italique.**

```
H1, P {  
  color : red ;  
}
```

```
P {  
  margin-left : 1cm ;  
  font-style : italic;  
}
```

# **3- LE SELECTEUR: CLASS et ID**

**Ce qu'on a vu jusqu'ici a quand même un défaut : cela implique que TOUS les paragraphes possèdent la même mise en forme. Comment faire pour que certains paragraphes seulement soient écrits d'une manière différente ?**

**Pour résoudre le problème, on peut utiliser ces attributs spéciaux qui fonctionnent sur toutes les balises :**

**l'attribut class.**

**l'attribut id.**

**Les deux attributs sont quasiment identiques. Il y a seulement une petite différence qu'on va voir par la suite.**

### 3-1 le sélecteur class

**C'est un attribut que l'on peut mettre sur n'importe quelle balise, aussi bien titre que paragraphe, image, etc.**

```
<h1 class =""> </h1 >
```

```
<p class =""> </p>
```

```
<img class ="" />
```

**Si on veut appliquer les mêmes propriétés sur plusieurs balises, on définit la classe comme suit :**

```
.rouge {  
  
    color: red;  
  
    text-align: center;  
  
}
```

**Dans ce premier exemple, on indique que tous les éléments de classe "rouge", quelques soient la balise, devront être affichés en rouge et aligné au centre.**

```
P.entete {  
    font-style: italic;  
    font-family: "calibri;  
}
```

**Dans le second exemple, seuls les éléments P de classe "entete" devront être affichés en italique et de police calibri.**



## Exemple:

### Dans le fichier CSS

```
.rouge {
```

```
    color: red;
```

```
    text-align: center;
```

```
}
```

```
P.entete {
```

```
    font-style: italic;
```

```
    font-family: "calibri";
```

```
}
```

### Dans la page HTML

```
<p class="rouge"> paragraphe en rouge et au centre</p>
```

```
<p> paragraphe</p>
```

```
<h1 class="rouge">titre en rouge et au centre</h1>
```

```
<p class= "entete"> paragraphe italic avec police calibri</p>
```

## 7-2 le sélecteur id

À la place d'utiliser le sélecteur class, on peut également utiliser le sélecteur id. L'avantage du id par rapport à class est qu'il est unique dans le code HTML, c'est un attribut que l'on peut mettre sur n'importe quelle balise.

```
<h1 id =""> </h1 > OU
```

```
<p id =""> </p> OU
```

```
<img id ="" />
```

Et on définit le sélecteur id comme suit:

```
#rouge {  
    color: blue;  
    text-align: right;  
}
```

## Exemple:

### Dans le fichier CSS

```
#par1
```

```
{
```

```
    text-align: center;
```

```
    color: red;
```

```
    font-family: arial
```

```
}
```

```
#par2
```

```
{
```

```
    text-align: right;
```

```
    color: green;
```

```
    font-family: verdana
```

```
}
```

### Dans la page HTML

**<p id="par1"> ce paragraphe est rouge et centré</p>**

**<p id="par2"> ce paragraphe est vert est aligné à droite</p>**

# **4- LES BALISES UNIVERSELLES**

**Il arrivera parfois que vous aurez besoin d'appliquer une class (ou un id) à certains mots qui ne sont pas entourés par des balises.**

**En effet, le problème de class ou id, c'est qu'il s'agit d'un attribut. Vous ne pouvez donc en mettre que sur une balise.**

**En fait, on a inventé deux balises dites universelles, **qui n'ont aucune signification particulière**:**

**<span>**

**<div>**

**<span> </span> : c'est une balise de type inline, c'est-à-dire une balise que l'on place au sein d'un paragraphe de texte, pour sélectionner certains mots uniquement. les balises <b> et <em> sont de la même famille.**

**<div> </div> : c'est une balise de type block, qui entoure un bloc de texte. Les balises <p>, <h1>, etc. sont de la même famille.**

## **5- APPLIQUER UN STYLE: SELECTEURS AVANCES**

## **\* : sélecteur universel**

```
* {  
  ...  
}
```

**Sélectionne toutes les balises sans exception.**

## **A B : une balise contenue dans une autre**

```
h3 em {  
  ...  
}
```

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

## **Exemple**

```
<h3 >Titre avec <em >texte important </em ></h3 >
```



## A>B: Combinateurs

**article > p {**  
...  
**}**

on combine les sélecteurs pour cibler plus finement les éléments dans nos documents. L'exemple suivant sélectionne **les paragraphes enfants directs** de `<article>` grâce au combineur enfant (`>`) :

`<p>paragrapheNon</p>`

`<article >Titre`

`<p>Paragraphe </p>`

`<p>Paragraphe </p>`

`<p>Paragraphe </p>`

`</article >`

`<p>paragraphe Non</p>`

`<p>paragraphe Non</p>`

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

article + p{ ... }

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

Exemple :

<article >Titre </article >

<p>Paragraphe </p>

<p>Paragraphe </p>

## A ~ B : les balises qui en suit une autre

article ~ p{  
... }

Sélectionne **toutes les balises** <p> située après un titre <h3>.

Exemple :

<article>Titre </article >

<p>Paragraphe </p>

<p>Paragraphe </p>

**article > p{  
color:red;  
}**

**<article >Titre**

**<p>Paragraphe </p>**

**<p>Paragraphe </p>**

**<p>Paragraphe </p>**

**</article >**

**<p>paragrapheNon</p>**

**<p>paragrapheNon</p>**

**article + p{  
color:red;  
}**

**<article >Titre**

**<p>Paragraphe </p>**

**<p>Paragraphe </p>**

**<p>Paragraphe </p>**

**</article >**

**<p>paragrapheNon</p>**

**<p>paragrapheNon</p>**

**article ~ p{  
color:red;  
}**

**<article >Titre**

**<p>Paragraphe </p>**

**<p>Paragraphe </p>**

**<p>Paragraphe </p>**

**</article >**

**<p>paragrapheNon</p>**

**<p>paragrapheNon</p>**

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

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

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

**Exemple :**

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

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

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

**Idem, mais l'attribut doit en plus avoir exactement pour valeur "Cliquez ici".**

**Exemple :**

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

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

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

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

**Exemple :**

```
<a href =" http :// site .com" title =" Quelque part par ici ">
```

# **6- QUELQUES PROPRIETES POUR COMMENCER**

## 6-Quelques propriétés pour commencer

### 6-1 Formatage du texte

#### *La taille*

**Pour modifier la taille du texte, on utilise la propriété CSS **font-size**.  
Indiquer une taille absolue : en pixels, en centimètres ou millimètres.  
Indiquer une taille relative : en pourcentage, "em " ou écrire la taille avec des mots en anglais comme ceux-ci :**

**xx-small : minuscule ;**

**x-small : très petit ;**

**small : petit ;**

**medium : moyen ;**

**large : grand ;**

**x-large : très grand ;**

**xx-large : gigantesque.**

## ***La police***

**La propriété CSS qui permet d'indiquer la police à utiliser est **font-family**.**

**Vous devez écrire le nom de la police comme ceci :**

```
sélecteur{  
    font - family : police ;  
}
```

**pour éviter les problèmes si l'internaute n'a pas la même police que vous, on précise en général plusieurs noms de police, séparés par des virgules :**

```
sélecteur{  
    font - family : police1, police2 , police3 , police4 ;  
}
```



## *Italique, gras, souligné:*

Pour mettre en italique, on utilise **font-style** qui peut prendre trois valeurs :

**italic** : le texte sera mis en italique.

**oblique** : le texte sera passé en oblique (les lettres sont penchées, le résultat est légèrement différent de l'italique proprement dit).

**normal** : le texte sera normal (par défaut). Cela vous permet d'annuler une mise en italique.

Pour mettre en gras est **font-weight** et prend les valeurs suivantes :

**bold** : le texte sera en gras ;

**normal** : le texte sera écrit normalement (par défaut).

**La propriété CSS associée porte bien son nom : **text-decoration**. Elle permet, entre autres, de souligner le texte, mais pas seulement. Voici les différentes valeurs qu'elle peut prendre :**

**underline** : souligné.

**line-through** : barré.

**overline** : ligne au-dessus.

**none** : normal (par défaut).

## *L'alignement :*

Le langage CSS nous permet de faire tous les alignements connus, en utilisant la propriété **text-align** et en indiquant l'alignement désiré :

**left** : le texte sera aligné à gauche (c'est le réglage par défaut).

**center** : le texte sera centré.

**right** : le texte sera aligné à droite.

**justify** : le texte sera " justifié". Justifier le texte permet de faire en sorte qu'il prenne toute la largeur possible sans laisser d'espace blanc à la fin des lignes.

### Remarque :

**Vous ne pouvez pas modifier l'alignement du texte d'une balise **inline** (comme `<span>`, `<a>`, `<em>`, `<strong>`. . .). L'alignement ne fonctionne que sur des balises de type **block** (`<p>`, `<div>`, `<h1>`, `<h2>`, . . .) et c'est un peu logique, quand on y pense : on ne peut pas modifier l'alignement de quelques mots au milieu d'un paragraphe ! C'est donc en général le paragraphe entier qu'il vous faudra aligner.**

## 6-2 La couleur et le fond

### *Couleur du texte*

la propriété qui permet de modifier la couleur du  
texte : **color**.

```
h1{  
  color : maroon ;  
}
```

```
P{  
  color : # FFFFFFFF ;  
}
```

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

## *Couleur de fond*

Pour indiquer une couleur de fond, on utilise la propriété CSS **background-color**. Elle s'utilise de la même manière que la propriété **color**, c'est-à-dire que vous pouvez taper le nom d'une couleur, l'écrire en notation hexadécimale

```
body{  
    Background-color : blue ;  
}
```

```
p{  
    Background-color : #3A4F57 ;  
}
```

## *Image de fond*

La propriété permettant d'indiquer une image de fond est **background-image**. Comme valeur, on doit renseigner `url("nom_de_l_image.png")`.

Par exemple :

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

## *Options disponibles pour l'image de fond*

On peut compléter la propriété **background-image** que nous venons de voir par plusieurs autres propriétés qui permettent de changer le comportement de l'image de fond.

✓ **background-attachment** : fixer le fond

La propriété CSS **background-attachment** permet de "fixer" le fond. L'effet obtenu est intéressant car on voit alors le texte "glisser" par-dessus le fond. Deux valeurs sont disponibles :

**fixed** : l'image de fond reste fixe ;

**scroll** : l'image de fond défile avec le texte (par défaut).



✓ **background-repeat : répétition du fond**

Par défaut, l'image de fond est répétée en mosaïque. Vous pouvez changer cela avec la propriété **background-repeat** :

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

**repeat-x** : le fond sera répété uniquement sur la première ligne, horizontalement.

**repeat-y** : le fond sera répété uniquement sur la première colonne, verticalement.

**repeat** : le fond sera répété en mosaïque (par défaut).

✓ **background-position : position du fond**

**On peut indiquer où doit se trouver l'image de fond avec **background-position**.**

**Cette propriété n'est intéressante que si elle est combinée avec **background-repeat: no-repeat;** (un fond qui ne se répète pas).**

**Vous devez donner à **background-position** deux valeurs en pixels pour indiquer la position du fond par rapport au coin supérieur gauche de la page (ou du paragraphe, si vous appliquez le fond à un paragraphe).**

**Ainsi, si vous tapez :**

**background - position : 30px 50px ;**

**. . . votre fond sera placé à 30 pixels de la gauche et à 50 pixels du haut.**

**Il est aussi possible d'utiliser ces valeurs en anglais :**

**top : en haut ;**

**bottom : en bas ;**

**left : à gauche ;**

**center : centré ;**

**right : à droite.**

**Il est possible de combiner ces mots. Par exemple, pour aligner une image en haut à droite, vous taperez :**

**background - position : top right ;**

## **background-size : redimensionnement et à la mise à l'échelle**

1. Si la propriété **background-size** est définie sur "**contain**", l'image d'arrière-plan sera **mise à l'échelle** et tentera de s'adapter à la zone de contenu.

Cependant, l'image conservera son rapport d'aspect (la relation proportionnelle entre la largeur et la hauteur de l'image)

2. Si la propriété **background-size** est définie sur "**100% 100%**", l'image d'arrière-plan s'étirera pour couvrir toute la zone de contenu :

3. Si la propriété **background-size** est définie sur "**cover**", l'image d'arrière-plan sera mise à l'échelle pour couvrir toute la zone de **contenu**. Notez que la valeur "cover" conserve le format d'image et qu'une partie de l'image d'arrière-plan peut être tronquée.

## ✓ Combiner les propriétés

vous pouvez utiliser une sorte de "super-propriété " appelée **background** dont la valeur peut combiner plusieurs des propriétés vues précédemment : **background-image**, **background-attachment**, **background-repeat**, et **background-position**.

On peut donc tout simplement écrire :

```
body
{
    background : url (" soleil .png ") fixed no-repeat top right ;
}
```

L'ordre des valeurs n'a pas d'importance. Vous pouvez combiner les valeurs dans n'importe quel ordre.

## Remarque

### Valeur initiale

- `background-image`: none
- `background-position`: 0% 0%
- `background-size`: auto auto
- `background-repeat`: repeat
- `background-origin`: padding-box
- `background-clip`: border-box
- `background-attachment`: scroll
- `background-color`: transparent

## ***La transparence***

**Le CSS nous permet de jouer très facilement avec les niveaux de transparence des éléments, Pour cela, nous allons utiliser des fonctionnalités de CSS3 : la propriété **opacity**.**

**La propriété opacity, très simple, 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, votre élément sera opaque à 60%. . . et on verra donc à travers.**

## Exemple:

**Voici comment on peut l'utiliser :**

**p**

**{**

**opacity : 0.6;**

**}**

un texte avec opacity de 0,6

texte normal

un texte avec opacity de 0,2



## 6-3 Les dimensions

**Par défaut, un bloc prend 100% de la largeur disponible, si on veut la modifier on utilise les propriétés suivantes :**

**width :** c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%).

**height :** c'est la hauteur du bloc. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).

## 6-4 Les marges

**Il faut savoir que tous les blocs possèdent des marges. Il existe deux types :**

- ✓ **les marges intérieures.**
- ✓ **les marges extérieures.**

**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.

Margin (marges externes)

Bordures

Padding (marges internes)

Contenu de l'élément

**margin et padding s'applique aux quatre côtés du bloc. Si vous voulez spécifier des marges différentes en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises.**

**pour margin :**

**margin-top** : marge extérieure en haut ;

**margin-bottom** : marge extérieure en bas ;

**margin-left** : marge extérieure à gauche ;

**margin-right** : marge extérieure à droite.

**Et la liste pour padding :**

**padding-top** : marge intérieure en haut ;

**padding-bottom** : marge intérieure en bas ;

**padding-left** : marge intérieure à gauche ;

**padding-right** : marge intérieure à droite.

## 6-5 Les bordures et les ombres

### **Bordures standard**

**Le CSS vous offre un large choix de bordures pour décorer votre page. De nombreuses propriétés CSS vous permettent de modifier l'apparence de vos bordures :**

- **La largeur : indiquez la largeur de votre bordure. `border-width` , Mettez une valeur en pixels**
- **La couleur : c'est la couleur de votre bordure. `border-color`, Utilisez, comme on l'a appris, soit un nom de couleur (black, red, . . .), soit une valeur hexadécimale (`#FF0000`)**
- **Le type de bordure : là, vous avez le choix. Votre bordure peut être un simple trait, ou des pointillés, ou encore des tirets, etc.**

**`border-style`, Voici les différentes valeurs disponibles :**

**none** : pas de bordure (par défaut) ;

**solid** : un trait simple ;

**dotted** : pointillés ;

**dashed** : tirets ;

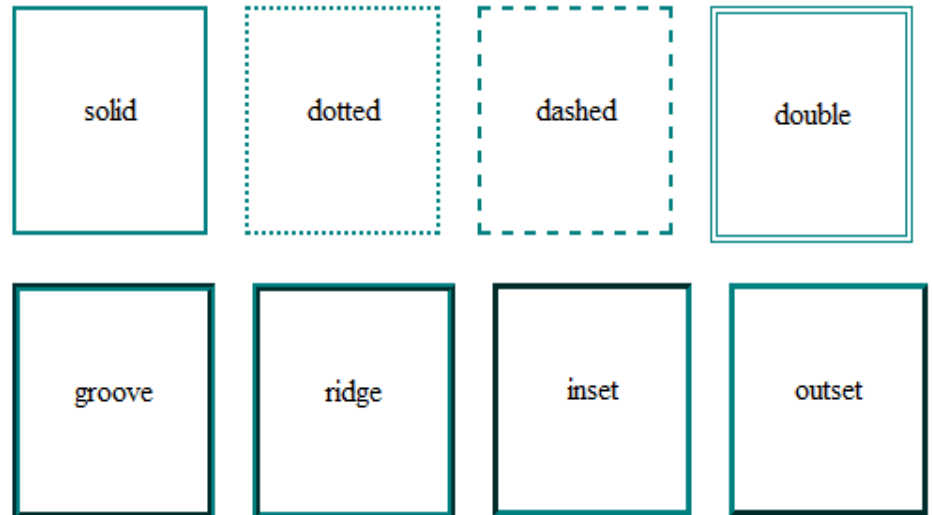
**double** : bordure double ;

**groove** : en relief ;

**ridge** : autre effet relief ;

**inset** : effet 3D global enfoncé ;

**outset** : effet 3D global surélevé.



## Combiner les propriétés

**vous pouvez même ici utiliser "super-propriété" appelée border dont la valeur peut combiner plusieurs des propriétés.**

**Ainsi, pour avoir une bordure bleue, en tirets, épaisse de 3 pixels autour de mes titres, je vais écrire :**

```
h1
```

```
{
```

```
border : 3px blue dashed ;
```

```
}
```

### **Remarque :**

**si vous voulez mettre des bordures différentes en fonction du côté (haut, bas, gauche ou droite), vous pouvez le faire sans problème.**

**Dans ce cas, vous devrez utiliser ces quatre propriétés :**

**border-top** : bordure du haut ;

**border-bottom** : bordure du bas ;

**border-left** : bordure de gauche ;

**border-right** : bordure de droite.



## Bordures arrondies

La propriété `border-radius` va nous permettre d'arrondir facilement les angles de n'importe quel élément. Il suffit d'indiquer la taille (l'importance) de l'arrondi en pixels :

```
p{ border - radius : 10px ;}
```

On peut aussi préciser la forme de l'arrondi pour chaque coin. Dans ce cas, indiquez quatre valeurs :

```
p { border - radius : 10px 5px 10px 5px; }
```

Les valeurs correspondent aux angles suivants dans cet ordre :

1. en haut à gauche ;
2. en haut à droite ;
3. en bas à droite ;
4. en bas à gauche.

**TP**

# EXERCICE

**Authentification**

Nom d'utilisateur

Mot de passe

## Les ombres

**Les ombres font partie des nouveautés récentes proposées par CSS3. Aujourd'hui, il suffit d'une seule ligne de CSS pour ajouter des ombres dans une page !**

**Nous allons ici découvrir deux types d'ombres :**

- **les ombres des boîtes ;**
- **les ombres du texte.**

## **box-shadow** : les ombres des boîtes

La propriété **box-shadow** s'applique à tout le bloc et prend quatre valeurs dans l'ordre suivant :

1. le décalage horizontal de l'ombre ;
2. le décalage vertical de l'ombre ;
3. l'adoucissement du dégradé ;
4. la couleur de l'ombre.

Par exemple, pour une ombre noire de 6 pixels, sans adoucissement, on écrira :

```
p { box - shadow : 6px 6px 0px black ; }
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ullamcorper sodales elit, sit amet pellentesque lectus aliquet quis. Etiam sem ipsum, rhoncus eu aliquam nec, mattis consectetur tortor. Mauris non lectus magna, vel interdum elit. Sed fermentum commodo commodo. Fusce imperdiet vestibulum neque, id pulvinar urna ultricies ullamcorper. Donec euismod, ipsum vehicula pretium tempor, mauris odio pellentesque metus, et ultrices arcu mauris sit amet leo. Curabitur ac scelerisque sem.

## **l'adoucissement du dégradé ;**

**p{**

**box - shadow : 6px 6px 6px black ;**

**}**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ullamcorper sodales elit, sit amet pellentesque lectus aliquet quis. Etiam sem ipsum, rhoncus eu aliquam nec, mattis consectetur tortor. Mauris non lectus magna, vel interdum elit. Sed fermentum commodo commodo. Fusce imperdiet vestibulum neque, id pulvinar urna ultricies ullamcorper. Donec euismod, ipsum vehicula pretium tempor, mauris odio pellentesque metus, et ultrices arcu mauris sit amet leo. Curabitur ac scelerisque sem.

**On peut aussi rajouter une cinquième valeur facultative : inset. Dans ce cas, l'ombre sera placée à l'intérieur du bloc, pour donner un effet enfoncé :**

**p{**

**box - shadow : 6px 6px 6px black inset ;**

**}**

## text-shadow : l'ombre du texte

**Avec text-shadow, vous pouvez ajouter une ombre directement sur les lettres de votre texte ! Les valeurs fonctionnent exactement de la même façon que box-shadow : décalage, adoucissement et couleur.**

**p{**

**text - shadow : 2px 2px 4px black ;**

**}**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ullamcorper sodales elit, sit amet  
pellentesque lectus aliquet quis. Etiam sem ipsum, rhoncus eu aliquam nec, mattis consectetur tortor.  
Mauris non lectus magna, vel interdum elit. Sed fermentum commodo commodo. Fusce imperdiet  
vestibulum neque, id pulvinar urna ultricies ullamcorper. Donec euismod, ipsum vehicula pretium tempor,  
mauris odio pellentesque metus, et ultrices arcu mauris sit amet leo. Curabitur ac scelerisque sem.

## **6-6 Le positionnement en CSS**

**Vous allez voir, il existe plusieurs techniques permettant d'effectuer la mise en page de son site. Chacune a ses avantages et ses défauts, ce sera à vous de sélectionner celle qui vous semble la meilleure selon votre cas.**

**En HTML, Il existe cinq façons différentes de créer des dispositions des éléments :**

- **Tableaux HTML (non recommandé)**
- **Propriété float CSS (méthode ancienne)**
- **Flexbox CSS**
- **Framework CSS (ex: Bootstrap)**
- **Grille CSS**




## ***A- Le positionnement flottant***

**Le CSS nous permet de faire flotter un élément autour du texte. On dit aussi qu'on fait un "habillage".**

**la propriété qui fait flotter est **float** ("flottant" en anglais). Cette propriété peut prendre deux valeurs très simples :**

**left :** l'élément flottera à gauche.

**right :** l'élément flottera à droite !



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vitae lorem imperdiet lacus molestie molestie. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec eu purus. Phasellus metus lorem, blandit et, posuere quis, tincidunt vitae, ante. Vivamus consequat mauris a diam. Vivamus nibh erat, hendrerit nec, aliquet ut, hendrerit quis, nunc. Vestibulum et turpis et elit tempor euismod.

**Il existe en fait une propriété CSS qui permet de dire : " Stop, ce texte doit être en dessous du flottant et non plus à côté ".**

**C'est la propriété **clear**, qui peut prendre ces trois valeurs :**

**left :** le texte se poursuit en-dessous après un float: left;

**right :** le texte se poursuit en-dessous après un float: right;

**both :** le texte se poursuit en-dessous, que ce soit après un float: left; ou après un float: right.

## Exemple: Faire flotter une image

### ✓ *En HTML*

`<p>` Ceci est un texte normal de paragraphe , écrit à la suite de l' image et qui l' habillera car l' image est flottante `</p>`

### ✓ *En CSS*

```
. imageflottante {  
    float : left ;  
}
```

**TP**



# Ma première page avec du style

Bienvenue sur ma page avec du style!

Il lui manque des images, mais au moins, elle a du style. Et elle a des liens, même s'ils ne mènent nulle part...

[Page d'accueil](#)

[Réflexions](#)

[Ma ville](#)

[Liens](#)

page réalisée pour EFM par votre nom et prénom

## ***B- Transformez vos éléments avec display***

**Il existe en CSS une propriété très puissante : `display`. Elle est capable de transformer n'importe quel élément de votre page d'un type vers un autre. Avec cette propriété magique, je peux par exemple imposer à mes liens (originellement de type inline) d'apparaître sous forme de blocs :**

```
a{  
  display : block ;  
}
```

**À ce moment-là, les liens vont se positionner les uns en-dessous des autres (comme des blocs normaux) et il devient **possible de modifier leurs dimensions** !**

**Voici quelques-unes des principales valeurs que peut prendre la propriété **display** en CSS (il y en a encore d'autres) :**

Valeur	Exemples	Description
inline	<a>, <em>, <span>...	Eléments d'une ligne. Se placent les uns à côté des autres.
block	<p>, <div>, <section>...	Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
inline-block	<select>, <input>	Eléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).
none	<head>	Eléments non affichés.

***Comment je reconnais une balise inline d'une balise block ?***

**block :** une balise de type block sur votre page web crée automatiquement un retour à la ligne avant et après.

**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.



**Les dimensions par défaut du contenu des éléments HTML vont avant tout être déterminées par le type d’affichage des éléments : en effet, les éléments de type **block** occuperont par défaut toute la largeur disponible dans leur élément parent tandis que les éléments de type **inline** n’occuperont que la largeur nécessaire à leur contenu.**

**Notez bien ici que modifier la taille de la boîte du contenu d’un élément de type **block** ne change pas les propriétés fondamentales de celui-ci. J’entends par là qu’un élément de type **block** commencera toujours sur une nouvelle ligne et ne viendra jamais se positionner à côté d’un autre élément de type **block** quelle que soit sa taille.**

### ***C- Les positionnements absolu, fixe et relatif***

**Il existe d'autres techniques un peu particulières permettant de positionner avec précision des éléments sur la page :**

**Le positionnement absolu :** il nous permet de placer un élément n'importe où sur la page (en haut à gauche, en bas à droite, tout au centre, etc.).

**Le positionnement fixe :** identique au positionnement absolu mais, cette fois, l'élément reste toujours visible, même si on descend plus bas dans la page. C'est un peu le même principe que background-attachment: fixed.

**Le positionnement relatif :** permet de décaler l'élément par rapport à sa position normale.

**Il faut d'abord faire son **choix entre les trois modes** de positionnement disponibles.**

**Pour cela, on utilise la propriété CSS position à laquelle on donne une de ces valeurs :**

**absolute** : positionnement absolu ;

**fixed** : positionnement fixe ;

**relative** : positionnement relatif.

**Mais cela ne suffit pas ! On a dit qu'on voulait un positionnement absolu, fixe ou relatif mais encore faut-il dire où l'on veut que **le bloc soit positionné sur la page**. Pour ce faire, on va utiliser **quatre propriétés CSS** :**

**left** : position par rapport à la gauche de la page ;

**right** : position par rapport à la droite de la page ;

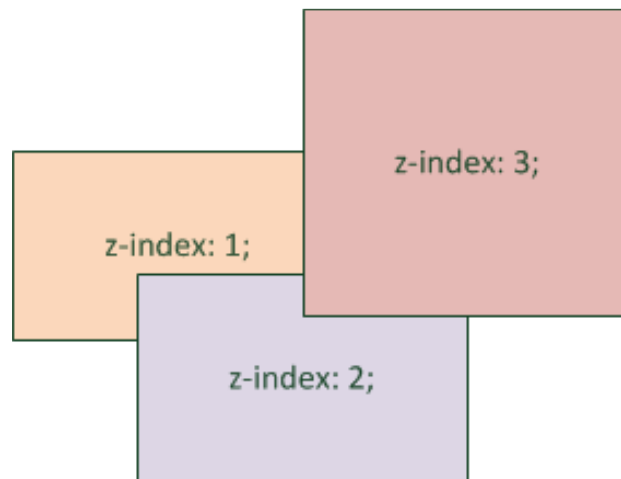
**top** : position par rapport au haut de la page ;

**bottom** : position par rapport au bas de la page

**On peut leur donner une valeur en pixels, comme 14px, ou bien une valeur en pourcentage 50%.**

**Les éléments positionnés en absolu** sont placés par-dessus le reste des éléments de la page ! Par ailleurs, si vous placez deux éléments en absolu vers le même endroit, ils risquent de se chevaucher. Dans ce cas, utilisez la propriété **z-index** pour indiquer quel élément doit apparaître au-dessus des autres.

L'élément ayant la valeur de z-index la plus élevée sera placé en dessus des autres.



**Le positionnement relatif** permet d'effectuer des "ajustements":  
l'élément est décalé par rapport à sa position initiale.

Prenons par exemple un texte important, situé entre deux balises `<strong>`. Pour commencer, je le mets sur fond rouge pour qu'on puisse mieux le repérer :

Pas de doute,  si on veut comprendre corre

```
Strong{  
  position      : relative ;  
  left          : 55px ;  
  top           : 10px ;  
}
```

**Le texte est alors décalé par rapport à sa position initiale.**

Pas de doute,  si on veut com

**TP**

## Présentation



Vous etes a la recherche d'une personne competente et efficace pour gerer vos projets web ?

Actuellement, je suis ... je fais ... actuellement, je suis ... je fais ... actuellement, je suis ... je fais ... actuellement, je suis ... je fais ...

Redaction cahier des charges  
elaboration de votre design & Integration precise  
Administration sur CMS : Wordpress & Drupal  
Reflexion strategique, statistiques, referencement



## Actualités

? Résultat

16/07/2020 : Formation en ligne

17/08/2020 : Actualité 3

03/12/2020 : Actualité 2

02/07/2020 : Actualité 1

## Qui suis-je ?

Description en quelques lignes ...

Description en quelques lignes ...

Description en quelques lignes ...

## Expérience

- Experience 1 ...
- Experience 2 ...
- Experience 3 ...

## Contact

Contact ...



## Inscription

Nom

Prénom

Sexe

☒ Féminin ☐ Masculin

Date de naissance

Niveau d'étude

Spécialité

Téléphone

Email

Site web  
personnel

Adresse

Photo

Choisir un fichier

Aucun fichier choisi

## *d- Les positionnements avec des boîtes flexibles ou flexbox*

Le flexbox est un modèle de disposition très puissant qui va nous permettre de contrôler facilement et avec précision **l'alignement, la direction, l'ordre et la taille de nos éléments**.

Avec le modèle des boîtes flexibles, nous allons pouvoir définir des **conteneurs flexibles**.

Ces conteneurs sont dits « flexibles » car **leurs enfants directs vont être des éléments flexibles** qui vont pouvoir se réarranger (se redimensionner, se réaligner, etc.)

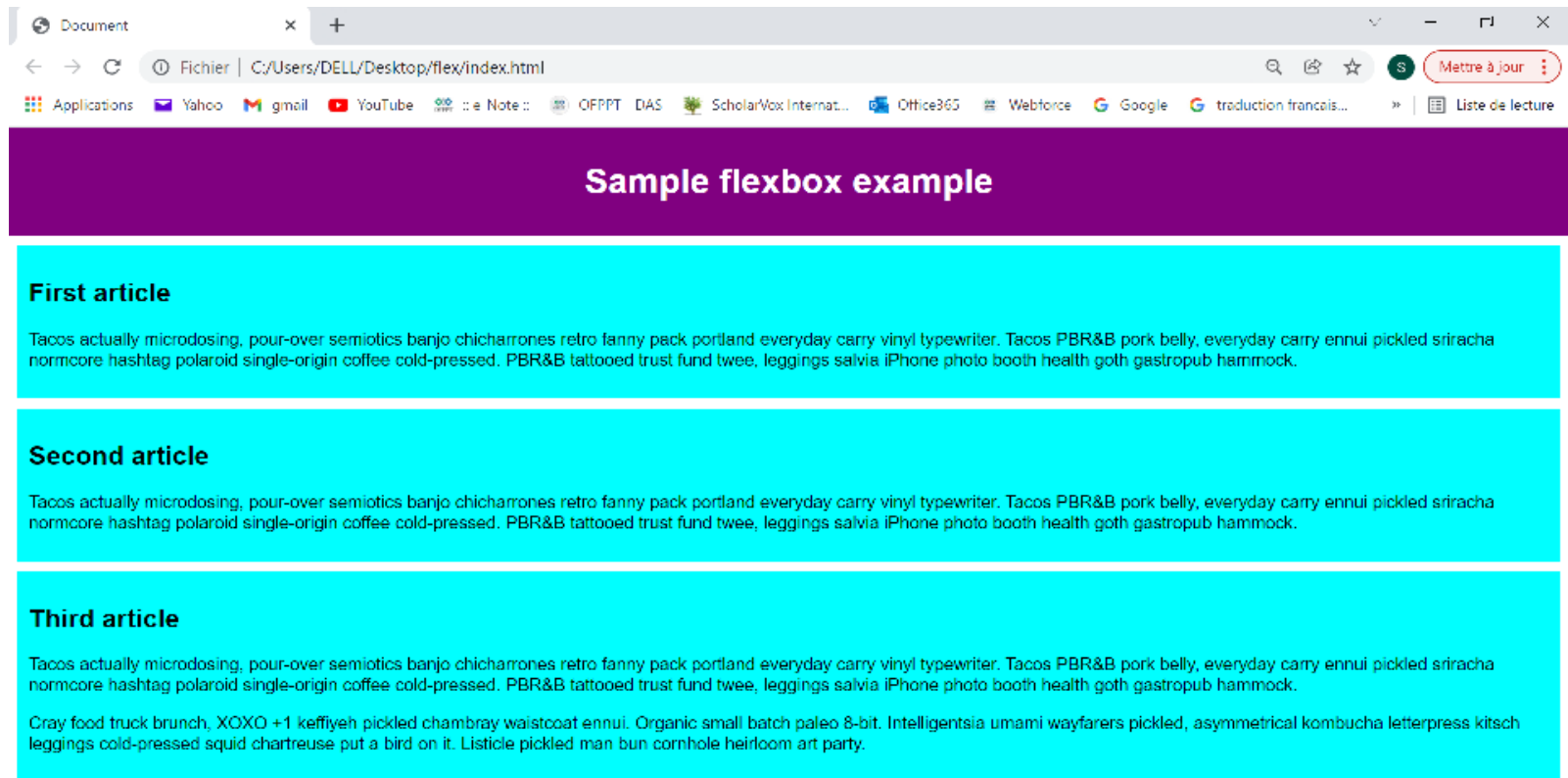
automatiquement dans leur conteneur lorsque celui-ci change de dimension.

Le flexbox représente ainsi en quelque sorte une alternative beaucoup plus puissante à la propriété **float** qui nous permettait de positionner nos boîtes **horizontalement** puisqu'on va avoir cette fois-ci un contrôle total sur la disposition des éléments.

Le but de cette partie est de vous présenter un nouveau mode d'affichage et de disposition très puissant : la disposition selon un modèle de boîtes flexibles ou **flexbox** avec **display : flex;** (ou **display : inline-flex**) et les différentes propriétés CSS liées à ce modèle.

## Exemple:

Soit un élément <header> avec un en-tête de haut niveau à l'intérieur, et un élément <section> contenant trois éléments <article>.



## *Comment utiliser Flexbox ?*

Le **conteneur flex** (flex container) ou bien la boîte dite « parent », contient les **éléments flex** (flex items) ou bien les boîtes « enfants ».

L'idée est définir, en utilisant les différentes propriétés CSS, l'ordre des "boîtes enfants" dans la "boîte parent".

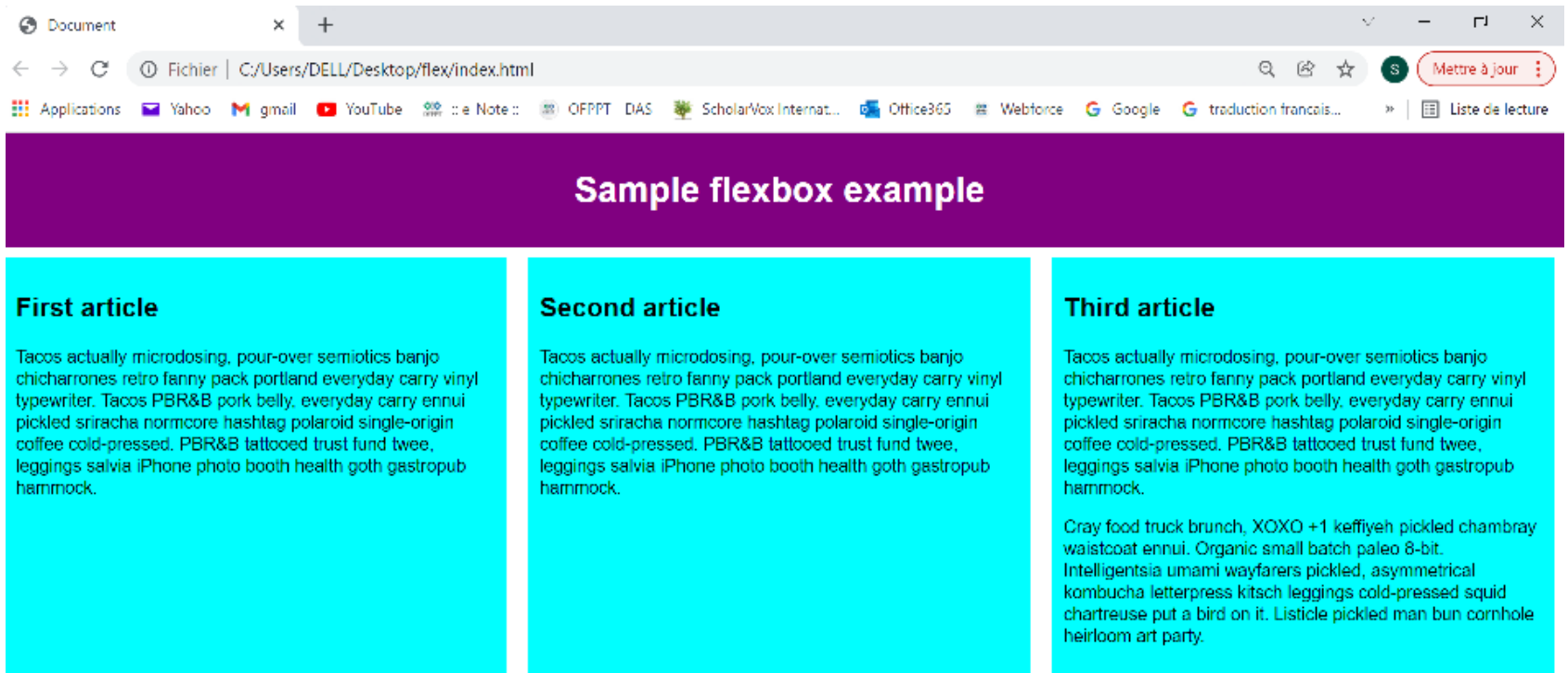
### Définir le conteneur flex (flex container)

Le conteneur est une balise html qui contient d'autres balises html.

Pour commencer, sélectionnons les éléments devant être présentés sous forme de boîtes flexibles. Pour ce faire, donnons une valeur spéciale à la propriété display du parent de ces éléments à disposer.

Dans ce cas, comme cela concerne les éléments <article>, nous affectons la valeur flex à l'élément <section> (qui devient un conteneur flex) :

```
section {  
    display: flex;  
}
```



**Nous avons ainsi notre disposition en plusieurs colonnes de largeur égale et toutes de même hauteur. Ceci parce que les valeurs par défaut données aux éléments flex (les enfants du conteneur flex) sont configurés pour résoudre des problèmes courants tels celui-ci.**

## Définir les axes du conteneur flex avec la propriété **flex-direction**

Lorsque les éléments sont disposés en boîtes flexibles, ils sont disposés le long de deux axes :

### **Axe principal : main axis**

Le conteneur flex possède un axe principal (main axis) sur lequel les éléments flex peuvent se suivre.

### **Axe secondaire : cross axis**

Le choix de l'axe principal impose la direction de l'axe secondaire (cross axis) qui lui est toujours perpendiculaire.



## Colonnes ou lignes ?

Flexbox dispose de la propriété **flex-direction** pour indiquer la **direction de l'axe principal** (direction dans laquelle les enfants flexibles sont disposés). Cette propriété est égale par défaut à **row** : ils sont donc disposés en ligne, dans le sens de lecture de la langue par défaut du navigateur (de gauche à droite, dans le cas d'un navigateur français). Ajoutez la déclaration suivante dans la règle CSS pour l'élément `<section>` de notre exemple:

```
flex-direction: column;
```

Cela dispose de nouveau les éléments en colonnes, comme c'était le cas avant l'ajout de la CSS. Avant de poursuivre, enlevez cette déclaration de l'exemple.



**L'axe principal** est défini par la propriété **flex-direction** qui peut prendre les valeurs suivantes:

- **row** : les éléments flex se suivent sur une ligne, ce qui correspond à la direction normale du sens d'écriture du document
- **row-reverse** : les éléments flex se suivent sur une ligne en ordre inversé. ce qui correspond à la direction inverse du sens d'écriture du document
- **column** : les éléments flex se suivent sur une colonne.
- **column-reverse** : les éléments flex se suivent sur une colonne en ordre inversé

Le choix des axes est essentiel car leur orientation va déterminer le résultat de l'application des propriétés suivantes :

**justify-content** : définit comment les éléments flex sont positionnés le long de **l'axe principal** ;

**align-items** : définit comment les éléments flex sont positionnés le long de **l'axe secondaire** ;

**align-self** : définit comment un seul élément flex est positionné le long de l'axe secondaire.

## Aligner le long de l'axe principal : la propriété **justify-content**

La propriété "justify-content" permet d'aligner les éléments le long de l'axe principal

Les valeurs de la propriété "justify-content" : flex-start, flex-end, center, space-around, space-between.

**flex-start** => les éléments flexibles sont placés à partir de **la ligne de début** du conteneur sur l'axe principal.



**flex-end** => les éléments flexibles sont placés à partir de **la ligne de fin** du conteneur sur l'axe principal



**center** => les éléments flexibles sont **centrés** le long de l'axe principal.



**space-between** => l'espace disponible est réparti de façon **égale entre chaque** élément.



**space-around** => l'espace disponible est réparti de **façon égale entre chaque élément, y compris au début et à la fin.**



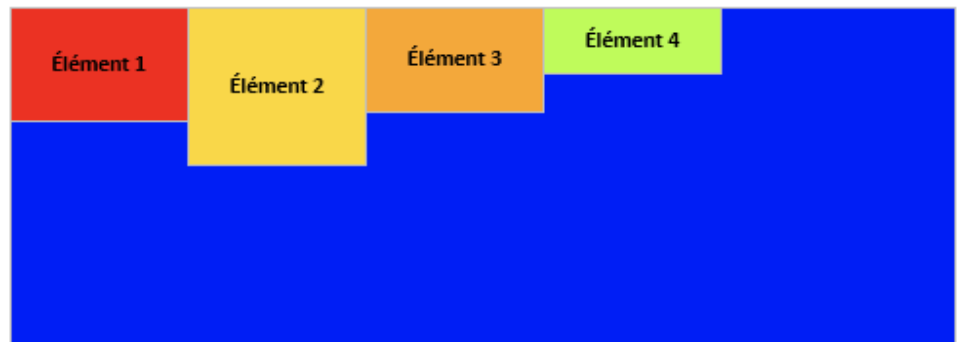
## Aligner le long de l'axe secondaire : la propriété **align-items**

La propriété "align-items" peut prendre 5 valeurs : **stretch**, **flex-start**, **flex-end**, **center** et **baseline**.

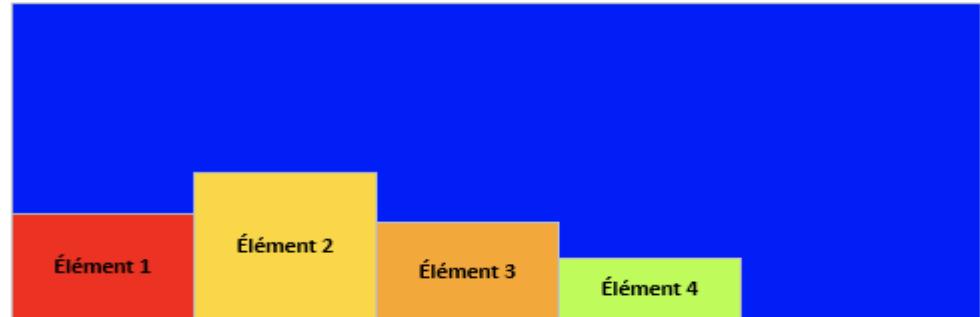
**stretch** => les éléments flexibles sont étirés le long de l'axe secondaire.



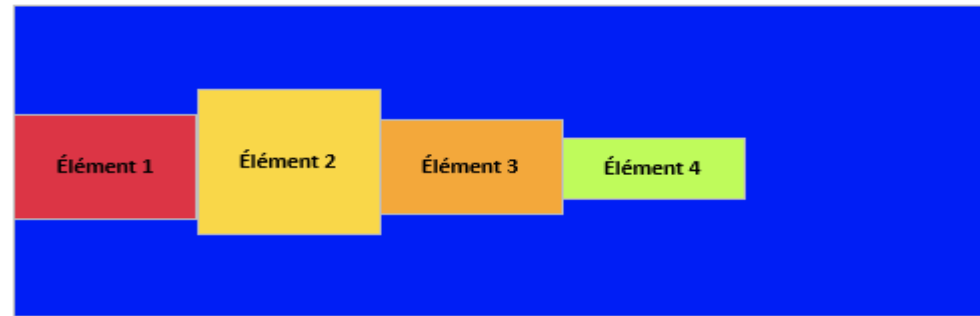
**flex-start** => les éléments flexibles sont alignés sur la ligne de début de l'axe secondaire.



**flex-end** => les éléments flexibles sont alignés sur la ligne de fin de l'axe secondaire.



**center** => les éléments flexibles sont centrés sur la ligne d'axe secondaire.

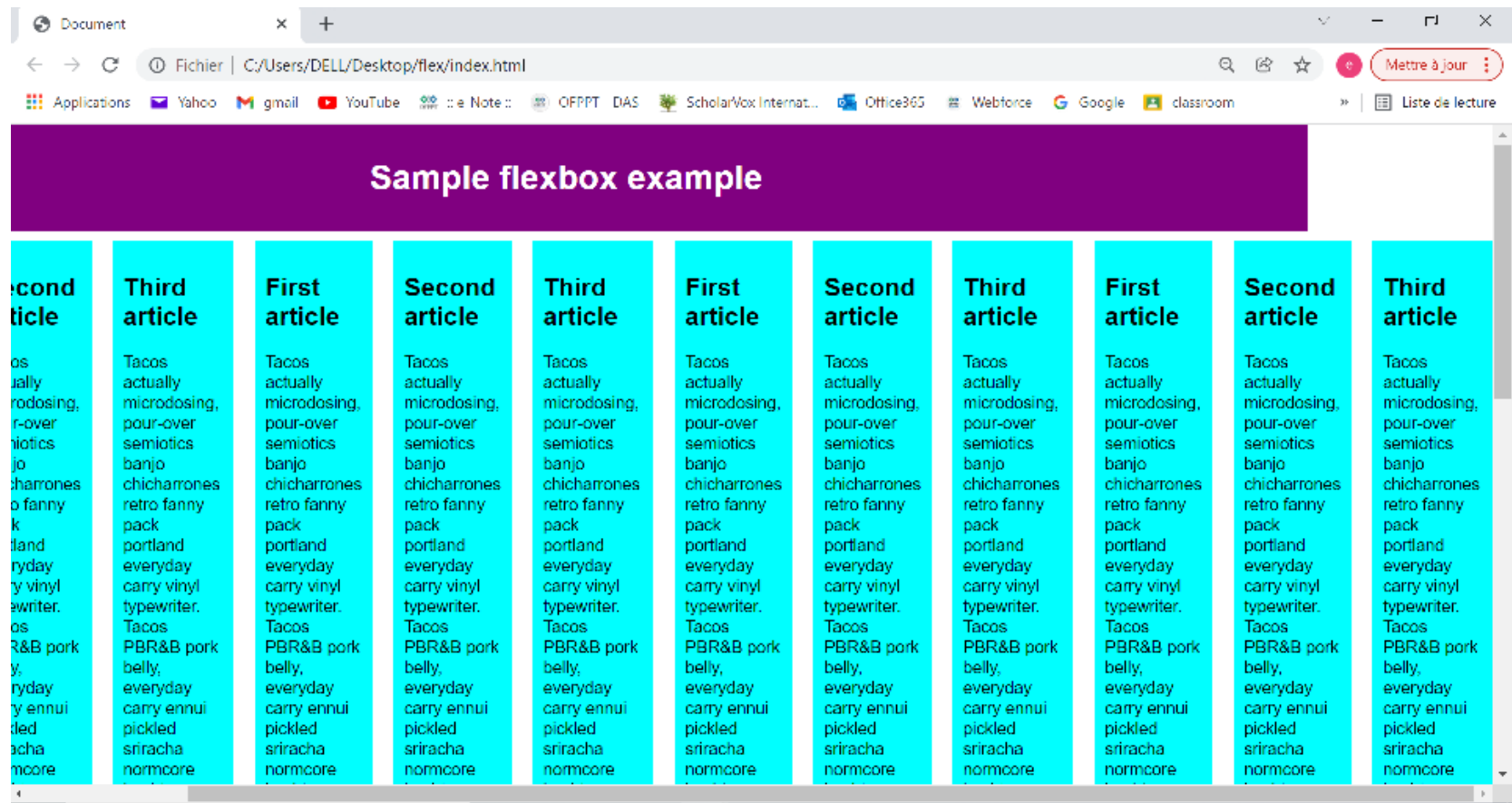


**baseline** => les éléments flexibles sont alignés sur leur ligne de base (les textes sont alignés).



## Enveloppement

**Problème:** quand votre structure est de largeur ou hauteur fixe, il arrive que les éléments flex débordent du conteneur et brisent cette structure.



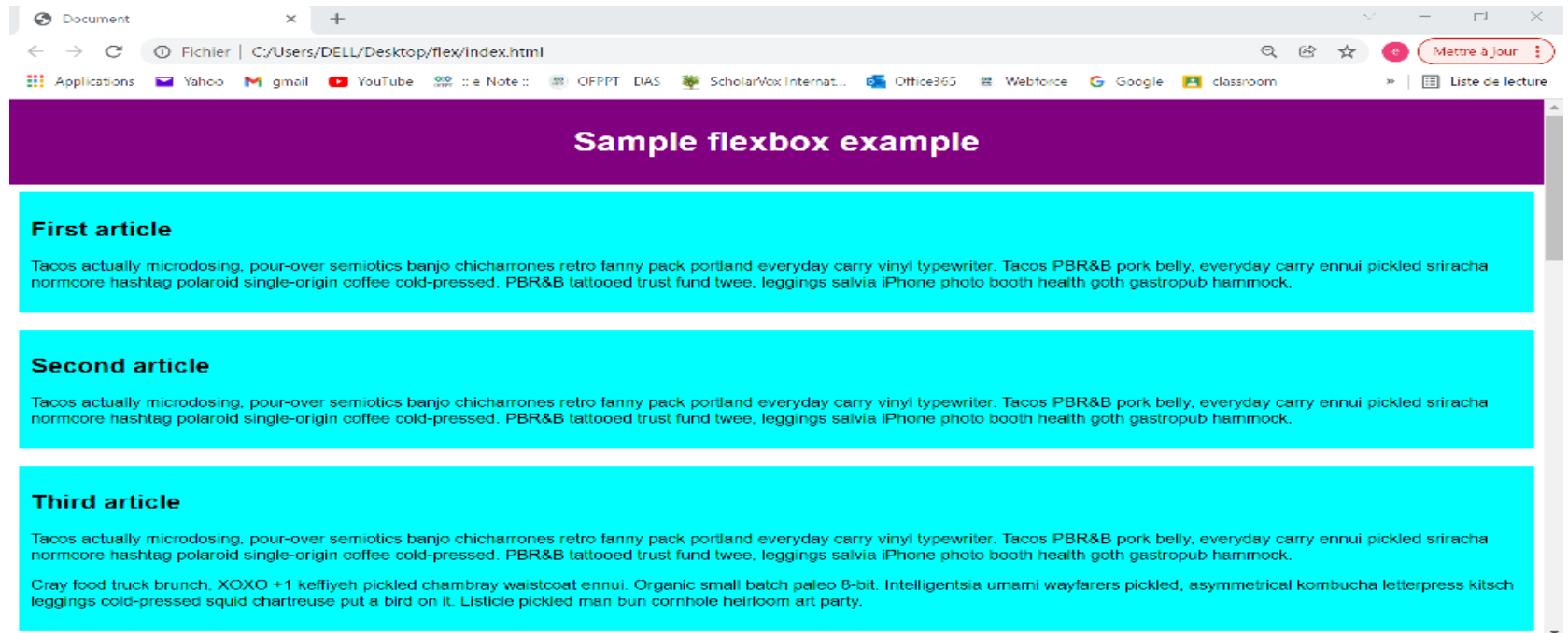
## Maitriser les passages à la ligne : la propriété **flex-wrap**

**La propriété "flex-wrap" peut prendre les valeurs suivantes :**

- **flex-wrap : nowrap** - c'est la valeur initiale. Si les éléments flexibles sont trop larges pour tenir dans leur conteneur, ils débordent du conteneur
- **flex-wrap : wrap** - si les éléments flexibles sont trop larges pour tenir dans le conteneur, ils passent automatiquement sur la ligne suivante (du haut vers le bas)
- **flex-wrap : wrap-reverse** - si les éléments flexibles sont trop larges pour le conteneur, ils passent automatiquement sur la ligne suivante (du bas vers le haut)

Ici, nous voyons que les enfants débordent du conteneur. Une façon d'y remédier est d'ajouter la déclaration suivante à votre règle pour <section> :

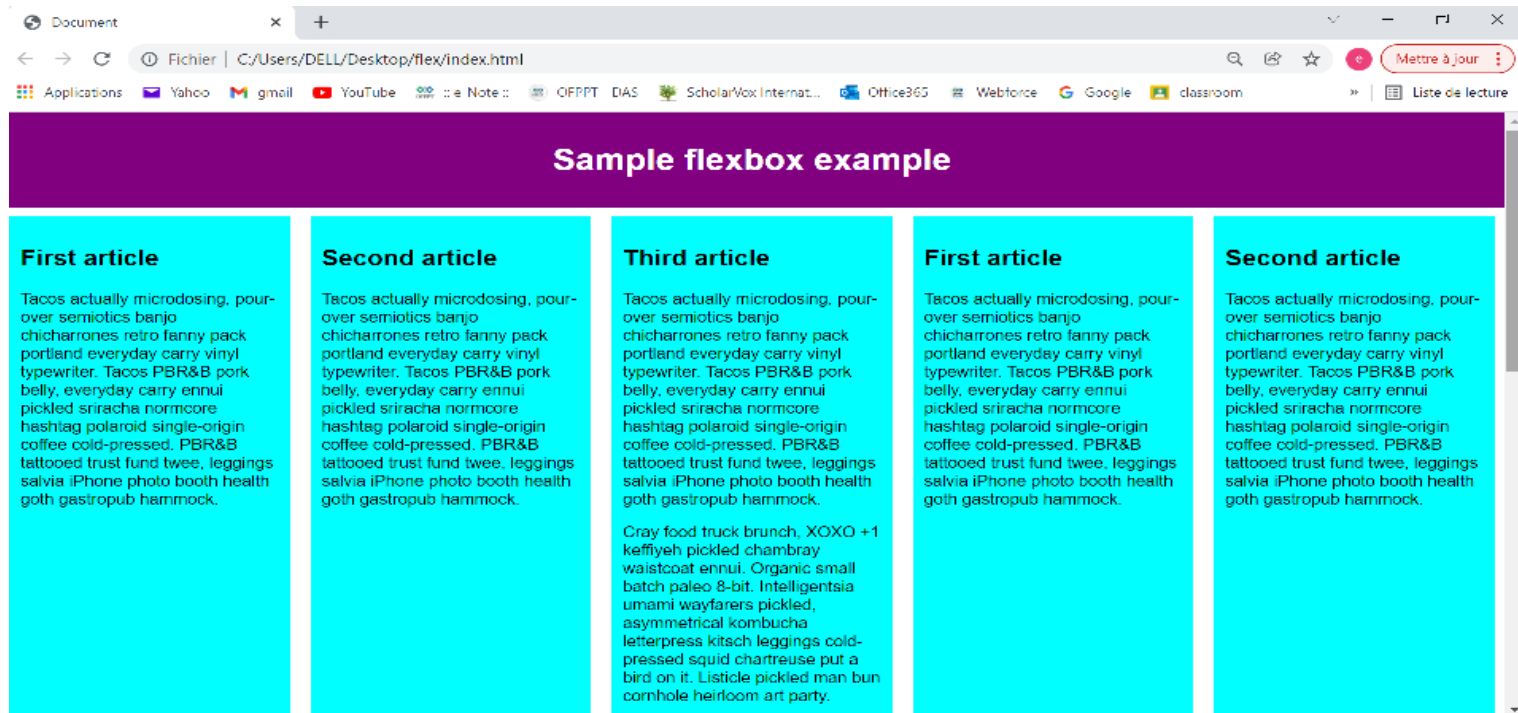
```
flex-wrap: wrap;
```



La propriété **flex** est une propriété raccourcie qui définit la capacité d'un élément flexible à modifier ses dimensions afin de remplir l'espace disponible de son conteneur.

Ajoutez aussi la déclaration suivante à votre css pour <article> : `flex: 200px;`

La déclaration `flex: 200px` pour les éléments article signifie que chacun aura **au moins 200px** de large ;



**Vous noterez aussi que chacun des enfants de la dernière rangée est plus large, de façon à ce que toute la rangée reste remplie.**



## Forme abrégée « flex-flow »

Notez maintenant qu'il y a une forme abrégée pour **flex-direction** et **flex-wrap** →

**flex-flow**. Ainsi, par exemple, vous pouvez remplacer :

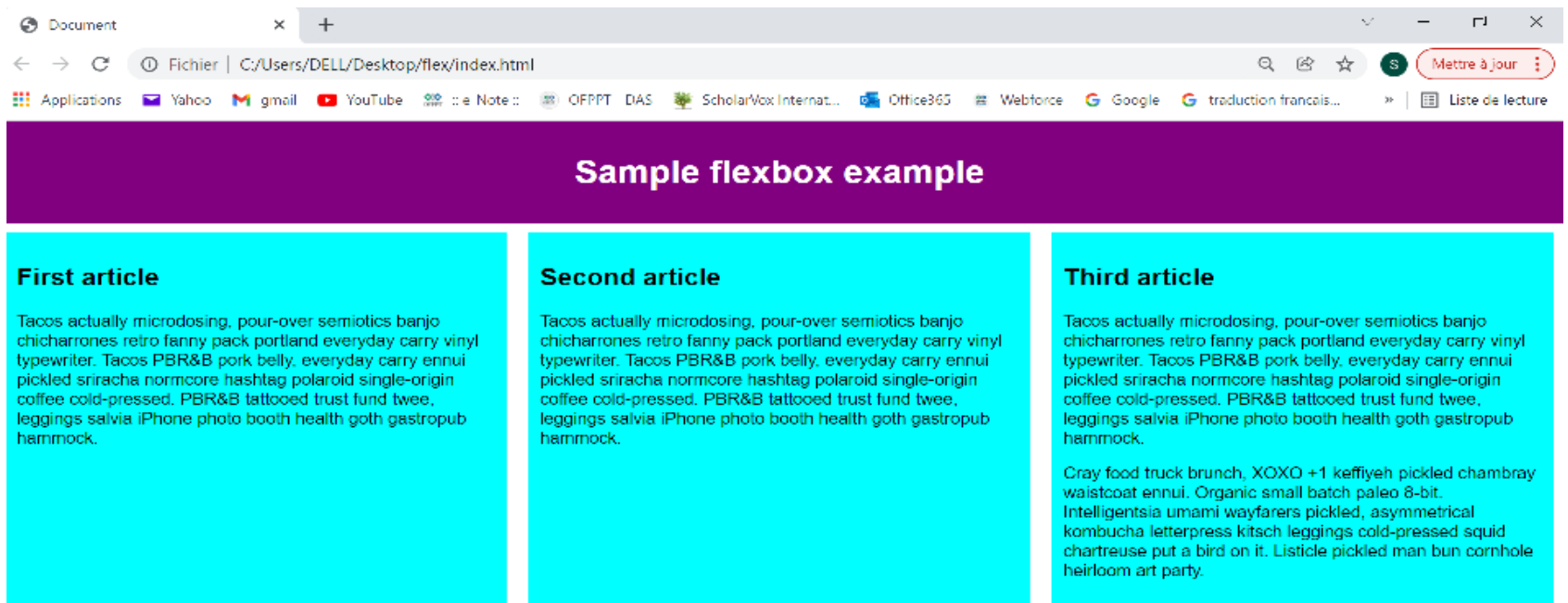
```
flex-direction: row;  
flex-wrap: wrap;
```

**Par:**

```
flex-flow: row wrap;
```

## Taille modulable des éléments flex

Revenons maintenant au premier exemple, et examinons comment nous pouvons contrôler la **proportion d'éléments flexibles** dans l'espace.



**Ajoutez d'abord :**  
**Sur CSS:**

```
article {  
    flex: 1;  
}
```

```
Article:last-child {  
    flex: 2;  
}
```



**Maintenant, lorsque vous actualisez, vous voyez que le troisième <article> occupe deux fois plus de largeur disponible que chacun des deux autres.**

## Positionner les éléments à l'intérieur d'un conteneur flex

**Les éléments flexibles peuvent être alignés, justifiés et distribués le long des axes de leur conteneur grâce à différentes propriétés.**

**Dans l'exemple suivant, on a choisi une direction "row", ce qui signifie que l'axe principal est l'axe horizontal et que l'axe secondaire est l'axe vertical.**

```
<div class="conteneur">
  <div class="element1">élément 1</div>
  <div class="element2">élément 2</div>
  <div class="element3">élément 3</div>
  <div class="element4">élément 4</div>
</div>
```

```
.conteneur {
  background-color : blue;
  height : 230px;
  width : 500px;
  display : flex;
  flex-direction : row;
  align-items : stretch | flex-start | flex-end | center | baseline;
  justify-content : flex-start | flex-end | center | space-around | space-between | space-evenly;
}
```

## **Remarque**

**Les simples exigences de mise en page suivantes sont difficiles sinon impossibles à réaliser de manière pratique et souple avec ces outils :**

**1-Centrer verticalement un bloc de contenu dans son parent ;**

**2-Faire que tous les enfants d'un conteneur occupent tous une même quantité de hauteur/largeur disponible selon l'espace offert ;**

**3-Faire que toutes les colonnes dans une disposition multi-colonnes aient la même hauteur même si leur quantité de contenu diffère.**

**La méthode **Flexbox** facilite la structuration et la mise en page souple et réactive, sans utiliser les marges et le positionnement.**

## Remarque:

### Les propriétés des éléments flex:

- La propriété **order** permet de contrôler l'ordre dans lequel les éléments apparaissent dans le conteneur flex.
- La propriété **flex-grow** permet de définir l'agrandissement possible d'un élément proportionnellement aux autres. La valeur de cette propriété est un nombre. Par exemple, si "flex-grow : 2" pour un élément, il occupe (si possible) deux fois plus d'espace que les autres.
- La propriété **flex-shrink** définit la possibilité pour un élément de rétrécir. Cette propriété accepte uniquement des valeurs positives
- La propriété **flex-basis** définit la taille par défaut d'un élément.
- La propriété **flex** est une propriété raccourcie **combinant** les propriétés "flex-grow", "flex-shrink" et "flex-basis"
- La propriété **align-self** correspond à l'application de la propriété "align-items" pour un seul élément

a propriété **flex** peut être définie avec une, deux ou trois valeurs.

**Avec une valeur, la syntaxe doit être :**

- un nombre sans unité : celui-ci est alors interprété comme la valeur de <flex-grow>
- ou une valeur de largeur valide : celle-ci est alors interprétée comme la valeur de <flex-basis>
- ou le mot-clé none.

**Avec deux valeurs**

- la première doit être un nombre sans unité qui correspond à la valeur de <flex-grow>.
- la seconde valeur doit être :
- un nombre sans unité : celui-ci est alors interprété comme la valeur de <flex-shrink>
- ou une valeur de largeur valide : celle-ci est alors interprétée comme la valeur de <flex-basis>

### **Avec trois valeurs**

- la première valeur doit être un nombre sans unité : celui-ci est alors interprété comme la valeur de <flex-grow>
- la deuxième valeur doit être un nombre sans unité : celui-ci est alors interprété comme la valeur de <flex-shrink>
- la troisième valeur doit être une valeur de largeur valide : celle-ci est alors interprétée comme la valeur de <flex-basis>



**TP**

**Logo**

**Links**

navigation

**Article 1**

**Article 2**

**Article 3**

**Article 4**

**Article 5**

**Article 6**

**Peid**

Logo

Links

navigation

Article 1

Article 2

Article 3

Article 4

Article 5

Article 6

```
<body>
  <header>
    <div class="logo">Logo</div>
    <div class="links">Links</div>
  </header>
  <nav>navigation</nav>
  <section>
    <article>Article 1</article>
    <article>Article 2</article>
    <article>Article 3</article>
    <article>Article 4</article>
    <article>Article 5</article>
    <article>Article 6</article>
  </section>
  <footer>Peid</footer>
</body>
```

Démarrer

Peid

## 6-7 Création d'apparences dynamiques

**Le CSS nous permet aussi de modifier l'apparence des éléments de façon dynamique, c'est-à-dire que des éléments peuvent changer de forme une fois que la page a été chargée. Nous allons faire appel à une fonctionnalité puissante du CSS : les pseudo-formats (pseudo-element et pseudo-class)**

```
Selector : pseudo-class {  
    property: value;  
}
```

```
Selector :: pseudo-element{  
    property: value;  
}
```

Une **pseudo-class** 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
- Stylez les liens visités et non visités différemment
- Styliser un élément lorsqu'il obtient le focus

```
selector:pseudo-class {  
  
    property: value;  
  
    }
```

## *Au survol*

**:hover** signifie "survoler", peut donc se traduire par : "Quand la souris est sur le lien " (quand on pointe dessus).

```
a: hover /* Apparence au survol des liens */
```

```
{  
  text-decoration : underline ;  
  color : green ;  
}
```

**vous pouvez modifier l'apparence des paragraphes lorsqu'on pointe dessus :**

```
p: hover /* Quand on pointe sur un paragraphe */
```

```
{  
  font-size:24px;  
  color : green ;  
}
```

## *Au clic et lors de la sélection*

**Vous pouvez interagir encore plus finement en CSS. Nous allons voir ici que nous pouvons changer l'apparence des éléments lorsque l'on clique dessus et lorsqu'ils sont sélectionnés !**

**:active** au moment du clic

**Le pseudo-format :active** permet d'appliquer un style particulier au moment du clic.

En pratique, il n'est utilisé que sur les liens.

On peut par exemple changer la couleur de fond du lien lorsque l'on clique dessus :

```
a: active /* Quand le visiteur clique sur le lien */  
{  
background - color : # FFCC66 ;  
}
```

**:focus** lorsque l'élément est sélectionné

Là, c'est un peu différent. **Le pseudo-format :focus** applique un style lorsque l'élément est sélectionné.

```
a: focus /* Quand le visiteur à déjà cliquer sur le lien */  
{  
background - color : # FFCC66 ;  
}
```



## *Lorsque le lien a déjà été consulté*

Par défaut, le navigateur colore le lien en un violet, Vous pouvez changer cette apparence avec **:visited** (qui signifie "visité" ).

```
a: visited /* Quand le visiteur a déjà vu la page concernée */  
{  
  color : #AAA;  
}
```

## **Remarque**

On dispose de pseudo-formats en CSS pour changer le style des éléments requis (:required) et invalides (:invalid).

```
: required  
{  
  background - color : red;  
}
```

**Sélectionnez et stylisez chaque élément <p> qui est le premier enfant de son parent :**

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

**Un pseudo-élément est un mot-clé ajouté à un sélecteur qui permet de mettre en forme certaines parties de l'élément ciblé par la règle.**

**Ainsi, le pseudo-élément `::first-line` permettra de ne cibler que la première ligne d'un élément visé par le sélecteur.**

**`/* La première ligne de chaque élément <p>. */`**

```
p::first-line {  
    color: blue;  
    text-transform: uppercase;  
}
```

le pseudo-élément **::selection** permettra de ne cibler que la partie sélectionner d'un élément visé par le sélecteur .

```
p::selection {  
    color: white;  
    background-color: black;  
}
```

le pseudo-élément **::first-letter** Sélectionne et stylise la première lettre de chaque élément **<p>** :

```
p::first-letter {  
    font-size: 200%;  
    color: #8A2BE2;  
}
```



## This is a title

LLorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque semper, leo eget pulvinar elementum, sapien elit commodo lacus, non ultrices diam ligula sit amet nibh. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Mauris lobortis tincidunt nibh, eu elementum augue tincidunt ut. Donec consequat, mauris sit amet volutpat tempus, risus mauris blandit neque, sed pharetra erat turpis sit amet ipsum. Phasellus commodo lobortis facilisis. Sed ultrices, sem ut tristique ultricies, nibh purus tincidunt dui, sit amet fringilla

ligula lacus at nisi. Aliquam magna tellus, rutrum vitae vestibulum vel, semper id mauris. Curabitur in facilisis justo. Pellentesque fringilla pulvinar condimentum. Maecenas pellentesque mi condimentum magna laoreet sed porttitor leo sodales. Aenean et felis ipsum. Nam at libero non velit gravida aliquam quis ac dolor. Morbi imperdiet nisi vitae lorem rutrum pharetra.

## Hot items

- LLorem ipsum dolor sit amet,
- consectetur adipiscing elit.
- semper, leo eget pulvinar
- elementum, sapien elit

## New items

- LLorem ipsum dolor sit amet,
- consectetur adipiscing elit.
- semper, leo eget pulvinar
- elementum, sapien elit
- commodo lacus, non ultrices

## In the spotlight



LLorem ipsum lor sit amet

lor sit amet, consectetur adipiscing elit. Quisque semper, leo eget pulvinar elementum, sapien elit commodo lacus, non ultrices diam ligula sit amet nibh. Lorem ipsum dolor sit amet,

[read more](#)



LLorem ipsum lor sit amet

lor sit amet, consectetur adipiscing elit. Quisque semper, leo eget pulvinar elementum, sapien elit commodo lacus, non ultrices diam ligula sit amet nibh. Lorem ipsum dolor sit amet,

[read more](#)



LLorem ipsum lor sit amet

lor sit amet, consectetur adipiscing elit. Quisque semper, leo eget pulvinar elementum, sapien elit commodo lacus, non ultrices diam ligula sit amet nibh. Lorem ipsum dolor sit amet,

[read more](#)

## ***7- Le responsive design ou "design adaptable".***

## **7-1 Introduction**

**Il y a quelques années encore, on ne pouvait accéder à Internet et au web que grâce à des ordinateurs de bureau.**

**Avec la miniaturisation des composants et les capacités de connexion de plus en plus puissantes, cependant, de nouveaux appareils connectés ont vu le jour et les sites web peuvent maintenant être consultés à partir de nombreux appareils différents : ordinateurs de bureau, ordinateurs portables, tablettes, smartphones, montres connectées, etc.**

**Lorsqu'on parle de « responsive design » ou de « design adaptable » en français, on fait référence à l'idée selon laquelle un site web devrait s'afficher aussi bien sur un écran de PC que sur un écran de smartphone ou sur n'importe quel type d'appareil.**

- **La conception Web réactive (Responsive Design) rend la page Web compatible et adaptable à tous les appareils. Elle est basée uniquement sur HTML et CSS (elle n'est pas un programme ou un code JavaScript)**



- **Le Responsive web design consiste à utiliser les langages CSS et HTML pour redimensionner, masquer, réduire, agrandir ou déplacer le contenu d'une page pour s'adapter à l'écran d'affichage**



**Aujourd'hui, nous pouvons utiliser principalement trois méthodes pour répondre aux défis amenés par les différentes tailles d'écran. On peut :**

- **Créer une application dédiée pour les mobiles ;**
- **Créer une « copie mobile » de notre site en utilisant l'initiative AMP (« Accelerated Mobile Pages ») de Google ;**
- **Utiliser les Media Queries ou requêtes media.**

**L'idée de base du responsive design est qu'un site web devrait présenter les mêmes informations quel que soit l'appareil qui l'affiche, en les réarrangeant pour conserver la meilleure ergonomie possible. Nous allons pouvoir achever cela en utilisant les **Media Queries**.**

**La règle `@media`, introduite dans CSS2, permet de définir différentes règles de style pour différents types de médias.**

**Les medias queries dans CSS3 ont étendu l'idée des types de médias CSS2 : examiner la capacité du périphérique au lieu de chercher son type.**

**Les requêtes multimédias peuvent être utilisées pour vérifier de nombreuses propriétés, telles que :**

- **Largeur et hauteur de la fenêtre**
- **Largeur et hauteur de l'appareil**
- **Orientation (mode paysage ou portrait)**
- **Résolution**

## Syntaxe:

**@media** Il utilise la règle pour inclure un bloc de propriétés CSS uniquement si une certaine condition est vraie.

**@media media-type and (media-feature-rule)**

```
{  
  /* CSS rules go here */  
}
```

Cela consiste en:

• **media-type** : Un type de média, qui indique au navigateur à quel type de média ce code est destiné (**all, print, screen, speech**).

**media-feature-rule** : Une expression multimédia, qui est une règle ou un test qui doit être réussi pour que le CSS contenu soit appliqué.

**CSS rules** : Un ensemble de règles CSS qui seront appliquées si le test réussit et que le type de média est correct.

**Par exemple, la requête suivante permet d'appliquer des styles à condition que la largeur de la zone d'affichage soit inférieure à 1250px :**

```
@media screen and (max-width : 1250px) {  
  body {  
    background-color : red;  
  }  
}
```

L'exemple suivant change la couleur d'arrière-plan selon la règle suivante :

- Si la largeur de la fenêtre est **1024 px au minimum**, la couleur est le **Rouge**  
et
- Si la largeur de la fenêtre est **600 px au minimum et 1023 px au maximum**, la couleur est le **lightgreen**  
et
- Si la largeur de la fenêtre est **inférieure à 600 px**, la couleur reste **blanche**

```
@media screen and (min-width :1024px) {  
    body{  
        background-color :red;  
    }  
}  
  
@media screen and (min-width :600px) and (max-width :1023px){  
    body{  
        background-color :green;  
    }  
}  
  
@media screen and (min-width :320px) and (max-width :599px){  
    body{  
        background-color :rgb(248, 244, 3);  
    }  
}
```

**Dans cet exemple, media queries est utilisée pour créer un menu de navigation réactif dont la conception varie selon les tailles d'écran.**

```
.demo{  
  background-color :blue;  
}  
.demo a{  
  display: inline;  
  
  color :white;  
  text-align :center;  
  padding :10px;  
  text-decoration :none;  
}  
@media screen and (max-width : 600px)  
{  
  .demo{  
    margin: auto;  
    width :50%;  
  }  
  .demo a{  
    display: block;  
  
  }  
}
```

```
<div class = "demo">  
  <a href = "#">Home</a>  
  <a href = "#">About</a>  
  <a href = "#">Tutorials</a>  
  <a href = "#">QA</a>  
  <a href = "#">Videos</a>  
  <a href = "#">Contact</a>  
</div>
```

**/\* 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 les téléviseurs \*/**

**@media tv{**

**}**

**/\* Sur tous types d'écrans orientés verticalement \*/**

**@media all and (orientation: portrait){**

**}**

**TP**



- Remarque : Taille du texte réactif

**La taille du texte peut être définie avec une unité "vw", ce qui signifie la "largeur de la fenêtre".**

**De cette façon, la taille du texte suivra la taille de la fenêtre du navigateur :**

```
<h1 style="font-size :10vw">Hello World</h1>
```

## ***8- Les animations en CSS***

**CSS permet l'animation d'éléments HTML sans utiliser JavaScript**

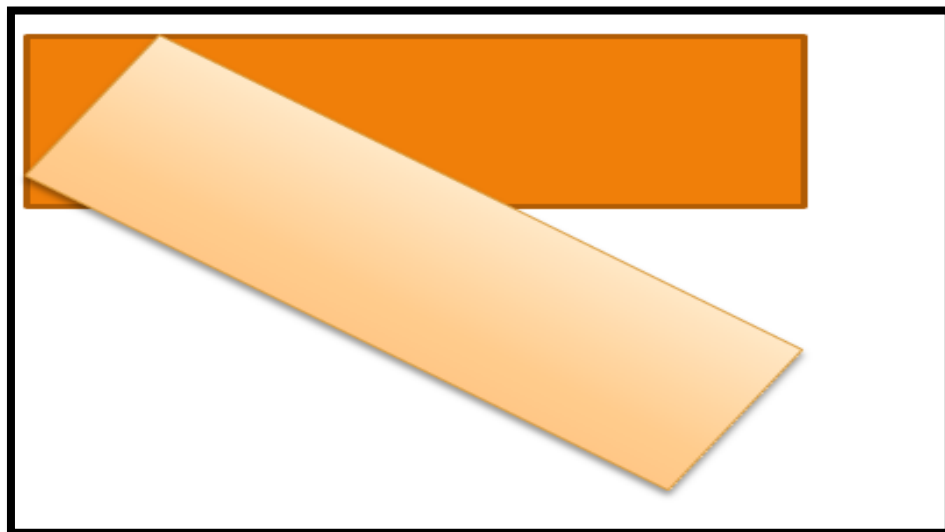
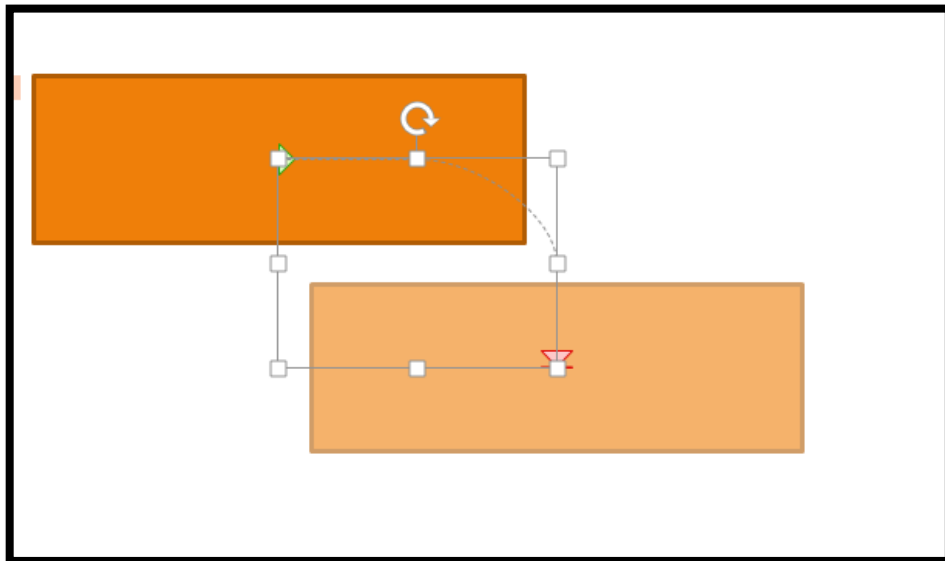
**Une animation permet à un élément de passer progressivement d'un style à un autre en modifiant les propriétés CSS.**

**Les types d'animations en CSS :**

- **Les transformations**
- **Les transitions**

**La propriété raccourcie (ou bien les propriétés détaillées correspondantes ) est utilisée pour créer une animation CSS. Elle permet de configurer la durée, le minutage et d'autres détails de l'animation.**

**L'apparence visuelle de l'animation est définie en utilisant des règles CSS de mise en forme au sein de la règle @keyframes.**



## 8-1 Les transformations 2D

Les transformations CSS 2D permettent de déplacer, faire pivoter, mettre à l'échelle et incliner des éléments.

### La méthode `translate()`

Exemple : déplacer l'élément `<div>` de 50 pixels vers la droite et de 100 pixels vers le bas par rapport à sa position actuelle :

```
div {  
    width : 300px;  
    height : 100px;  
    background-color : cyan;  
    border : 1px solid black;  
}  
div :hover{  
transform:translate(50px,100px);  
}
```

### La méthode `rotate()`

Exemple : faire pivoter l'élément `<div>` dans le sens des aiguilles d'une montre de 20 degrés :

```
div {  
    width : 300px;  
    height : 100px;  
    background-color : cyan;  
    border : 1px solid black;  
}  
div :hover{  
    transform : rotate(20deg);  
}
```

## La méthode `scale()`

Exemple :augmenter la taille de l'élément `<div>` de deux fois sur sa largeur et trois fois sa hauteur :

```
div {  
    width : 300px;  
    height : 100px;  
    background-color : cyan;  
    border : 1px solid black;  
}  
div :hover{  
    transform : scale(2,3);  
}
```

## La méthode `skew()`

Exemple :incliner l'élément `<div>` de 20 degrés le long de l'axe X et de 10 degrés le long de l'axe Y :

```
div {  
    width : 300px;  
    height : 100px;  
    background-color : cyan;  
    border : 1px solid black;  
}  
div :hover{  
    transform : skew(20deg,10deg);  
}
```

## La méthode `rotateX()`:

Permet de faire une rotation par rapport à l'axe X (on peut utiliser aussi `rotateY` et `rotateZ`).

Exemple : faire pivoter l'élément `<div>` par rapport à l'axe X de 60 degrés :

```
div {  
    width : 300px;  
    height : 100px;  
    background-color : cyan;  
    border : 1px solid black;  
}  
div :hover{  
    transform : rotateX(60deg);  
}
```

## 8-2 Transitions CSS

Les transitions CSS permettent de modifier les valeurs des propriétés sur une durée donnée.

**Exemple :** un élément `<div>` rouge de 100 px \* 100 px subit un effet de transition pour la propriété `width`, d'une durée de 2 secondes :

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

ou

```
div {  
  width : 100px;  
  height : 100px;  
  background : red;  
  transition : width 2s,height 4s;  
}
```

Lorsqu'un utilisateur passe la souris sur l'élément `<div>`, une nouvelle valeur pour la propriété `width` s'applique.

L'effet de transition démarre lorsque la propriété CSS spécifiée (largeur) change de valeur.

```
div :hover {  
  width : 300px;  
}
```



## 8-3 La règle @keyframes

La **règle @keyframes** permet aux auteurs de définir les étapes qui composent la séquence d'une animation CSS. Cela permet de contrôler une animation plus finement que ce qu'on pourrait obtenir avec les **transitions**.

Les propriétés détaillées rattachées à la propriété raccourcie animation sont :

**animation-name** : Cette propriété déclare **un nom** à l'animation (utilisé comme référence à l'animation pour la règle @keyframes ).

**animation-delay** : Cette propriété définit **le délai** entre le chargement de l'élément et démarrage de l'animation.

**animation-direction** : Cette propriété précise si l'animation doit alterner entre deux directions de progression (faire des allers-retours) ou recommencer au début à chaque cycle de répétition.

**animation-duration** : Cette propriété définit la durée d'un cycle de l'animation.

**animation-fill-mode** : Cette propriété indique les valeurs à appliquer aux propriétés avant et après l'exécution de l'animation.

**animation-iteration-count** : Cette propriété détermine le nombre de répétition de l'animation. Pour répéter une animation infiniment, On utilise la valeur infinite.

**animation-play-state** : Cette propriété permet d'interrompre (« pause ») ou de reprendre l'exécution d'une animation.

**animation-timing-function** : Cette propriété configure la fonction de minutage d'une animation.

La spécification d'un style CSS à l'intérieur de la règle **@keyframes** permet à une animation de passer progressivement du style actuel au nouveau style.

**Exemple 1:**

```
@keyframes example {  
  from {background-color : red;}  
  to {background-color : yellow;}  
}  
  
/* L'élément sur lequel s'appliquer l'animation */  
  
div {  
  width : 100px;  
  height : 100px;  
  background-color : red;  
  animation-name : example; /*définit le nom de l'animation*/  
  animation-duration : 4s; /*définit la durée de l'animation*/  
}
```

## Exemple 2:

```
@keyframes example {  
  0%   {background-color : red;}  
  25%  {background-color : yellow;}  
  50%  {background-color : blue;}  
  100% {background-color : green;}  
}  
/* The element to apply the animation to */  
div {  
  width : 100px;  
  height : 100px;  
  background-color : red;  
  animation-name : example;  
  animation-duration : 4s;  
}
```

## Exemple :

```
@keyframes example {  
  0%   {background-color :red; left :0px; top :0px;}  
  25%  {background-color :yellow; left :200px; top :0px;}  
  50%  {background-color :blue; left :200px; top :200px;}  
  75%  {background-color :green; left :0px; top :200px;}  
  100% {background-color :red; left :0px; top :0px;}  
}
```

```
div {  
  width : 100px;  
  height : 100px;  
  position : relative;  
  background-color : red;  
  animation-name : example;  
  animation-duration : 4s;  
}
```

## Remarque:

**Si plusieurs règles @keyframes existent avec le même nom, c'est la dernière qui est utilisée.**