Les feuilles de style CSS

1- Définition d'un style

Pour définir un style, il faut écrire:
- Les balises XHTML (séparées par des virgules s'il y en a plusieurs) auxquelles il doit s'appliquer.
- Les différents attributs du style enfermés entre deux accolades, chacuns séparés des autres par un point virgule.

```
Balise1, Balise2 {
propriété de style: valeur;
propriété de style: valeur;
}
```

Exemple:

```
H4, P {
font-family: Arial;
font-style: italic;
font-color: green;
}
```

Les paragraphes et les titres de niveau 4 seront en vert, en italique, et en police Arial.

Sélecteurs contextuels

Il est possible de sélectionner une balise en fonction des éléments qui l'entourent, grâce aux sélecteurs contextuels : **Eléments imbriqués**

```
Balise1 Balise2 {
/* style; */
}
```

Deux balises non séparées par une virgule permettent d'appliquer un style seulement si la Balise2, se trouve dans la Balise1. Exemple :

```
<div><p>texte1</p></div>
<p>texte2</p>
```

Le code suivant ne s'appliquera qu'à **texte1 et pas à texte2**.

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

Eléments consécutifs

```
Balise1+Balise2 {
/* style; */
}
```

Deux balises successives séparées d'un signe + permettent d'appliquer le style seulement si la Balise2, suit directement la Balise1. Exemple :

```
<i>texte</i><p>texte1</p>
<p>texte2</p>
```

Le code suivant ne s'appliquera qu'au **texte1**.

```
i+p {
color: blue;
}
```

Le sélecteur universel

Le sélecteur universel (*) sélectionne tous les éléments HTML de la page.

```
* {
  text-align: center;
  color: blue;
}
```

Les commentaires

Il est possible de placer des commentaires dans votre feuille de style sans qu'ils soient interprétés par les navigateurs, avec la syntaxe suivante :

```
<!DOCTYPE html>
<head>
<title>Titre de la page</title>
<style>
<!--
/* Ceci est un commentaire CSS */
Balise { propriété de style: valeur; propriété de style: valeur; }
Balise { propriété de style: valeur; propriété de style: valeur; }
-->
</style>
</head>
```

Les styles peuvent être incorporés dans un document Html ou Xhtml de trois manières différentes :

Dans l'en-tête de votre page

Les styles d'une page web peuvent être déclarés, dans les balises <**head**> et </**head**> grâce à la balise <**style**>.

```
<!DOCTYPE html>
<head>
<title>Titre de la page</title>
<style>
<!--
Vos styles ici
-->
</style>
</head>
```

Dans une balise html

Il est également possible de définir le style dans une balise html grâce à l'attribut **style**.

```
<!DOCTYPE>
<head>
<title>Titre de la page</title>
</head>
<body>
<balise style="propriete_style:valeur;" ></balise>
</body>
```

Dans un fichier externe

On peut aussi importer les feuilles des styles d'un fichier externe grâce à la balise <**link**> placée entre les balises <**head**> et </**head**> du document.

```
<!DOCTYPE>
<head>
<title>Titre de la page</title>
<link rel="stylesheet" type="text/css" href="style.css"/>
</head>
```

Il est aussi possible d'utiliser des styles différents pour des balises de même type, pour cela on utilise le concept de classe et d'identifiant.

Les classes

Pour définir une classe, il faut préciser la balise, la faire suivre d'un point, puis le nom de classe que vous souhaitez.

```
p.vert{
font: Verdana 14px;
color: #336600; }
```

Pour affecter une classe à une balise HTML, il suffit de lui ajouter un attribut **class** avec le nom de la classe choisie.

```
<p class="vert"> Texte en vert </p>
```

Les classes universelles

Si aucune balise n'est précisée, la classe sera prise en compte comme une classe universelle et pourra être utilisée sur n'importe qu'elle balise.

```
.vert{
font: Verdana 14px;
color: #336600;
}
```

Les sélecteurs identifiants

Pour les sélecteurs identifiants, la procédure est la même que pour les classes. La seule différence est qu'ils permettent de faire référence à un élément unique d'une page. Définition d'un identifiant:

```
p#vert{
font: Verdana 14px;
color: #336600;
}
```

Appel d'un identifiant:

```
<p id="vert"> Texte en vert </p>
```

Les pseudo-classes de texte

:first-line permet d'appliquer un style à la première ligne d'un paragraphe.

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

:first-letter permet d'appliquer un style à la première lettre d'un paragraphe.

```
p:first-letter {
font-size: 200%;
font-weight: bold;
}
```

Les pseudo-classes de lien

:link permet de définir le style des liens hypertextes n'ayant pas encore été consultés.

```
a:link {
font: Verdana 14px;
color: #336600;
}
```

:visited permet de définir le style des liens hypertextes déjà visités.

```
a:visited {
font: Verdana 14px;
color: #330000;
}
```

Les pseudo-classes dynamiques

:hover permet d'affecter un style à la balise sélectionnée lors d'un survol par le curseur de la souris.

```
a:hover {
font: Verdana 14px;
color: #336600;}
```

:active permet de définir un style à la balise sélectionnée lorsque l'utilisateur clique sur l'élément.

```
a:active {
font: Verdana 14px;
color: #330000; }
```

2 : Liste des propriétés

La liste complète des propriétés et recommandations concernant les feuilles de style, En voici une sélection :

2.1. Les styles de police

font-family
définit un nom de police ou une famille de police <nom> ou <famille>
police précise (Arial, Times, Helvetica..) ou famille(serif, sans-serif, cursive, fantasy, monospace)
H3 {font-family: Arial;}

font-style
définit le style de l'écriture : normal ou italique ou oblique
H3 {font-style: italic;}

font-weight
définit l'épaisseur de la police : normal ou bold ou bolder ou lighter ou valeur numérique
P {font-weight: bold;} // police en gras

font-size
définit la taille de la police : xx-small ou x-small ou small ou médium ou large ou x-large ou xx-large ou larger ou smaller ou taille précise en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
P {font-size: 12pt;}

font-variant
définit une variante par rapport à la normale
normal ou small-caps
P {font-variant: small-caps;}

font
raccourci pour les différentes propriétés de police
P {font: bold italic;}

2.2. Les styles du texte

text-align
définit l'alignement du texte left ou justify ou
center ou right
H1 {text-align: center;}

text-indent
définit un retrait dans la première ligne d'un bloc de texte, souvent utilisé avec <P>, n'oubliez pas dans ce cas </P>. spécifié en inches (in) ou en centimètres (cm) ou en pixels (px)
P {text-indent: 1cm;}

text-decoration

définit une décoration (?) du texte, soit barré, clignotant, blink ou underline ou line-through ou verline ou none
A:visited {text-decoration: none blink;} // Lien non souligné et clignotant

text-transform
définit la casse du texte (majuscule, minuscule)
uppercase (met les caractères en majuscules) ou lowercase (met les caractères en minuscules)
ou capitalize (met le premier caractère en majuscule)
P {text-transform: uppercase;}

color
définit la couleur du texte, par exemple en hexadécimal
H3 {color: #000080;}

word-spacing
définit l'espace entre les mots en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
P {word-spacing: 5pt;}

letter-spacing
définit l'espace entre les lettres spécifié en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
P {letter-spacing: 2pt;}

line-height
définit l'interligne soit l'espace entre les lignes du texte en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
P {line-height: 10pt;}

width
détermine la longueur d'un élément de texte ou d'une image en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
H1 {width: 200px;}

height
détermine la hauteur d'un élément de texte ou d'une image en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
H1 {height: 100px;}

white-space
espace ou blanc
normal ou pre ou nowrap
PRE {white-space: pre ;}

2.3. Les arrière-plans

background-color
définit la couleur de l'arrière-plan couleur (par exemple en hexadécimal) ou transparent
H1 {background-color: #000000 ;}
background-image
définit l'image de l'arrière-plan
URL de l'image
BODY {background-image: url(images/image.gif);}

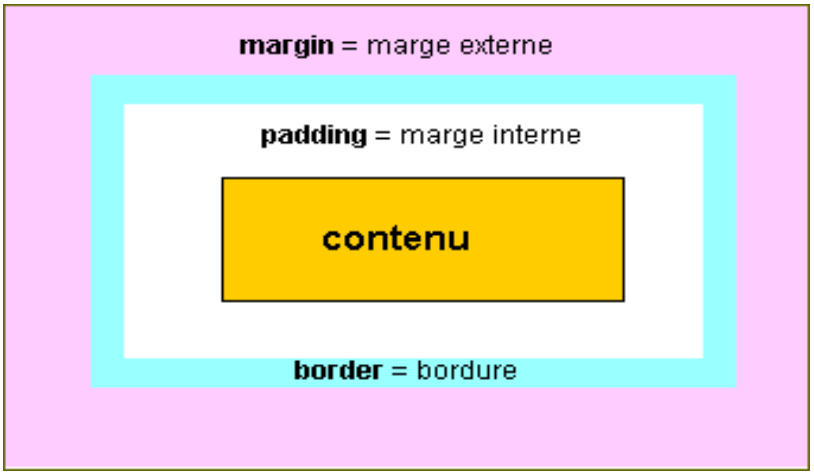
background-repeat
définit la façon de répéter l'image d'arrière-plan
repeat ou no-repeat ou repeat-x (x = nombre de répétitions horizontales) ou
repeat-y (y = nombre de répétitions verticales)
P {background-image: url(images/image.gif); background-repeat: repeat-4;}

background-attachment
spécifie si l'image d'arrière-plan reste fixe avec les déplacements de l'écran scroll ou fixed
```
BODY {background-image: url(images /image.gif);
        background-attachement: fixed;}
```

background-position
spécifie la position de l'image d'arrière-plan par rapport au coin
supérieur gauche de la fenêtre {1, 2}
{top ou center ou bottom , left ou center ou right} ou en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
```
BODY {background-image: url(images/image.gif);
        background-position: right top;}
```

background
raccourci pour les différentes propriétés d'arrière-plan
```
P {background: url(images\image.gif) fixed repeat;}
```

2.4. Les marges permettent de définir l'espace entre les éléments HTML



margin-top
détermine la valeur de la marge supérieure
en unité de longueur ou auto
```
P { margin-top: 5px ; }
```

margin-right
détermine la valeur de la marge droite
en unité de longueur ou auto
```
P { margin-right: 10px ; }
```

margin-bottom
détermine la valeur de la marge inférieure
en unité de longueur ou auto
```
P { margin-bottom: 5px ; }
```

margin-left
détermine la valeur de la marge gauche
en unité de longueur ou auto
```
P { margin-left: 10px ; }
```

margin
regroupe les différentes propriétés de la marge en suivant l'ordre **(top, right, bottom et left)**
```
P { margin: 5px 10px 5px 10px ;} pareil que : P { margin: 5px 10px ;}
```

2.5. Les bords et les "enrobages"

border-top-width
donne l'épaisseur du bord supérieur thin ou medium ou thick ou spécifié par l'auteur
H3 {border-top-width: thin;}

border-right-width
donne l'épaisseur du bord droit thin ou medium ou thick ou spécifié par l'auteur
H3 {border-right-width: medium;}

border-bottom-width
donne l'épaisseur du bord inférieur thin ou medium ou thick ou spécifié par l'auteur
H3 {border-bottom-width: thick;}

border-left-width
donne l'épaisseur du bord gauche thin ou medium ou thick ou spécifié par l'auteur
H3 {border-left-width: 0.5cm;}

border-width
regroupe les différentes propriétés de border-width

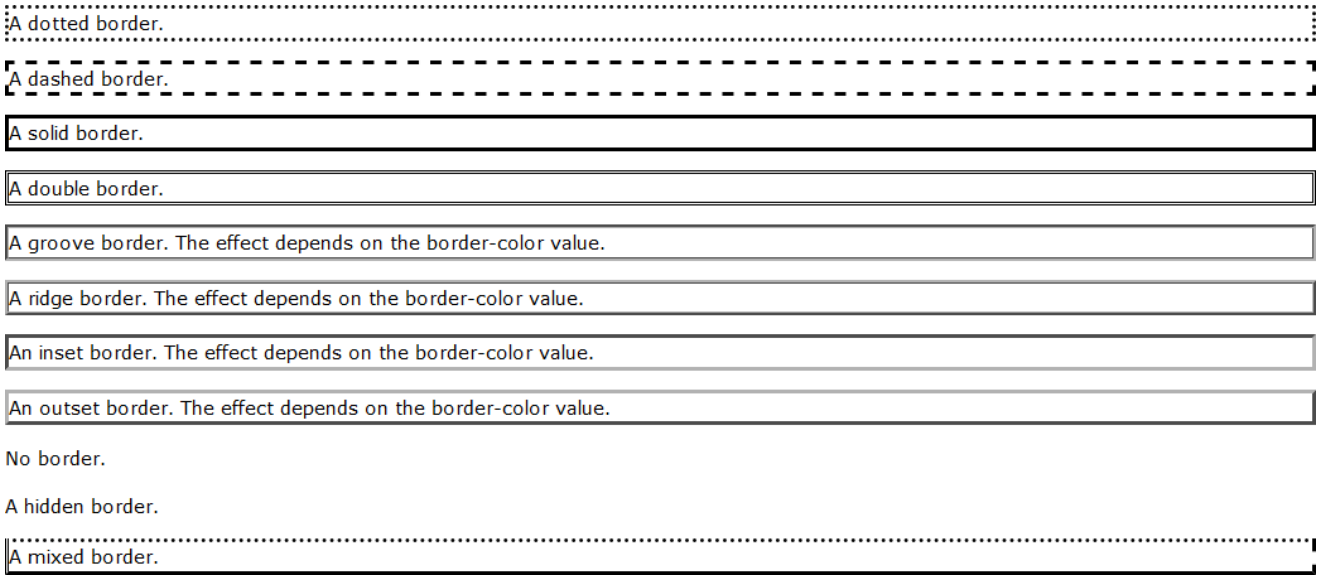
border-color
détermine la couleur de la bordure
H3 {border-color: yellow ;}

border-style
détermine le style du trait de la bordure none, solid(solide), dotted(pointillé), dashed, double, groove(rainurée 3D), ridge(striée 3D), inset(incrustée 3D), outset(bordure de départ 3D)

border
regroupe toutes les propriétés des bords

```
<!DOCTYPE html>
<head><meta charset="utf-8">
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style:
hidden;}p.mix {border-style:
dotted dashed solid double;}
</style>
</head>
<body>
<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
```

```
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>
</body>
</html>
```




```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8">
<style>
p.normal {
  border: 2px solid red;}
p.round1 {
  border: 2px solid red;
border-radius: 5px;}
p.round2 {
  border: 2px solid red;
border-radius: 8px;}
p.round3 {
  border: 2px solid red;
  border-radius: 12px;}
</style>
</head>
<body>
<h2>Propriété border-radius </h2>
<p>Cette propriété spécifie le type de bordure à afficher:</p>

<p class="normal">Normal border</p>
<p class="round1">Round border</p>
<p class="round2">Rounder border</p>
<p class="round3">Roundest border</p>

</body>
```

</html>

padding-top
valeur de remplissage haut entre l'élément (contenu) et le bord en points (pt), inches (in), centimètres (cm),pixels (px) ou pourcentage (%)
H3 {padding-top: 3px;}

padding-right
valeur de remplissage droite entre l'élément et le bord en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
H3 {padding-right: 3px;}

padding-bottom
valeur de remplissage bas entre l'élément et le bord en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
H3 {padding-bottom: 3px;}

padding-left
valeur de remplissage gauche entre l'élément et le bord en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%) H3 {padding-left: 3px;}

padding
regroupe les différentes propriétés de remplissage

2.6. Les listes

list-style-type
détermine le type de puces ou de numérotation disc ou circle ou square decimal ou lower-roman ou upper-roman ou lower-alpha ou upper-alpha
OL {list-style-type: square;}

list-style-image
permet de remplacer les puces par une image url ou none
OL {list-style-image : url('images/image.gif');}

list-style-position

spécifie si les puces sont à l'intérieur ou à l'extérieur du texte inside ou outside
UL {list-style-position: inside ;}

list-style
regroupe toutes les propriétés de liste

2.7. width et max-width

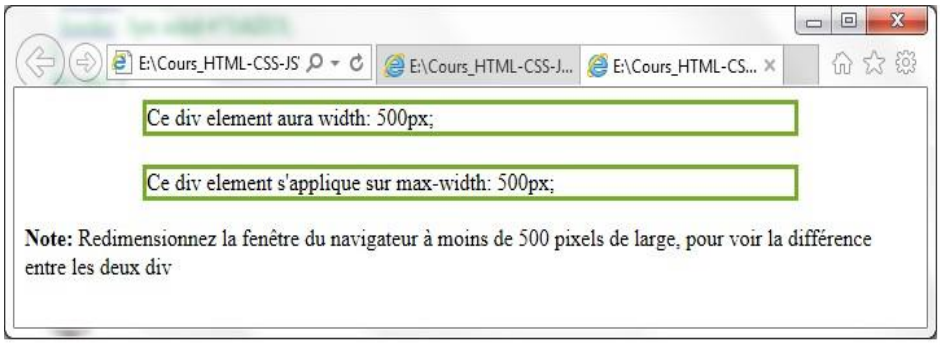
Comme mentionné dans le chapitre précédent; un élément de niveau bloc occupe toujours toute la largeur disponible (s'étend à gauche et à droite aussi loin que possible).

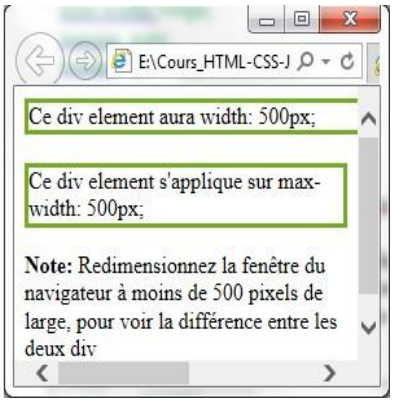
Définir la largeur d'un élément au niveau du bloc l'empêchera de s'étendre sur les bords de son conteneur. Ensuite, vous pouvez définir les **marges sur auto pour centrer l'élément horizontalement dans son conteneur.** L'élément prendra la largeur spécifiée et l'espace restant sera divisé de manière égale entre les deux marges:

L'utilisation de max-width améliorera la gestion des petites fenêtres par le navigateur. Ceci est important pour rendre un site utilisable sur de petits appareils:

Remarque: le problème avec width dans ce ci-dessous se produit, lorsque la fenêtre du navigateur est inférieure à la largeur de l'élément. Le navigateur ajoute ensuite une barre de défilement horizontale à la page.

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8">
<style>
div.ex1 {
width:500px;
margin: auto;
  border: 3px solid #73AD21;
}
div.ex2 {
  max-width:500px;
  margin: auto;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
<div class="ex1">Ce div element aura width: 500px;</div>
<br>
<div class="ex2">Ce div element s'applique sur max-width: 500px;</div>
<p><strong>Note:</strong> Redimensionnez la fenêtre du navigateur à moins de 500 pixels de large, pour voir la différence entre les deux div</p>
</body>
</html>
```





Propriétés d'affichage	Description	Valeurs
display:	Spécifie la manière dont un élément est affiché.	none bloc inline inline-block
visibility:	Spécifie si un élément est visible.	visible hidden collapse
clip:	Spécifie la zone visible d'un élément.	
overflow:	Gére les dépassements de blocs.	visible hidden scroll auto
z-index:	Spécifie la position d'empilement d'un bloc.	Nombre négatif ou positif de -100 à 100.

Exemple 1 CSS

```
<html>
<head>
<title>Exemple 1 - CSS</title>
<style type="text/css">
.liste {
list-style-type:none;
width: 100px;
}
.liste li {
border: 1px black solid;
background-color: gray;
text-align:center;
}
.liste a {
display:block;
color:red ;
text-decoration:none ;
}
.liste a:hover {
background-color:yellow;
font-weight: bold;}

</style>
</head>
<body>
<ul class="liste">
```

```
  <li><a href="#">Chapitre 1</a></li>
  <li><a href="#">Chapitre 2</a></li>
  <li><a href="#">Chapitre 3</a></li>
  <li><a href="#">Chapitre 4</a></li>
</ul>
</body> </html>
```

Menu changeant la couleur au passage de la souris :

Exemple 2 CSS

```
<html>
<head>
<title>Exemple 2 - CSS</title>
</head>
<html><body>
<table width="220" border="0" cellspacing="0" style="text-align:center;">
 <tr >
    <td width="33%" style="border-top: 2px solid black; border-left: 2px solid black;"> 1  </td>
   <td width="33%">2</td>
   <td width="33%" style="border-top: 2px solid black; border-right: 2px solid black;"> 3 </td>
 </tr>
 <tr>
  <td width="33%">4</td>
  <td width="33%">5</td>
  <td width="33%">6</td>
 </tr>
 <tr>
  <td width="33%" style="border-bottom: 2px solid black; border-left: 2px solid black;"> 1 </td>
  <td width="33%">2</td>
  <td width="33%" style="border-bottom: 2px solid black; border-right: 2px solid black;">3</td>
 </tr>
</table>
</body>
</html>
```

1	2	3
4	5	6
1	2	3

Exemple 3 CSS

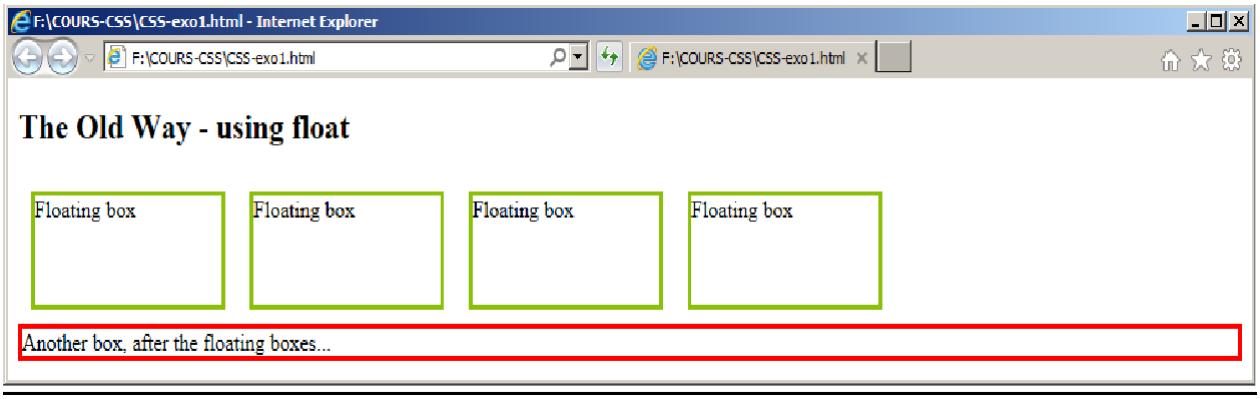
```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
list-style: square inside url("images/sqpurple.gif");
}
```

```
</style>
</head>
<body>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
</body>
</html>
```

- Coffee
- Tea
- Coca Cola

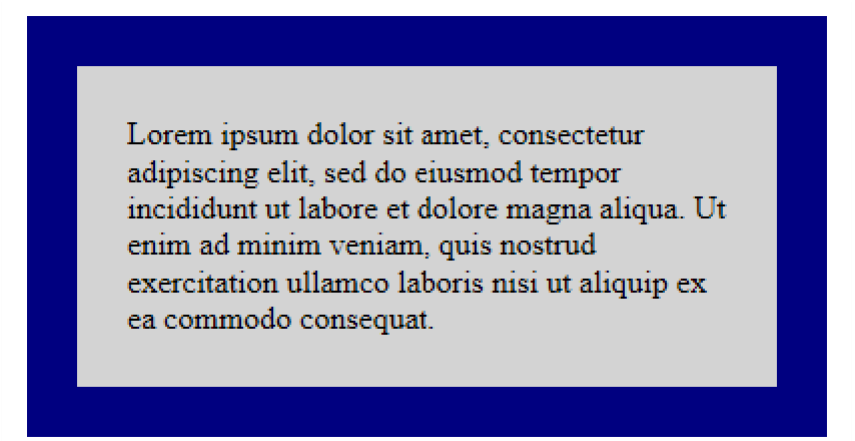
Exemple 4 CSS

```
<!DOCTYPE html>
<html>
<head>
<title>Exemple 4 - CSS</title>
<style>
.floating-box {
float: left;
width: 150px;
height: 75px;
margin: 10px;
border: 3px solid #8AC007;
}
.after-box {
clear: left;
border: 3px solid red;
}
</style>
</head>
<body>
<h2>The Old Way - using float</h2>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="after-box">Another box, after the floating boxes...</div>
</body>
</html>
```



Exemple 5 CSS

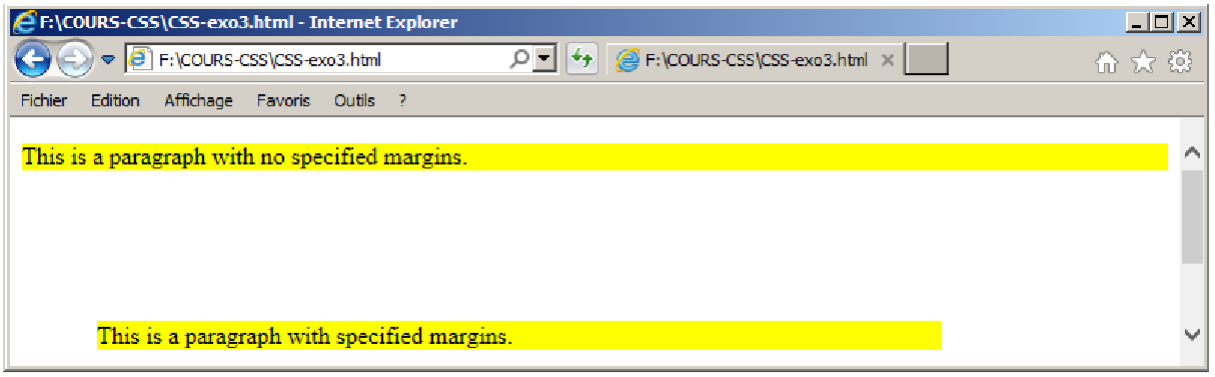
```
<!DOCTYPE html>
<html>
<head>
<title>Exemple 5 - CSS</title>
<style>
div {
background-color: lightgrey;
width: 300px;
padding: 25px;
border: 25px solid navy;
margin: 25px;
}
</style>
</head>
<body>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</div>
</body></html>
```



Exemple 6 CSS

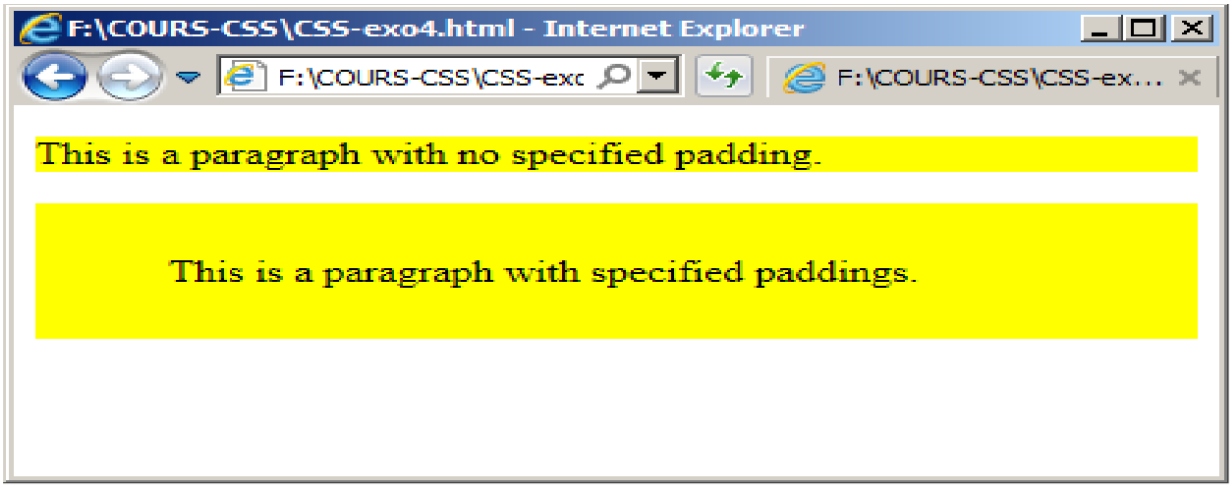
```
<!DOCTYPE html>
<html>
<head>
<title>Exemple 6 - CSS</title>
<style>
p {
background-color: yellow;
}
p.ex {
margin: 100px 150px 100px 50px;
}
/* équivaut à :
p.ex {
margin-top: 100px;
margin-right: 150px;
margin-bottom: 100px;
margin-left: 50px;
}
*/
</style>
```

```
</head>
<body>
<p>This is a paragraph with no specified margins.</p>
<p class="ex">This is a paragraph with specified margins.</p>
</body></html>
```



Exemple 7 CSS

```
<!DOCTYPE html>
<html>
<head>
<title>Exemple 7 - CSS</title>
<style>
p {
background-color: yellow;
}
p.padding {
padding: 25px 50px;
/* padding-top: 25px;
   padding-right: 50px;
   padding-bottom: 25px;
   padding-left: 50px;
*/
}
</style>
</head>
<body>
<p>This is a paragraph with no specified padding.</p>
<p class="padding">This is a paragraph with specified paddings.</p>
</body></html>
```



2.8. Les propriétés de position :

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

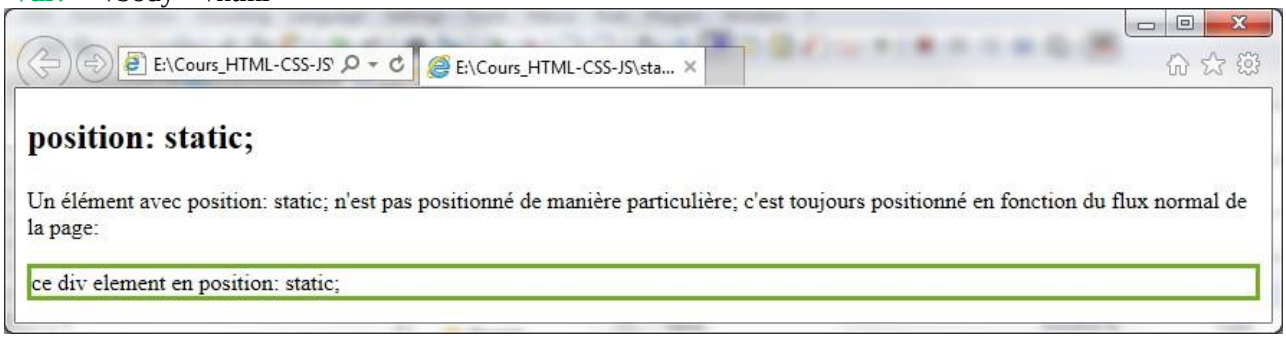
position: static;

Les éléments HTML sont positionnés statiques par défaut.

Les éléments positionnés statiques ne sont pas affectés par les propriétés top, bottom, left et right.

Un élément avec position: static; n'est pas positionné de manière particulière; il est toujours positionné en fonction du flux normal de la page:

```
<!DOCTYPE html>

<html>

<head> <meta charset="utf-8">

<style>

div.static {

position: static;

  border: 3px solid #73AD21;

}

</style></head>

<body>

<h2>position: static;</h2>

<p>Un élément avec position: static; n'est pas positionné de manière particulière; c'est

toujours positionné en fonction du flux normal de la page:</p>

<div class="static">

  ce div element en position: static;

</div> </body></html>
```

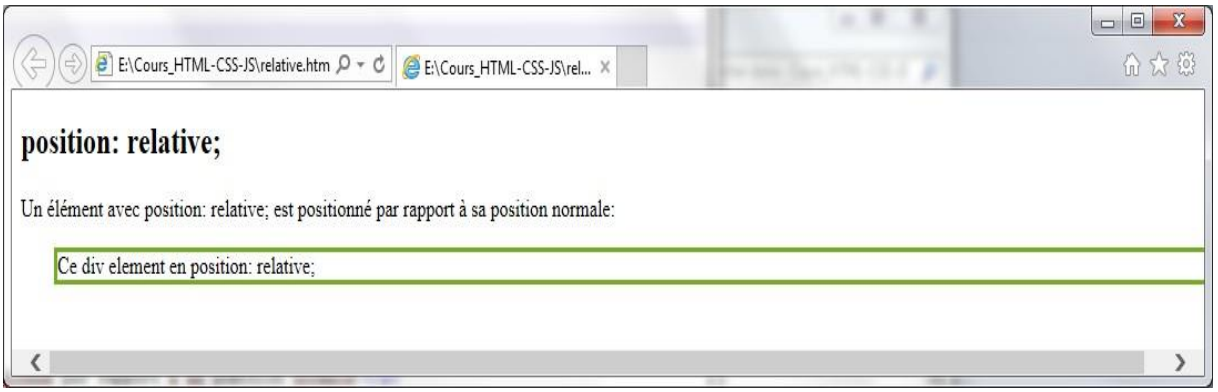


position: relative;

Un élément avec position: relative; est positionné par rapport à sa position normale.

Si vous définissez les propriétés top, right, bottom et left d'un élément relativement positionné, il s'éloignera de sa position normale dans le code. Les autres contenus ne seront pas ajustés pour s'insérer dans les espaces laissés par l'élément.

```
<!DOCTYPE html>

<html>

<head>

<style>

div.relative {

position: relative;

left: 30px;

  border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: relative;</h2>

<p> Un élément avec position: relative; est positionné par rapport à sa position normale:</p>

<div class="relative">

Ce div element en position: relative;

</div>

</body></html>
```



position: fixed;

Un élément avec position: fixed; est positionné par rapport à la fenêtre d'affichage, ce qui signifie qu'il reste toujours au même endroit, même si la page est défilée. Les propriétés top, right, bottom et left sont utilisées pour positionner l'élément.

Un élément fixed ne laisse pas d'espace dans la page où il aurait normalement été situé.

```
<!DOCTYPE html>

<html>

<head> <meta charset="utf-8">

<style>

div.fixed {

position: fixed;

bottom: 0;

 right: 0;

 width: 300px;

 border: 3px solid #73AD21;

}

</style></head>

<body>

<h2>position: fixed;</h2>

<p>Un élément avec position: fixed; est positionné par rapport à la fenêtre d'affichage, ce qui signifie qu'il reste toujours au même endroit même si la page est déroulée:</p>

<div class="fixed">

Ce div element en position: fixed;

</div>

</body></html>
```

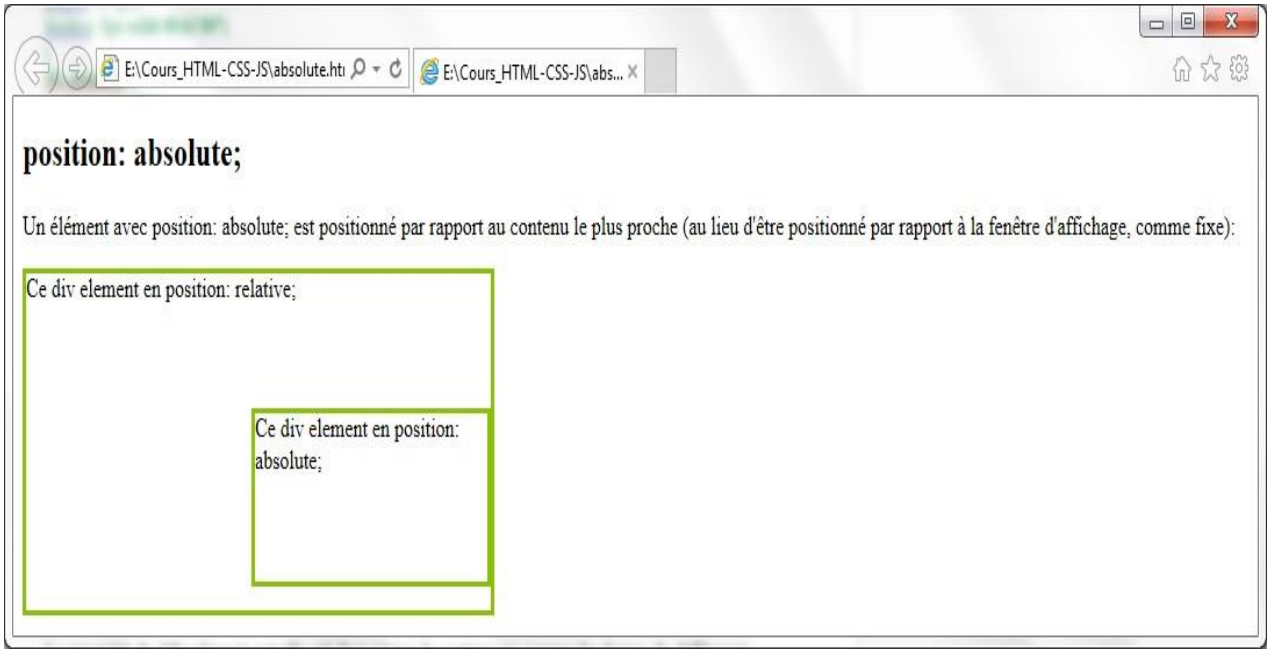
position: absolute;

Un élément avec position: absolute; est positionné par rapport à l'ancêtre positionné le plus proche (au lieu de positionné par rapport à la fenêtre d'affichage, comme fixe).

Pourtant; Si un élément absolu positionné n'a pas d'ancêtres positionnés, il utilise le corps du document (la fenêtre d'affichage) et se déplace avec le défilement de page.

Remarque: Un élément "positionné" est un élément dont la position est tout sauf static.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
div.relative {
position: relative;
width: 400px;
height: 200px;
border: 3px solid #8AC007;
}
div.absolute {
position: absolute;
top: 80px;
right: 0;
width: 200px;
height: 100px;
border: 3px solid #8AC007;
}
</style> </head>
<body>
<h2>position: absolute;</h2>
<p> Un élément avec position: absolute; est positionné par rapport au contenu le plus proche (au lieu d'être positionné par rapport à la fenêtre d'affichage, comme fixe):</p>
<div class="relative">Ce div element en position: relative;
<div class="absolute">Ce div element en position: absolute; </div>
</div>
</body> </html>
```



position: sticky; (collant)

Un élément avec position:sticky; est positionné en fonction de la position de défilement de l'utilisateur.Un élément sticky bascule entre relative et fixed, selon la position de défilement. Il est positionné de manière relative jusqu'à ce qu'une position de décalage donnée soit atteinte dans la fenêtre - puis il "sticks" en place (comme position:fixed)

```
<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="utf-8">

<title>mise en page des boites | CSS3 </title>

<style type="text/css">

@import url(http://fonts.googleapis.com/css?family=Mr+Bedfort);

html,

body {

scroll-behavior: smooth;

}

h1 {

color: #99b92c;

font-size: 4em;

font-weight: normal;

text-shadow: 1px 1px 1px #cccccc;

background:white;

border-bottom:2px dotted #ccc;

margin:0;

padding:0.8em;

}

.main-container{

max-width:600px;
```

```
margin:0 auto;
border:solid 10px green;
padding:10px;
margin-top:40px;
}
.main-container *{
padding:10px;
background:#aaa;
border:dashed 5px #000;
}
.main-container * + *{
margin-top:20px;
}
.main-header{
height:50px;
background:#aaa;
border-color:red;
}
.main-content{
min-height:1000px;
}
.main-header{
position:sticky;
top:0;
}
</style>
<body>
<h1>Boite sticky...</h1>
<section>
<div>
```

```
<article class="article text-component">

<p>La position CSS sticky a une très bonne prise en charge du navigateur.

il a fallu assez de temps pour que la prise en charge du navigateur se produise,et au

moment où il l'a fait, la fonctionnalité a été oubliée.

</p>

</article>

</div>

</section>

<section>

<aside class="main-container">

<header class="main-header">HEADER</header>

<div class="main-content">MAIN CONTENT</div>

<footer class="main-footer">FOOTER</footer>

</aside>

</section>

</body>

</html>
```

2.9. La propriété Overflow

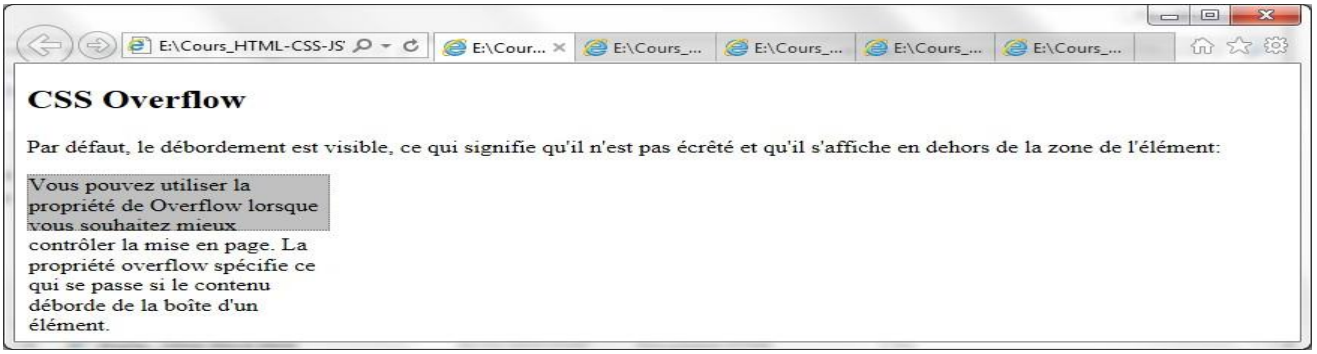
La propriété de débordement spécifie s'il faut couper le contenu ou ajouter des barres de défilement lorsque le contenu d'un élément est trop volumineux pour tenir dans la zone spécifiée.

La propriété Overflow a les valeurs suivantes:
• visible (par défaut). Le débordement n'est pas coupé. Le contenu s'affiche en dehors de la zone de l'élément
• hidden - Le débordement est tronqué et le reste du contenu sera invisible.
• scroll - Le débordement est tronqué et une barre de défilement est ajoutée pour afficher le reste du contenu.
• auto - Similaire au scroll, mais ajoute des barres de défilement uniquement lorsque cela est nécessaire

Exemple : overflow: visible

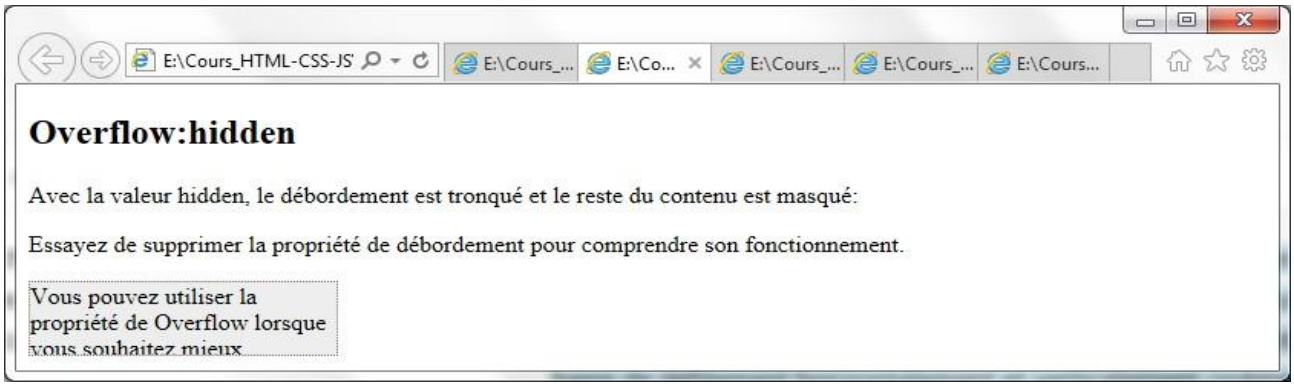
```
<!DOCTYPE html>

<html>

<head> <meta charset="utf-8"><style>
```

```
div {
  background-color: silver;
  width: 200px;
  height: 50px;
  border: 1px dotted black;
  overflow: visible;}
</style></head>
<body>
<h2> Overflow:visible</h2>
<p>Par défaut, le débordement est visible, ce qui signifie qu'il n'est pas écrêté et qu'il s'affiche en dehors de la zone de l'élément:</p>
<div>Vous pouvez utiliser la propriété de débordement lorsque vous souhaitez mieux contrôler la mise en page. La propriété overflow spécifie ce qui se passe si le contenu déborde de la boîte d'un élément.</div></body></html>
```



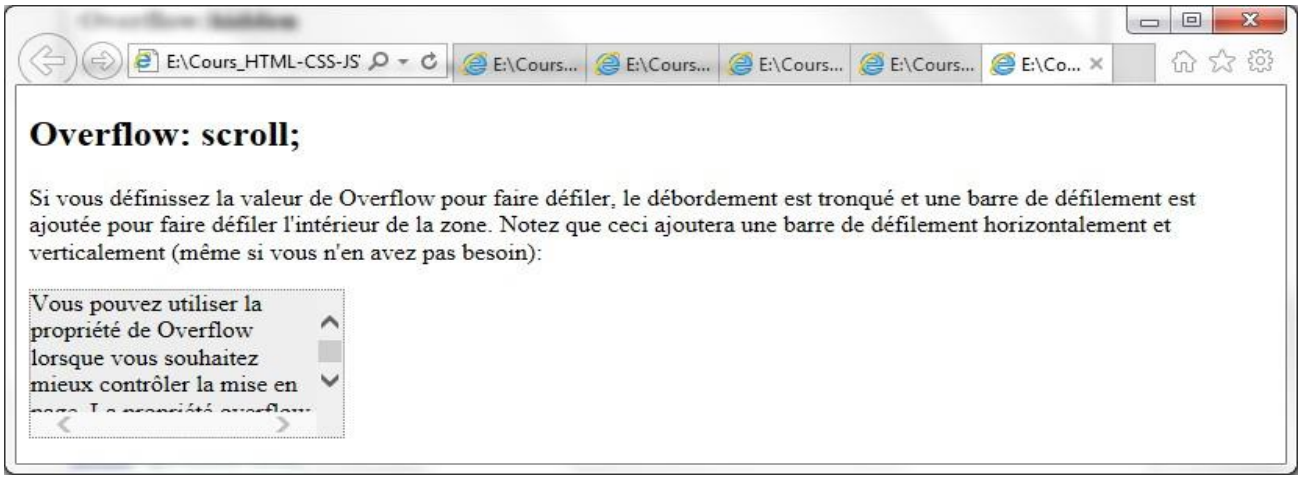
Exemple : overflow: hidden

```
div {
  background-color: silver;
  width: 200px;
  height: 50px;
  border: 1px dotted black;
  overflow: hidden; }
```



Exemple : overflow: scroll

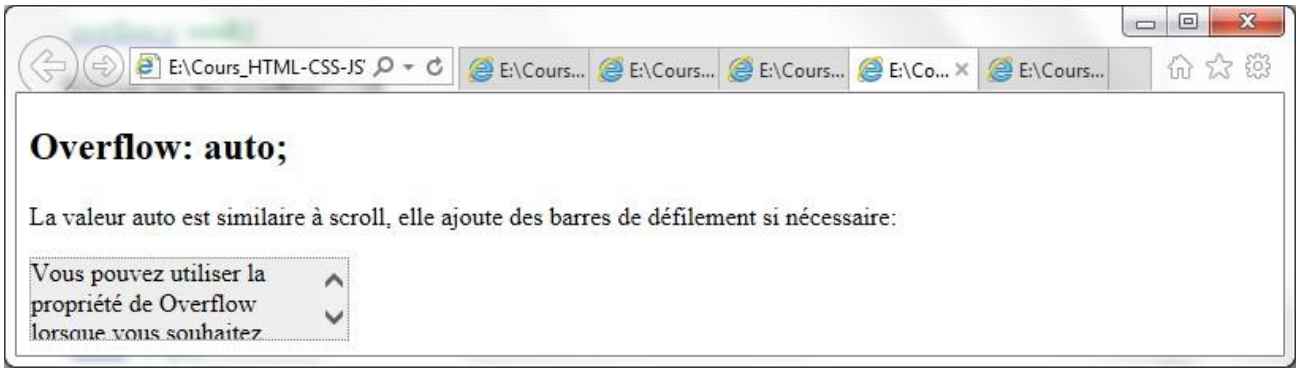
```
div {
  background-color: silver;
  width: 200px;
  height: 100px;
  border: 1px dotted black;
  overflow: scroll;}
```




```
div {
  background-color: silver;
  width: 200px;
  height: 50px;
  border: 1px dotted black;
  overflow-x: hidden;
  overflow-y: scroll;}
```

Exemple : overflow: auto

```
div {
  background-color: silver;
  width: 200px;
  height: 50px;
  border: 1px dotted black;
  overflow: auto;}
```



2.10. La propriété float

La propriété float est utilisée pour positionner et mettre en forme le contenu, par ex. laissez une image flotter à gauche du texte dans un conteneur.

La propriété float peut avoir l'une des valeurs suivantes:

• left - L'élément flotte à gauche de son conteneur

• right - L'élément flotte à la droite de son conteneur

• none – par défaut L'élément ne flotte pas (sera affiché exactement là où il se trouve dans le code).

• inherit - L'élément hérite de la valeur float de son parent

2.11. La propriété clear

La propriété clear spécifie quels éléments peuvent flotter à côté de l'élément effacé et de quel côté.

La propriété clear peut avoir l'une des valeurs suivantes:

• none - Autorise les éléments flottants des deux côtés. C'est par défaut

• left - Aucun élément flottant autorisé sur le côté gauche

• right- Aucun élément flottant autorisé sur le côté droit

• both - Aucun élément flottant autorisé à gauche ou à droite

• inherit - L'élément hérite de la valeur vide de son parent

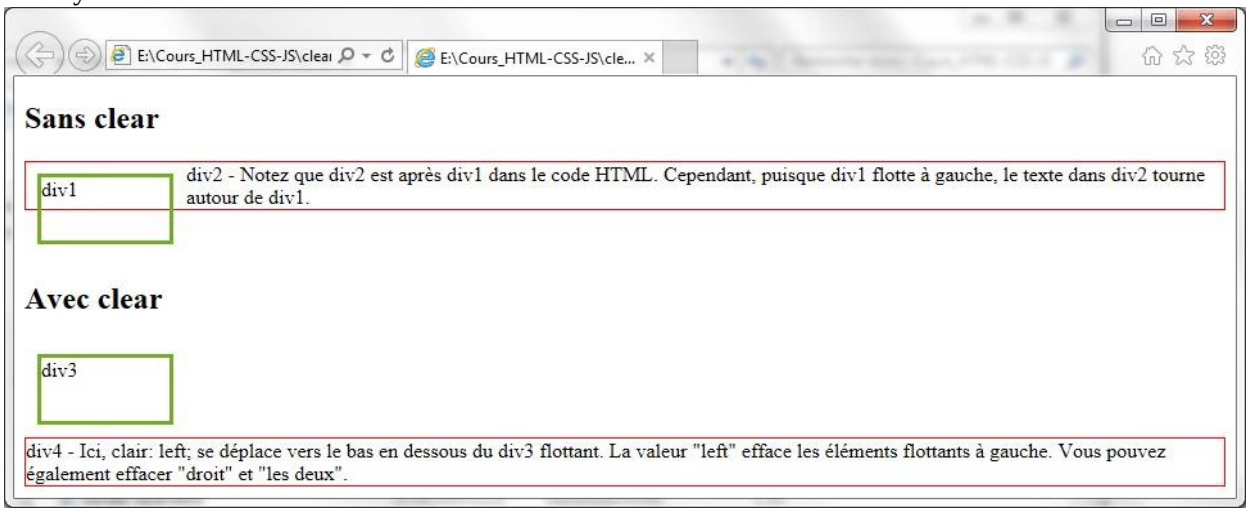
Le moyen le plus courant d'utiliser la propriété clear est après avoir utilisé une propriété float sur un élément.

Lorsque vous supprimez des éléments flottants, vous devez faire correspondre ce qui est clear à celui-ci: **Si un élément est flotté à gauche, vous devez effacer à gauche**. Votre élément flottant continuera à flotter, mais l'élément clear apparaîtra en dessous de celui-ci sur la page Web.

L'exemple suivant efface le flottant à gauche. Signifie qu'aucun élément flottant n'est autorisé du côté gauche (de la div):

```
<!DOCTYPE html>
<html>
<head><style>
.div1 {
  float: left;
  width:  100px;
  height:   50px;
  margin: 10px;
  border: 3px solid #73AD21;}
.div2 {
  border: 1px solid red;}
.div3 {
  float: left;
  width: 100px;
  height: 50px;
```

```
  margin: 10px;
  border: 3px solid #73AD21;}
.div4 {
  border: 1px solid red;
  clear: left;}
</style></head>
<body>
<h2>Sans clear</h2>
<div class="div1">div1</div>
<div class="div2"> div2 - Notez que div2 est après div1 dans le code HTML. Cependant, puisque div1 flotte vers la gauche, le texte dans div2 circule autour de div1
</div>
<br><br>
<h2>Avec clear</h2>
<div class="div3">div3</div>
<div class="div4"> div4 - Ici, clear: left; déplace div4 en dessous du div3 flottant. La valeur "left" efface les éléments flottant vers la gauche. Vous pouvez également effacer "droite" et "les deux".
</div>
</body></html>
```



2.12. La propriété Display

Par rapport à display: inline, la principale différence est que display: inline-block permet de définir une largeur et une hauteur sur l'élément.

De plus, avec **display: inline-block, les marges supérieures et inférieures sont respectées, mais avec display: inline, elles ne le sont pas.**

Par rapport à display: block, la différence majeure est que **display: inline-block n'ajoute pas de saut de ligne après l'élément**, ce qui permet à l'élément de s'asseoir à côté d'autres éléments.

L'exemple suivant montre les différents comportements d'affichage:

display : inline, display: inline-block et display: block:

```
<!DOCTYPE html>
<html><head><style>
span.a {
  display: inline; /* the default for span */
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
background-color: yellow; }
span.b {
  display: inline-block;
  width:  100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
background-color: yellow; }
span.c {
  display: block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow; }
</style></head>
<body>
<h1>The display Property</h1>
<h2>display: inline</h2>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

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

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

<h2>display: block</h2>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

</body></html>

The display Property

display: inline

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

display: inline-block

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

display: block

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat.

Aliquam

venenatis

gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

<!DOCTYPE html>

<html>

<head><style>

.nav {

  background-color: yellow;

  list-style-type: none;

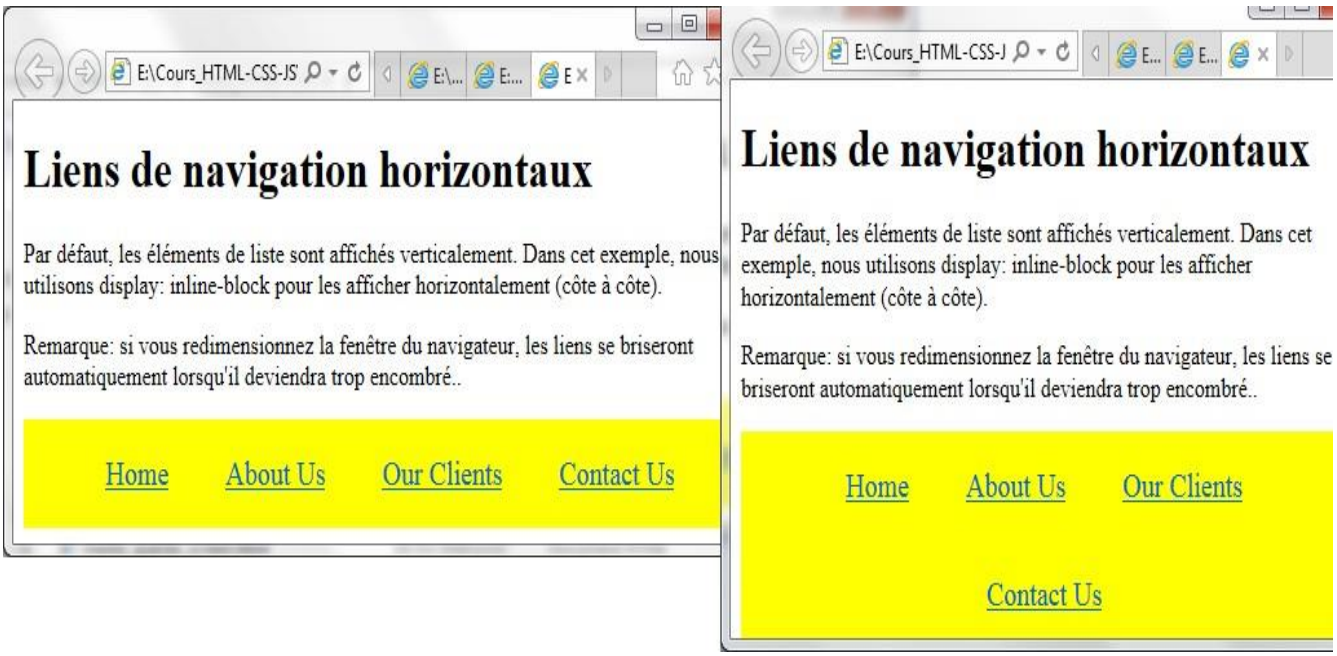
  text-align: center;

  margin: 0;

  padding: 0;}

.nav li {

```
  display: inline-block;
  font-size: 20px;
  padding: 20px;}
</style></head>
<body>
<h1>Liens de navigation horizontaux</h1>
<p>Par défaut, les éléments de la liste sont affichés verticalement. Dans cet exemple, nous utilisons display: inline-block pour les afficher horizontalement (côte à côte).</p>
 <p>Remarque : Si vous redimensionnez la fenêtre du navigateur, les liens se rompent automatiquement lorsqu'ils deviennent trop encombrés.</p>
<ul class="nav">
  <li><a href="#home">Home</a></li>
  <li><a href="#about">About Us</a></li>
  <li><a href="#clients">Our Clients</a></li>
  <li><a href="#contact">Contact Us</a></li>
</ul>
</body></html>
```



2.13. CSS Opacity / Transparency

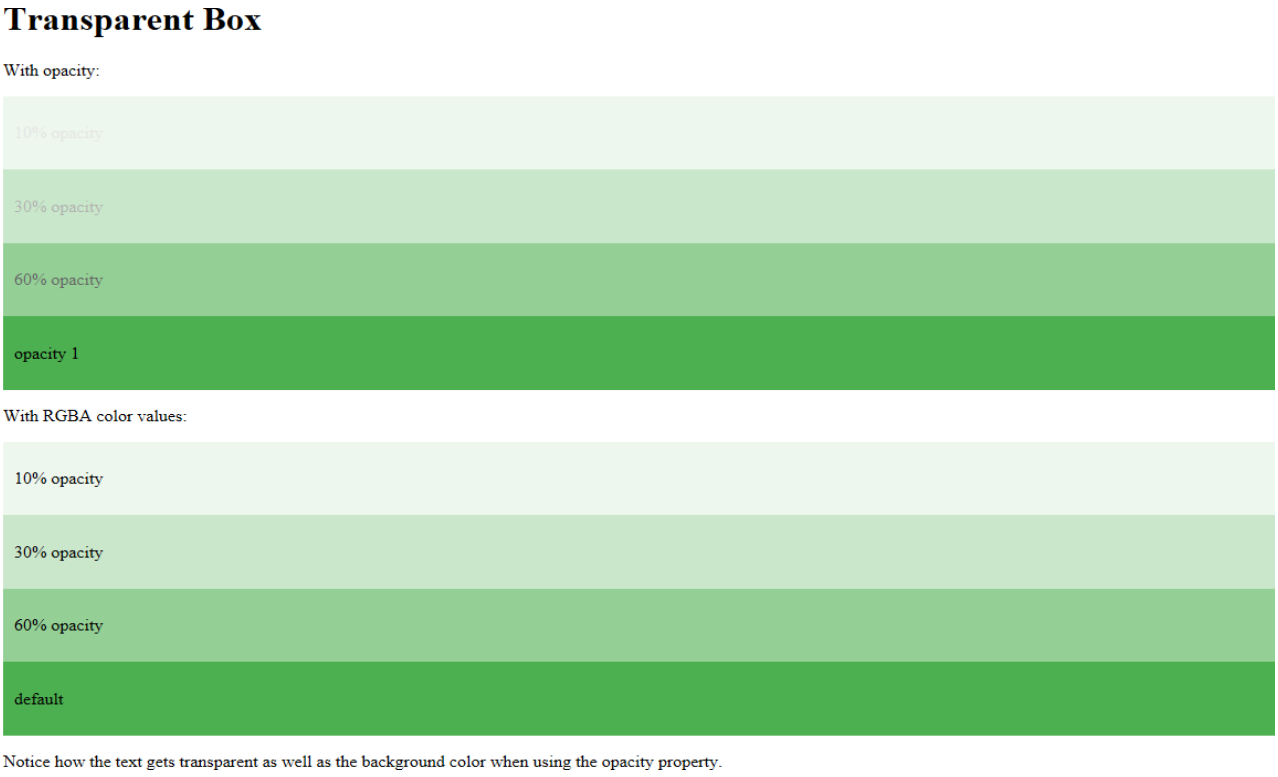
La propriété opacity spécifie l'opacité / la transparence d'un élément.

Remarque: Internet Explorer 8 et les versions antérieures utilisent le filtre: alpha (opacity = x). Le x peut prendre une valeur comprise entre 0 et 100. Une valeur inférieure rend l'élément plus transparent.

```
<!DOCTYPE html>

<html>

<head><style>

img {

  opacity: 0.5;

  filter: alpha(opacity=50); /* Pour IE8 et antérieur */}

img:hover {

  opacity: 1.0;

  filter: alpha(opacity=100); /* Pour IE8 et antérieur */}

</style></head>

<body>

<h1>Transparence de l'image</h1>

<p>La propriété opacity est souvent utilisée avec le sélecteur: hover pour modifier l'opacité au survol de la souris:</p>

<img src="img_forest.jpg" alt="Forest" width="170" height="100">

<img src="img_mountains.jpg" alt="Mountains" width="170" height="100">

<img src="img_5terre.jpg" alt="Italy" width="170" height="100">

<p><b>Note:</b> Dans IE, il faut ajouter un !DOCTYPE pour que le sélecteur: hover fonctionne sur d'autres éléments que l'élément a.</p>

</body></html>
```

Si vous ne souhaitez pas appliquer d'opacité aux éléments, comme dans notre exemple ci-dessus, utilisez les **valeurs de couleur RGBA.** L'exemple suivant définit l'opacité de la couleur d'arrière-plan et non du texte:

```
<!DOCTYPE html>
<html><head><style>
div {
  background: rgb(76, 175, 80);
  padding: 10px;}
div.first {
  background: rgba(76, 175, 80, 0.1);}
div.second {
  background: rgba(76, 175, 80, 0.3);}
div.third {
  background: rgba(76, 175, 80, 0.6);}
</style></head>
<body>
<h1>Transparent Box</h1>
<p>With opacity:</p>
<div style="opacity:0.1;"><p>10% opacity</p></div>
<div style="opacity:0.3;"><p>30% opacity</p></div>
<div style="opacity:0.6;"><p>60% opacity</p></div>
<div><p>opacity 1</p></div>
<p>With RGBA color values:</p>
<div class="first"><p>10% opacity</p></div>
<div class="second"><p>30% opacity</p></div>
<div class="third"><p>60% opacity</p></div>
<div><p>default</p></div>
<p> Remarquez comment le texte devient transparent ainsi que la couleur d'arrière-plan lors de l'utilisation de la propriété opacity.</p>
</body></html>
```



2.14. CSS Responsive : Media Query

Comment utiliser les media query CSS pour créer une navigation réactive.

Une media query consiste en un type de média (*mediatype*) et peut contenir une ou plusieurs expressions, dont la résolution est true ou false.

```
@media not|only mediatype and (expressions) {
  CSS-Code;
}
```

Le résultat est true si le type de média spécifié correspond au type de périphérique sur lequel le document est affiché et si toutes les expressions de la requête de média sont vraies. Lorsqu'une media query est vraie, la feuille de style ou les règles de style correspondantes sont appliquées conformémentaux règles en cascade normales.

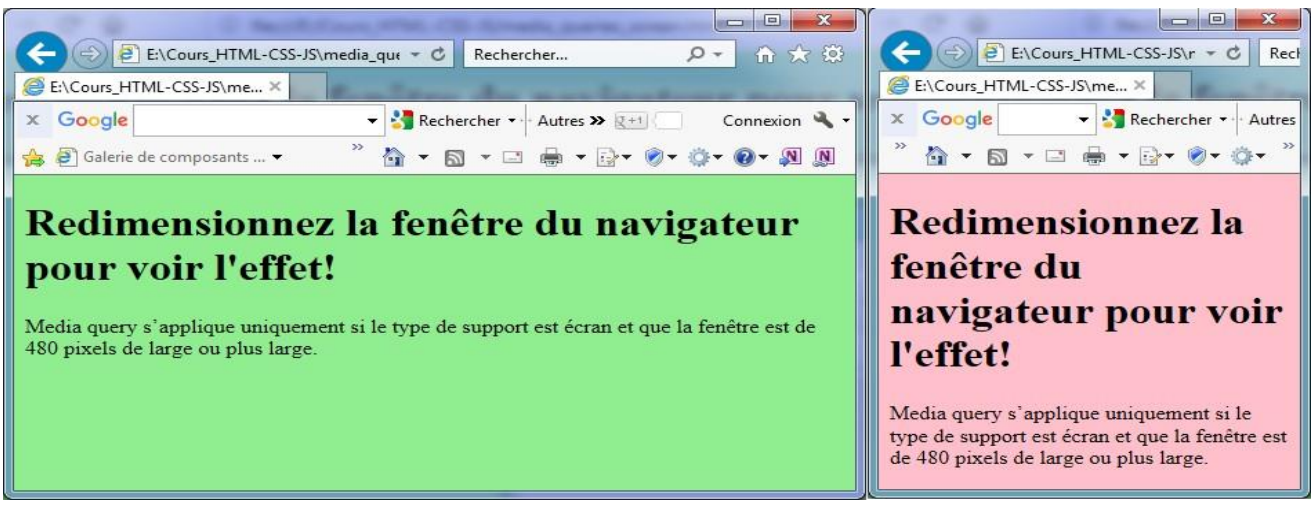
À moins que vous n'utilisiez les opérateurs not ou only, **le type de support est facultatif et le type all sera impliqué.**
Vous pouvez également avoir des feuilles de style pour différents médias :
<link rel="stylesheet" media="*mediatype* and|not|only (*expressions*)" href="*print.css*">

Valeur de *mediatype*	**Description**
all	Utilisé pour tous les types de périphériques
print	Utilisé pour les imprimantes
screen	Utilisé pour les écrans d'ordinateur, tablettes, smartphones, etc.
speech	Utilisé pour les lecteurs d'écran qui "reads" la page à voix haute

Exemple Media Query :

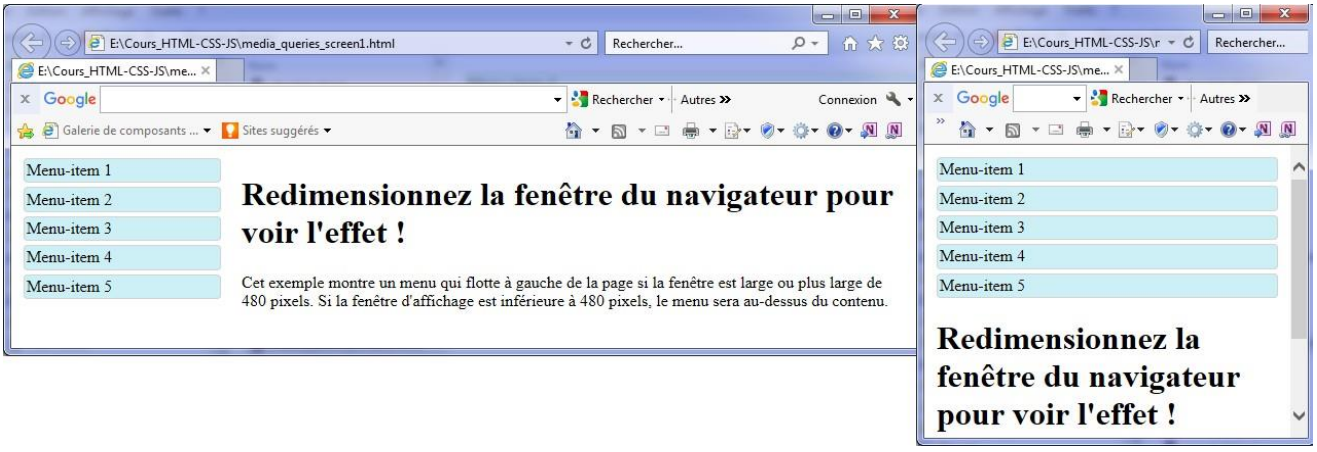
L'exemple suivant modifie la couleur d'arrière-plan en lightgreen si la fenêtre est large ou plus large de 600 pixels (si la fenêtre est inférieure à 600 pixels, la couleur d'arrière-plan sera rose):

```
<!DOCTYPE html>
<html><head>
<meta charset="utf-8">
<style>
body {
  background-color: pink;
        }
@media screen and (min-width: 600px) {
  body {
    background-color: lightgreen;
          }
}
</style>
</head>
<body>
<h1>Redimensionnez la fenêtre du navigateur pour voir l'effet!</h1>
<p>Media query s’applique uniquement si le type de support est écran et que la fenêtre est de 600 pixels de large ou plus large.</p>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
.wrapper {overflow: auto;}
#main {margin-left: 4px;}
#leftsidebar {
  float: none;
  width: auto;
}
#menulist {
  margin: 0;
  padding: 0;
}
.menuitem  {
  background: #CDF0F6;
  border: 1px solid #d4d4d4;
  border-radius: 4px;
  list-style-type: none;
  margin: 4px;
  padding: 2px;}
@media screen and (min-width: 480px) {
  #leftsidebar {width: 200px; float: left;}
  #main {margin-left: 216px;}
}
</style>
</head>
<body>
<div class="wrapper">
  <div id="leftsidebar">
```

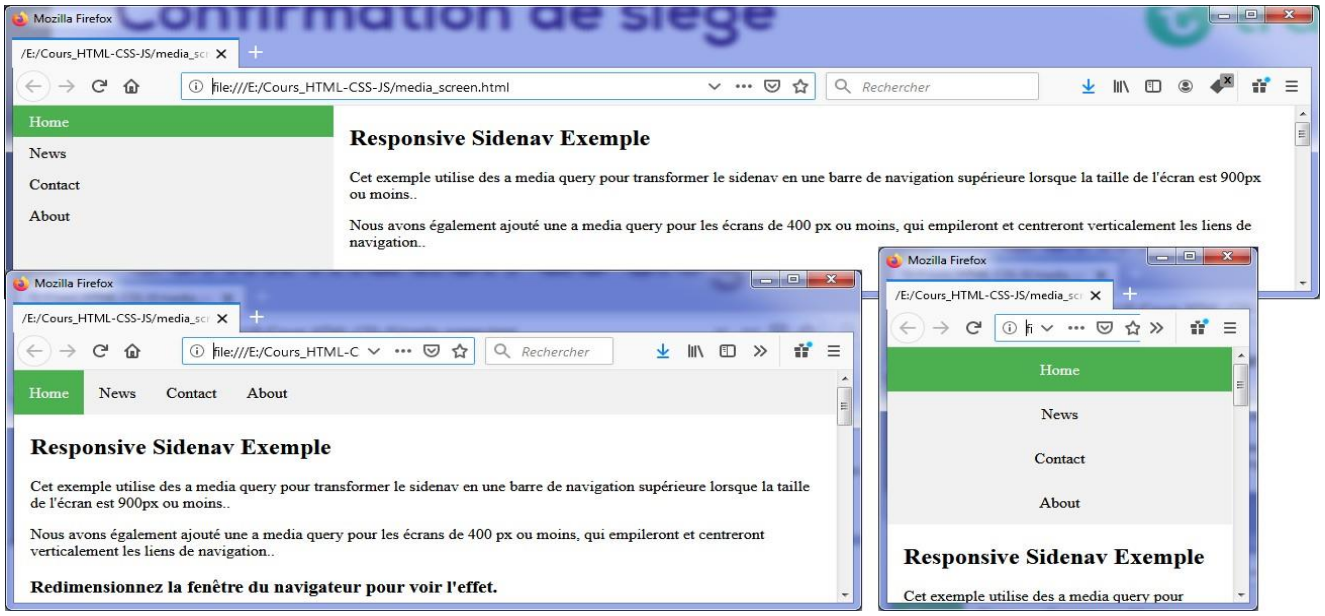
```
    <ul id="menulist">
      <li class="menuitem">Menu-item 1</li>
      <li class="menuitem">Menu-item 2</li>
      <li class="menuitem">Menu-item 3</li>
      <li class="menuitem">Menu-item 4</li>
      <li class="menuitem">Menu-item 5</li>
    </ul>
  </div>
    <div id="main">
    <h1>Redimensionnez la fenêtre du navigateur pour voir l'effet !</h1>
    <p>Cet exemple montre un menu qui flotte à gauche de la page si la fenêtre est large ou plus large de 480 pixels. Si la fenêtre d'affichage est inférieure à 480 pixels, le menu sera au-dessus du contenu.</p>
    </div>
</div></body></html>
```




```
<!DOCTYPE html>
<html><head>
<meta charset="utf-8">
<style>
body {margin: 0;}
ul.sidenav {
  list-style-type: none;
  margin: 0;
  padding: 0;
```

```
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
overflow: auto;}
ul.sidenav li a {
display: block;
color: #000;
padding: 8px 16px;
  text-decoration: none;}
 ul.sidenav li a.active {
 background-color: #4CAF50;
 color: white;
}
ul.sidenav li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
div.content {
  margin-left: 25%;
  padding: 1px 16px;
  height: 1000px;
}
@media screen and (max-width: 900px) {
  ul.sidenav {
    width: 100%;
    height: auto;
    position: relative;
  }
    ul.sidenav li a {
    float: left;
```

```
    padding: 15px;
  }
    div.content {margin-left: 0;}
}
@media screen and (max-width: 600px) {
  ul.sidenav li a {
    text-align: center;
    float: none;
  }
}
</style></head>
<body>
<ul class="sidenav">
  <li><a class="active" href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
<div class="content">
  <h2>Responsive Sidenav Exemple</h2>
  <p>Cet exemple utilise des a media query pour transformer le sidenav en une barre de navigation supérieure lorsque la taille de l'écran est 900px ou moins..</p>
  <p>Nous avons également ajouté une a media query pour les écrans de 600 px ou moins, qui empileront et centreront verticalement les liens de navigation..</p>
  <h3>Redimensionnez la fenêtre du navigateur pour voir l'effet.</h3>
</div>
</body></html>
```



2.15. Responsive Form

Redimensionnez la fenêtre du navigateur pour voir l'effet. Lorsque la largeur de l'écran est inférieure à 600 pixels, placez les deux colonnes les unes sur les autres et non les unes à côté des autres.

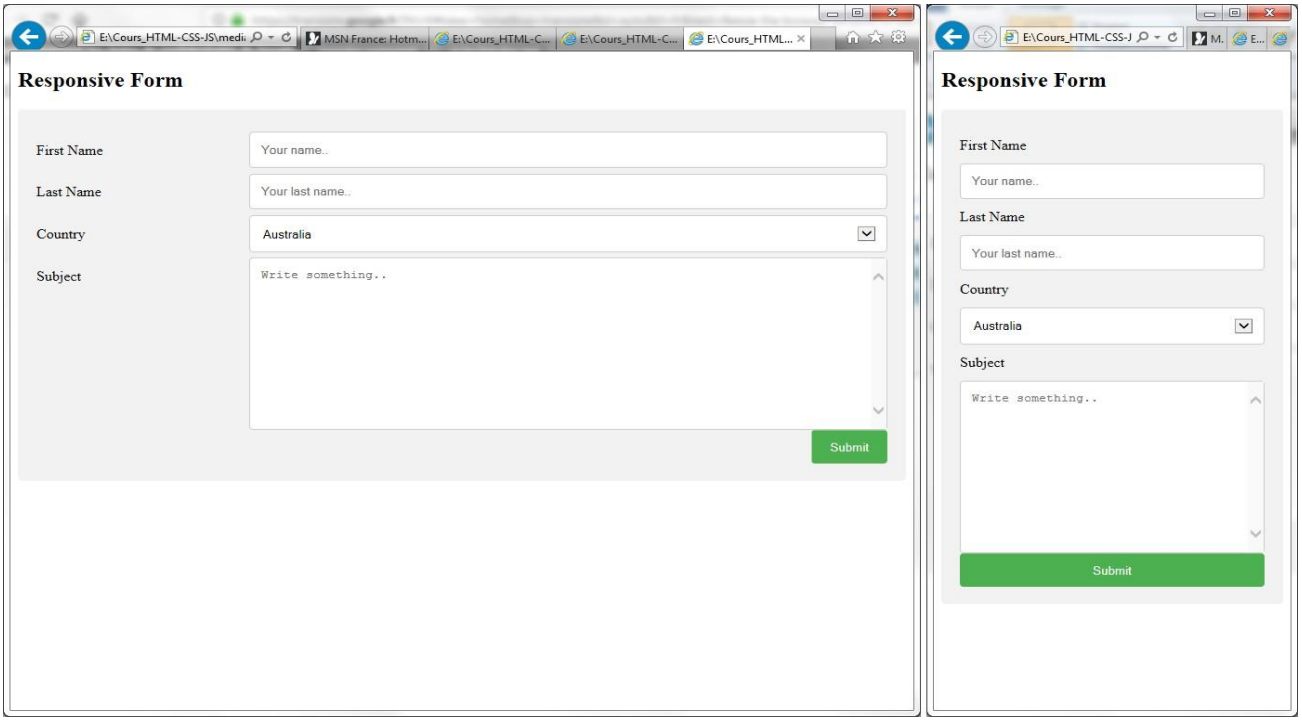
```
<!DOCTYPE html>

<html><head>

<style>

* {

  box-sizing: border-box;

}

input[type=text], select, textarea {

  width: 100%;

  padding:  12px;

  border: 1px solid #ccc;

  border-radius: 4px;

  resize: vertical;}

label {

  padding: 12px 12px 12px 0;

  display: inline-block;}
```

```
input[type=submit] {
  background-color: #4CAF50;
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  float: right;
}
input[type=submit]:hover {
  background-color: #45a049;
}
.container {
  border-radius: 5px;
  background-color: #f2f2f2;
  padding: 20px;
}
.col-25 {
  float: left;
  width: 25%;
  margin-top: 6px;
}
.col-75 {
  float: left;
  width: 75%;
  margin-top: 6px;
}
```

```
/* Effacer les flotteurs après les colonnes */
.row:after {
  content: "";
  display: table;
  clear: both;
}
/* Disposition sensible: lorsque l'écran a moins de 600 pixels de large, empilez les deux colonnes plutôt que les unes à côté des autres */
@media screen and (max-width: 600px) {
  .col-25, .col-75, input[type=submit] {
    width: 100%;
    margin-top: 0;
  }
}
</style></head>
<body>
<h2>Responsive Form</h2>
<div class="container">
  <form action="/action_page.php">
  <div class="row">
    <div class="col-25">
      <label for="fname">First Name</label>
    </div>
    <div class="col-75">
      <input type="text" id="fname" name="firstname" placeholder="Your name..">
    </div>
  </div>
  <div class="row">
```

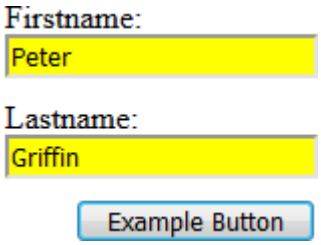
```
  <div class="col-25">
    <label for="lname">Last Name</label>
  </div>
  <div class="col-75">
    <input type="text" id="lname" name="lastname" placeholder="Your last name..">
  </div>
 </div>
 <div class="row">
  <div class="col-25">
    <label for="country">Country</label>
  </div>
  <div class="col-75">
    <select id="country" name="country">
      <option value="australia">Australia</option>
      <option value="canada">Canada</option>
      <option value="usa">USA</option>
    </select>
  </div>
 </div>
 <div class="row">
  <div class="col-25">
    <label for="subject">Subject</label>
  </div>
  <div class="col-75">
    <textarea id="subject" name="subject" placeholder="Write something.."
style="height:200px"></textarea>
  </div>
 </div>
```

```
 <div class="row">
  <input type="submit" value="Submit">
 </div>
 </form>
</div>
</body>
</html>
```




```
<!DOCTYPE html>
<html>
<head>
<style>
input[type="text"] {
width: 150px;
display: block;
  margin-bottom: 10px;
  background-color: yellow;
}
input[type="button"] {
```

```
  width: 120px;
  margin-left: 35px;
  display: block;
}
</style>
</head>
<body>
<form name="input" action="" method="get">
  Firstname:<input type="text" name="Name" placeholder="Peter" size="20">
  Lastname:<input type="text" name="Name" placeholder="Griffin" size="20">
  <input type="button" value="Example Button">
</form>
</body></html>
```



2.16. CSS Tooltip

HTML: Utilisez un élément conteneur (comme) et ajoutez-lui la classe "tooltip". Lorsque l'utilisateur survole ce , le texte de l'info-bulle s'affiche.

Le texte de l'info-bulle est placé dans un élément en ligne (comme) avec class = "tooltiptext".

CSS: La position d'utilisation de la classe de l'info-bulle "tooltip": relative, nécessaire pour positionner le texte de l'info-bulle ("tooltiptext" = position: absolute).

La classe tooltiptext contient le texte de l'info-bulle. Il est masqué par défaut et sera visible en survol (voir ci-dessous). Nous y avons également ajouté quelques styles de base: largeur de 120 pixels, couleur de fond noir, couleur de texte blanche, texte centré et remplissage supérieur et inférieur de 5 pixels.

La propriété CSS border-radius est utilisée pour ajouter des coins arrondis au texte de l'info-bulle.

Le sélecteur: hover est utilisé pour afficher le texte de l'info-bulle lorsque l'utilisateur déplace la souris sur la avec class = "tooltip".

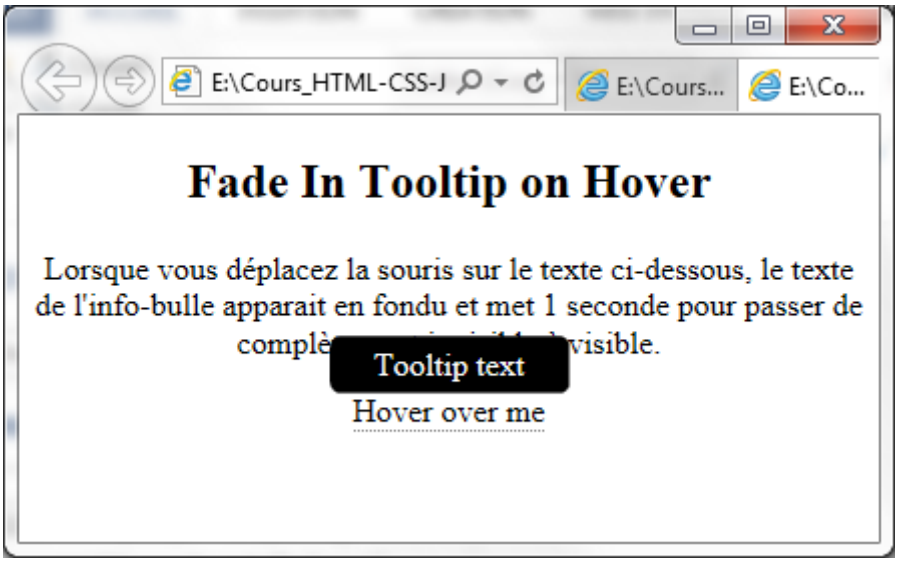
```
<!DOCTYPE html>
<html>
```

```
<head><meta charset="utf-8">
<style>
.tooltip {
  position: relative;
  display: inline-block;
  border-bottom: 1px dotted black;}
.tooltip .tooltiptext {
  visibility: hidden;
  width: 120px;
  background-color: black;
  color: #fff;
  text-align: center;
  border-radius: 6px;
  padding: 5px 0;
  position: absolute;
  z-index: 1;
  bottom: 100%;
  left: 50%;
  margin-left: -60px;
   /* Fondu dans l'infobulle - prend 5 seconde pour passer de 0% à
  100% d'opacité :  */
   opacity: 0;
  transition: opacity 5s;}
.tooltip:hover .tooltiptext {
  visibility: visible;
  opacity: 1;}
</style>
</head>
<body style="text-align:center;">
<h2>Fade In Tooltip on Hover</h2>
```

Lorsque vous déplacez la souris sur le texte ci-dessous, le texte de l'info-bulle apparait en fondu et met 1 seconde pour passer de complètement invisible à visible.

Survolez-moi
  Tooltip text

</body></html>



2.17. CSS Flexbox

Module de mise en forme CSS Flexbox

Avant le module Flexbox, il existait quatre modes de présentation:

Block, pour les sections d'une page Web
Inline, pour le texte
Table, pour les données de table à deux dimensions
Position, pour la position explicite d'un élément

Le module de présentation de boîte flexible facilite la conception d'une structure de présentation réactive flexible sans utiliser de flottement ou de positionnement.

Eléments Flexbox

Pour commencer à utiliser le modèle Flexbox, vous devez d'abord définir un conteneur Flex.

L'élément ci-dessus représente un conteneur flexible (la zone bleue) avec trois éléments flexibles.

<!DOCTYPE html>

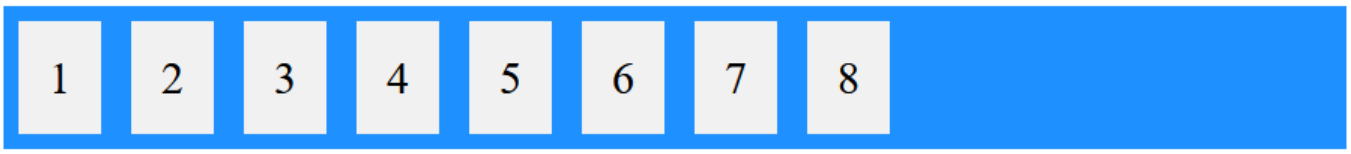
<html>

<head>

<meta charset="utf-8">

<style>

```
.flex-container {
  display: flex;
  background-color: DodgerBlue;
}
.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
</style>
</head>
<body>
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
</div>
<p>Une disposition flexible doit avoir un élément parent avec la propriété <em> display </em> définie sur <em> flex </em>.</p>
<p>Les éléments enfants directs du conteneur flexible deviennent automatiquement des éléments flexibles.</p>
</body>
</html>
```



Une disposition flexible doit avoir un élément parent avec la propriété *display* définie sur *flex* .

Les éléments enfants directs du conteneur flexible deviennent automatiquement des éléments flexibles.

Élément parent (conteneur)

Le conteneur flex devient flexible en définissant la propriété display sur flex:

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

La propriété flex-direction

La propriété flex-direction définit la direction dans laquelle le conteneur veut empiler les éléments flex.

La valeur de column empile les éléments flex verticalement (de haut en bas):

```
.flex-container {
  display: flex;
  flex-direction: column;
}
```

La valeur column-reverse inverse de colonne empile les éléments flex verticalement (mais de bas en haut):

```
.flex-container {
  display: flex;
  flex-direction: column-reverse;
}
```

La valeur de row empile les éléments flex horizontalement (de gauche à droite):

```
.flex-container {
  display: flex;
  flex-direction: row;
}
```

La valeur row-reverse empile les éléments flex horizontalement (mais de droite à gauche):

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

La propriété flex-wrap

La propriété flex-wrap spécifie si les éléments flex doivent être habillés ou non.

Les exemples ci-dessous ont 8 éléments flexibles, pour mieux démontrer la propriété flex-wrap.

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

Exemple :

```
<!DOCTYPE html>

<html>
<head>

<meta charset="utf-8">

<style>

.flex-container {

  display: flex;

  flex-wrap: wrap;

  background-color: DodgerBlue;

}

.flex-container > div {

  background-color: #f1f1f1;

  width: 100px;

  margin: 10px;

  text-align: center;

  line-height: 75px;

  font-size: 30px;

}

</style>

</head>

<body>

<h1>Flex-wrap Property</h1>

<p>Le "flex-wrap: wrap;" spécifie que les éléments flex vont s’enveloppés si nécessaire:</p>
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
</div>

<p>Essayez de redimensionner la fenêtre du navigateur.</p>

</body>

</html>
```

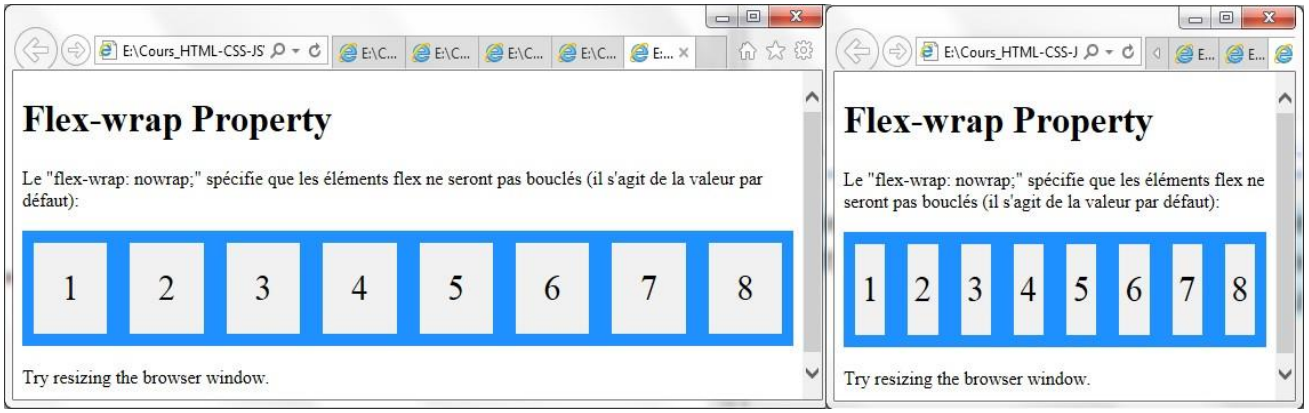
La valeur nowrap indique que les éléments flex ne seront pas enveloppés (il s'agit de la valeur par défaut):

```
.flex-container {
  display: flex;
  flex-wrap: nowrap;
}
```

Exemple :

```
<!DOCTYPE html>

<html>
<head><meta charset="utf-8">

<style>

.flex-container {
```

```
  display: flex;
  flex-wrap: nowrap;
  background-color: DodgerBlue;
}
.flex-container>div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>
<h1>Flex-wrap Property</h1>
<p>Le "flex-wrap: nowrap;" spécifie que les éléments flex ne seront pas enveloppés (il s'agit de la valeur par défaut):</p>
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
</div>
<p> Essayez de redimensionner la fenêtre du navigateur.</p>
</body></html>
```



La valeur wrap-reverse indique que les éléments flexibles seront renvoyés à la ligne si nécessaire, dans l'ordre inverse:

```
.flex-container {
  display: flex;
  flex-wrap: wrap-reverse;
}
```

La propriété flex-flow

La propriété flex-flow est une propriété abrégée permettant de définir les propriétés flex-direction et flex-wrap.

```
.flex-container {
  display: flex;
  flex-flow: row wrap;
}
```

La propriété justify-content

La propriété justify-content est utilisée pour aligner les éléments flex:

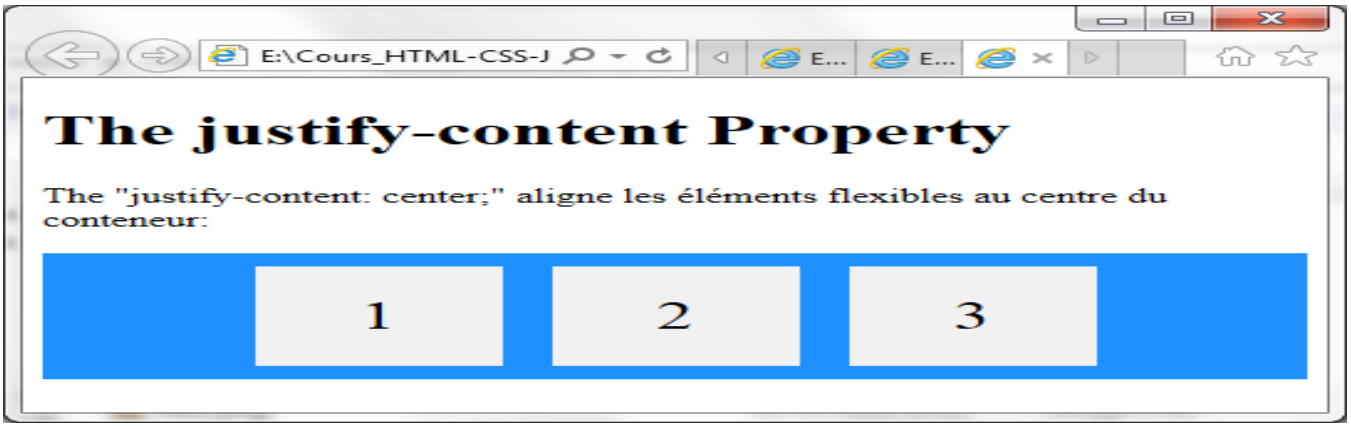
La valeur center : aligne les éléments flex au centre du conteneur:

```
.flex-container {
  display: flex;
  justify-content: center;
}
```

Exemple

```
<html>

<head><meta charset="utf-8">

<style>

.flex-container {

  display: flex;

  justify-content: center;
  background-color: DodgerBlue;

}
```

```
.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>
<h1>The justify-content Property</h1>
<p>The "justify-content: center;" aligne les éléments flexibles au centre du conteneur:</p>
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
</body>
</html>
```



La valeur flex-start aligne les éléments flex au début du conteneur (par défaut):

```
.flex-container {
  display: flex;
  justify-content: flex-start;
}
```

La valeur flex-end aligne les éléments flex à la fin du conteneur:

```
.flex-container {
  display: flex;
  justify-content: flex-end;
}
```

La valeur space-around affiche les éléments flexibles avec un espace avant, entre et après les lignes:

```
.flex-container {
  display: flex;
  justify-content: space-around;
}
```

La valeur space-between affiche les éléments flex avec un espace entre les lignes:

```
.flex-container {
  display: flex;
  justify-content: space-between;
}
```

La propriété align-items

La propriété align-items est utilisée pour aligner les éléments flex verticalement.
La valeur center aligne les éléments flexibles au milieu du conteneur:

Dans ces exemples, nous utilisons un conteneur de 200 pixels de hauteur afin de mieux démontrer la propriété align-items.

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

La valeur flex-start aligne les éléments flex en haut du conteneur:

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

La valeur flex-end aligne les éléments flex au bas du conteneur:

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

La valeur stretch étend les éléments flex pour remplir le conteneur (par défaut):

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

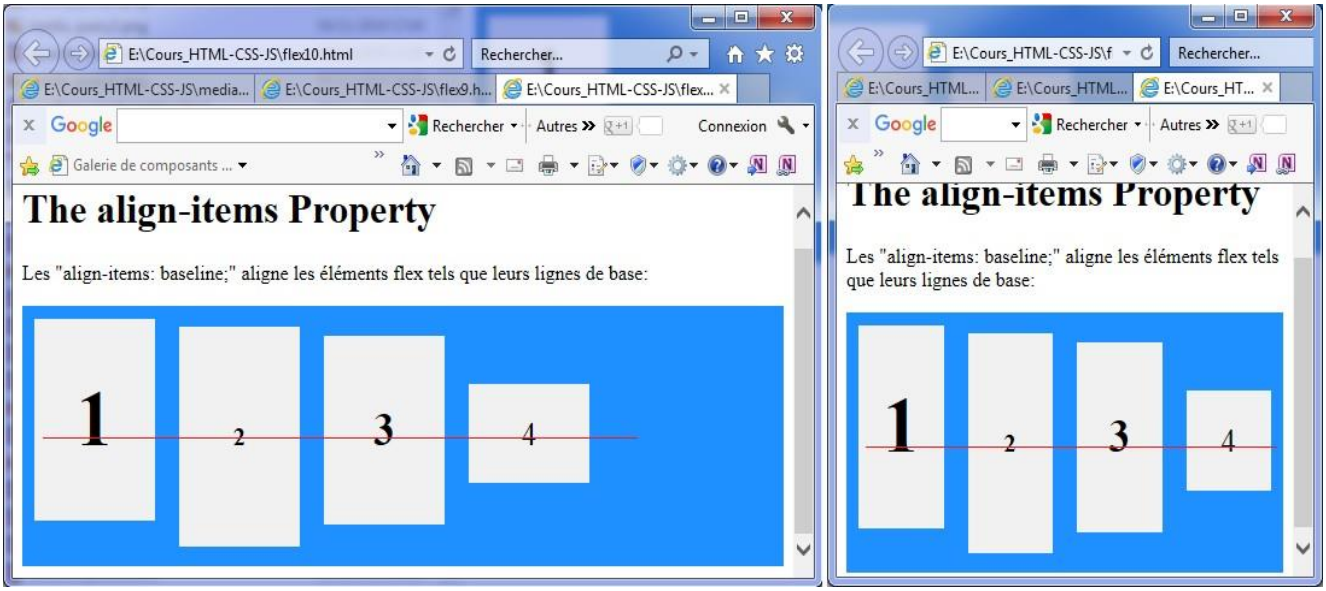
La valeur de baseline aligne les éléments flexibles, tels que leurs lignes de base.

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

Remarque: l'exemple utilise une taille de police différente pour démontrer que les éléments sont alignés sur la ligne de base du texte (baseline):

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<style>

.flex-container {

  display: flex;

  height: 200px;

  align-items: baseline;

  background-color: DodgerBlue;

}


.flex-container > div {

  background-color: #f1f1f1;

  width: 100px;

  margin: 10px;

  text-align: center;

  line-height: 75px;

  font-size: 30px;

}
```

```
</style>
</head>
<body>
<h1>The align-items Property</h1>
<p>Les "align-items: baseline;" aligne les éléments flex tels que leurs lignes de base:</p>
<div class="flex-container">
  <div><h1>1</h1></div>
  <div><h6>2</h6></div>
  <div><h3>3</h3></div>
  <div><small>4</small></div>
</div>
</body></html>
```



La propriété align-content

La propriété align-content est utilisée pour aligner les lignes flexibles.

La valeur space-between affiche les lignes de flexion avec un espace égal entre elles:

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: space-between;
}
```