Principes de base

Dans ce chapitre, nous allons aborder les différentes propriétés de base CSS qui permettent de mettre en forme les différents éléments HTML d'une page web.

1. Mise en forme

CSS offre une multitude de propriétés permettant de rendre nos pages élégantes et ergonomiques. Bien évidemment, le but n'est pas de tout connaître par cœur, mais au moins de connaître les propriétés les plus utiles et les plus courantes.

a. Caractères et polices

La propriété qui permet de spécifier une police de caractères donnée à utiliser est la propriété `font-family`.

`font-family` a pour valeurs possibles des noms de polices de caractères séparés par des virgules. C'est une propriété héritée, c'est-à-dire qu'elle se transmet à l'intérieur des balises imbriquées.

Voici un exemple d'utilisation de la propriété `font-family` pour un élément de type paragraphe `p` :

```
p{
      font-family: Arial, Verdana, sans-serif;
}
p{
      font-family: Times New Roman;
}
```

Il faut que les polices mentionnées dans l'exemple soient au préalable installées sur le poste. Mais avec CSS3, il est possible de télécharger des polices en même temps qu'elles seront affichées.

Il est également possible d'agir sur la taille des caractères en utilisant la propriété `font-size`.

Les valeurs possibles pour la propriété `font-size` sont des tailles exprimées en px, en %, en em et en ex.

Les px (pixels) sont des unités dont le rendu dépend de la résolution du périphérique d'affichage.

Les % permettent d'avoir une valeur proportionnelle à la taille de la fenêtre du navigateur ou de l'élément parent.

Les em sont des unités relatives. Elles sont relatives à la taille de police de l'élément. Par exemple, 1 em équivaut à 100 % de cette taille.

Les ex sont des unités relatives à la hauteur de la minuscule de l'élément.

Voici un exemple d'utilisation de la propriété `font-size` pour des éléments de type `p` et de type `h1` :

```
 p{
      font-size: 15px;
}
h1{
      font-size: 110%;
```

```
}
```

C'est une propriété héritée.

Voici quelques autres propriétés utiles pour la mise en forme des textes :

- `font-weight` : permet de préciser l'épaisseur de l'écriture. La valeur `bold` permet de mettre le texte en gras.
- `font-style` : permet de mettre le texte en italique avec la valeur `italic`.
- `text-decoration` : permet de souligner, surligner ou barrer le texte avec les valeurs `underline`, `overline` et `line-through`.
- `text-transform` : permet de mettre un texte en lettres majuscules ou minuscules avec les valeurs `uppercase` et `lowercase`.
- `text-justify` : permet de justifier du texte.

Un exemple pour ces dernières propriétés :

```
 a:hover{
      text-decoration: none;
      font-weight: bold;
}
p{
      text-transform: uppercase;
      text-decoration: underline;
      font-style: italic;
}
```

b. Couleurs et images

La propriété qui permet de spécifier la couleur d'un texte est la propriété `color`.

Les valeurs qui permettent de désigner une couleur sont :

- Des mots-clés tels que `red` pour rouge, `blue` pour la couleur bleue.
- Un code numérique hexadécimal, par exemple `#FFFFFF` pour la couleur blanche.
- La fonction `rgb`, par exemple `rgb(156, 25, 26)`.

```
 body{
      color: #0000ff;
}
h1{
      color: orange;
}
```

C'est une propriété héritée.

La propriété qui permet de spécifier la couleur de fond du texte, d'un paragraphe ou d'un bloc donné est la propriété `background-color`.

Voici un exemple d'utilisation de la propriété `background-color` :

```
body {
      background-color: orange;
}

h2{
      background-color: #00ff00;
}
```

Cette propriété n'est pas héritée.

Pour définir une image d'arrière-plan, il faut utiliser la propriété `background-image` à laquelle il faut ajouter la valeur suivante `url` (le nom de l'image avec son chemin relatif ou absolu).

```
body{
      background-image: url(images/image.png);
}
```

Voici deux autres propriétés intéressantes en rapport avec les images :

- `background-position`, afin de préciser la position horizontale et verticale de l'image.
- `background-attachment`, pour définir si l'image doit suivre le contenu lorsque l'utilisateur le fait défiler, ou si elle doit rester fixe.

```
body{
    background-position: center bottom;
    background-attachment:fixed;
}
```

c. Bordures

Les propriétés de bordure concernent les blocs de texte et les éléments qui possèdent des dimensions, comme les images.

Il est possible de spécifier le style de bordure et plus précisément le type de trait de contour avec la propriété `border-style`.

Voici quelques-unes des valeurs possibles pour cette propriété :

- `none` ou `hidden` : pas de bordure.
- `solid` : trait plein.
- `dotted` : pointillés.
- `double` : trait plein double.

Il est possible d'affiner le style de bordure pour chaque côté avec les propriétés suivantes :

- `border-top-style` : pour le style de la bordure du haut.
- `border-bottom-style` : pour le style de la bordure du bas.
- `border-right-style` : pour le style de la bordure de droite.
- `border-left-style` : pour le style de la bordure de gauche.

Voici un exemple d'utilisation de `border-style` :

```
p {
	border-style: solid;
}
```

L'épaisseur du trait de contour peut être fixée avec la propriété `border-width`.

```
p {
	border-style: solid;
	border-width: 5px;
}
```

L'épaisseur des 4 traits de contour est de 5px.

Sur le même principe, on peut spécifier des épaisseurs différentes pour chacune des 4 bordures avec `border-top-width`, `border-bottom-width`, `border-left-width` et `border-right-width`.

Dans l'exemple ci-dessous, les traits des contours n'ont pas tous la même épaisseur. Dans l'ordre, le trait du haut a une épaisseur de 2px, le trait de droite 3px, le trait du bas 3px et le trait de gauche 2px.

```
p {
	border-style: solid;
	border-width: 2px 3px 3px 2px;
}
```

d. Tableaux

Initialement, les colonnes d'un tableau ne sont pas fixes et varient en fonction du contenu. Pour qu'elles soient fixes, il faut le spécifier à l'aide de la propriété `table-layout` qui doit prendre la valeur `fixed`. Il s'agit d'une propriété héritée.

Voici un exemple d'utilisation de la propriété `table-layout` :

```
table{
	table-layout: fixed;
}
```

Il est possible de régler l'espace qu'il y a entre deux cellules qui sont côte à côte dans un tableau grâce à la propriété `border-spacing`. Si l'on spécifie pour cette propriété une seule valeur, celle-ci représente

l'espacement pour toutes les bordures. Par contre, si l'on spécifie deux valeurs, il s'agit de l'espacement horizontal puis vertical.

```
table{
      border-spacing: 1px 4px;
}
```

2. Positionnement

Tout l'aspect visuel d'une application web repose sur le positionnement des différents éléments dans la page. Il est donc primordial de maîtriser les différentes possibilités offertes par CSS en la matière.

a. Marges

Pour modifier les marges extérieures et intérieures pour un élément de type bloc, il faut respectivement utiliser les propriétés CSS `margin` et `padding`.

Les valeurs de ces marges s'expriment en px, en em ou en % si l'on souhaite avoir une taille proportionnelle.

La syntaxe pour définir une marge extérieure est la suivante :

```
margin: length;
```

La syntaxe pour définir une marge intérieure est la suivante :

```
padding: length;
```

Considérons l'exemple ci-dessous dans lequel un élément de type `div` contient un élément de type `p` qui comporte une marge extérieure de 20px et une marge intérieure de 50px :

```
div {
   margin:0;
   padding:2px;
   background:orange;
   width : 150px ;
 }

p {
   margin:20px;
   padding:50px;
   background-color:yellow;
 }
```

Voici le résultat que l'on obtient :

Il est possible de ne pas fixer les mêmes tailles de marges extérieures en haut, à droite, en bas et à gauche. Il y a deux possibilités pour le faire :

- Utiliser la propriété `margin` en spécifiant les tailles des marges dans l'ordre haut, droite, bas et gauche, les unes à la suite des autres avec un espace entre chaque taille.
- Utiliser les propriétés `margin-top`, `margin-right`, `margin-bottom` et `margin-left`.

Ainsi, dans cet exemple, les deux règles sont strictement équivalentes :

```
p{
      margin:4px 5px 4em 0;
}
p{
      margin-top:4px;
      margin-right:5px;
      margin-bottom:4em;
      margin-left:0;
}
```

Pour les marges intérieures, c'est le même principe.

```
p{
      padding:4px 5px 4em 0;
}
p{
      padding-top:4px;
      padding-right:5px;
      padding-bottom:4em;
      padding-left:0;
}
```

b. Positionnement des éléments

Dans une page web, il est possible de positionner les différents blocs les uns par rapport aux autres ou à un endroit précis.

Par défaut, les éléments de type « `bloc` » se placent les uns en dessous des autres. Ce sont les éléments de type `div`, `section`, `nav`, `header`, `main`, `form`, `p`, `h1`, `h2`…

Par défaut, les éléments de type « `en ligne` » se placent les uns à la suite des autres. Ce sont notamment les éléments `img`, `span`, `input`, `a`…

Ce qu'il faut comprendre, c'est que grâce à CSS et à ses propriétés, il devient possible de changer le mode d'affichage d'un élément.

Il faut noter que la plupart des propriétés destinées à positionner les éléments sont destinées à des éléments de type bloc ou disposant de la propriété `display: block`, c'est notamment le cas pour les propriétés `position`, `top`, `right`, `bottom` et `left`.

c. Propriété display

Avec la propriété `display`, il est possible de modifier le « flux » normal d'un élément et donc d'agir sur la façon dont il s'affiche.

Elle a pour syntaxe :

```
display : value ;
```

Voici quelques-unes des valeurs possibles et utilisées pour la propriété `display` :

- `inline` : l'élément est affiché comme un élément de type `inline`, c'est-à-dire en ligne.
- `block` : l'élément est affiché comme un élément de type `bloc` tel que `div`. L'élément suivant sera affiché à la ligne.
- `inline-block` : l'élément est affiché à la fois comme un élément de type `bloc` et `inline`. L'élément suivant sera affiché sur la même ligne.
- `inline-table` : l'élément est affiché comme un élément de type `table` mais de type `en ligne`, pas de type `bloc`.
- `flex` : l'élément est affiché comme un élément de type `flex-box`.
- `inline-flex` : l'élément est affiché comme un élément de type `flex`, mais `en ligne`.
- `table` : cette valeur permet d'afficher l'élément comme un élément de type `table`.
- `none` : spécifie que l'élément ne sera pas affiché.

Il peut être intéressant d'utiliser la propriété `display` pour changer le type d'un élément. Par exemple, il est parfois utile de transformer un élément de type `block` en un élément de type `inline`, cela dans le but d'avoir accès aux différentes propriétés des éléments de type `inline`.

Si, par exemple, on souhaite que deux blocs de type `div` ne s'affichent pas comme c'est nativement le cas l'un en dessous de l'autre, mais s'affichent l'un à la suite de l'autre, il est possible de le faire en ajoutant la propriété `display : inline` dans le code CSS :

```
<!DOCTYPE html>
<html>
  <head>
<style>
div {
 margin:0;
 padding:0px;
 background:orange;
 width: 200px;
 display: inline;
```

```
 border-color : black;
 border-style: solid;
 }
</style>
 </head>

  <body>

    <div >
              DIV1
   </div>
    <div >
              DIV2
   </div>

 </body>
</html>
```

En effet, si chacune des deux `div` a la propriété `display:inline`, elles seront affichées l'une à la suite de l'autre en ligne. Faites le test et mettez en commentaire dans le CSS la ligne `display: inline`. Par défaut, les div s'affichent bien l'une sous l'autre.

La différence entre les propriétés `display : none` et `visibility : hidden`, c'est que `display : none` n'affiche pas du tout l'élément alors que `visibility: hidden` masque seulement l'élément et parfois, cela laisse des espaces vides dans le document.

d. Propriétés top, right, bottom, left

Il est possible de positionner un élément en spécifiant un certain décalage par rapport au haut, à la droite, au bas ou à la gauche de l'élément parent.

Le décalage peut s'exprimer entre autres en px, en em ou en %.

Ces propriétés sont surtout utilisées dans le cas d'un positionnement en position absolue. Le positionnement absolu a pour effet de retirer un élément du flux, sa position est alors déterminée par référence à son conteneur.

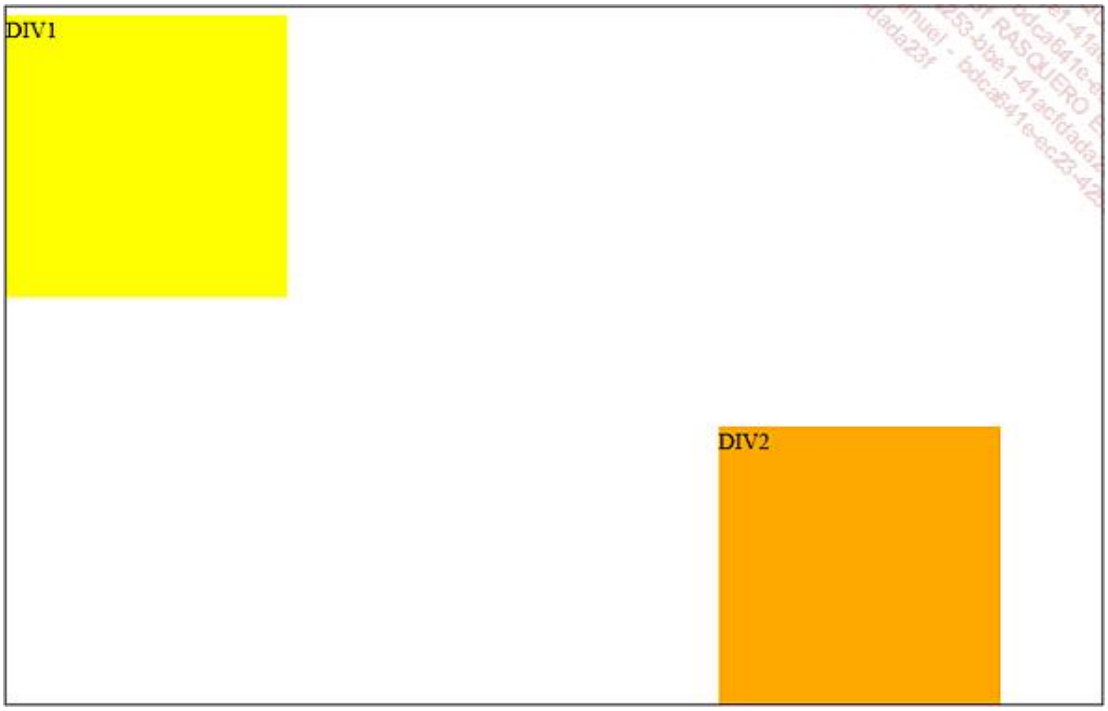
Cela est possible avec les propriétés suivantes :

- `top` : pour spécifier le décalage avec le haut.
- `right` : pour spécifier le décalage avec la droite.
- `bottom` : pour spécifier le décalage avec le bas.
- `left` : pour spécifier le décalage avec la gauche.

Voici un exemple pour `top`, `right`, `bottom` et `left` :

```
<!DOCTYPE html>
<html>
  <head>
<style>

div {
      height: 200px;
      width: 200px;

}

div#div1 {
      position: absolute;
      z-index: 2;
      background-color: yellow;
      top:10px;
      left:0;
}
div#div2 {
      position: absolute;
      z-index:1;
      background-color: orange;
      right: 10%;
      bottom: 0;
}


</style>
 </head>

  <body>

   <div id="div1">DIV1</div>
   <div id="div2">DIV2</div>

 </body>
</html>
```

Dans cet exemple, la div numéro un se place à `10px` du haut du document et est collée à gauche. La div numéro deux se place tout en bas du document et à `10%` du côté droit.



e. Propriété position

La propriété `position` permet, comme son nom l'indique, d'agir sur le positionnement direct d'un élément de type `block`.

Elle a pour syntaxe :

```
position: value;
```

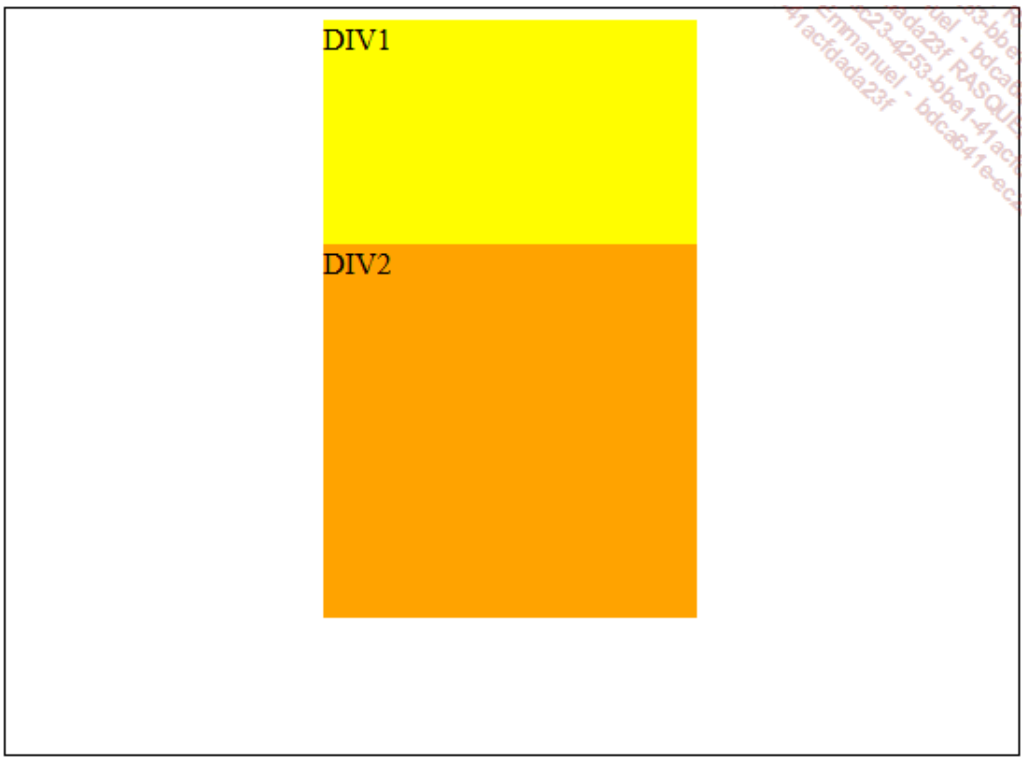
Différentes valeurs sont possibles pour cette propriété :

- `relative` : permet de placer un élément par rapport à un élément référent.
- `fixed` : l'élément est fixé, il ne bouge pas en cas de défilement.
- `absolute` : cette valeur sort l'élément totalement du flux. Il n'y a plus d'ordre d'affichage. On peut agir par la suite sur son placement avec les propriétés `top`, `right`, `bottom` et `left`.

Voici un exemple avec deux div : `div1` est en position relative, soit dans le flux, `div2` est sortie du flux normal. Sa position a pour valeur `absolute`. En indiquant `top:120px` et `left:0px`, cela a pour effet de positionner `div2` à `120px` du haut de `div1` et à `0px` du côté gauche de `div1`.

```
<!DOCTYPE html>
<html>
  <head>
<style>

div {
      height: 200px;
      width: 200px;
      left:200px;
```

```
}

div#div1 {
      position: relative;
      background-color: yellow;
}
div#div2 {
      position: absolute;
      background-color: orange;
       top: 120px;
    left: 0;
}


</style>
 </head>

  <body>

   <div id="div1">DIV1  <div id="div2">DIV2</div>
   </div>

 </body>
</html>
```



f. Propriété float

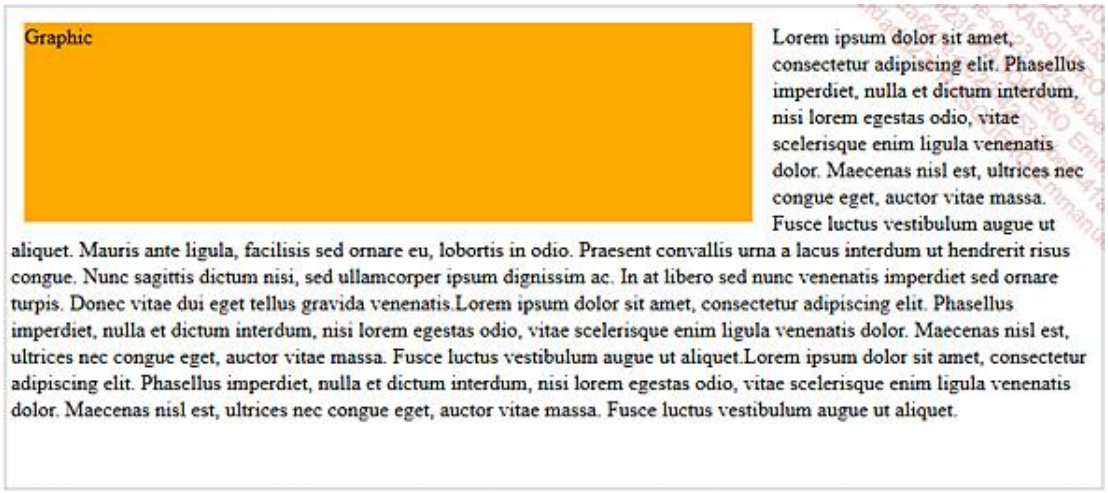
Il est possible de déclarer un élément comme étant flottant à gauche ou à droite. Cela a pour effet de placer le bloc le plus à gauche ou le plus à droite. Cette propriété a été créée par rapport au besoin d'habiller un graphique ou une image par du texte, comme on peut voir couramment dans les journaux ou la presse.

Attention, l'élément sera placé hors du flux.

Pour placer l'élément en mode flottant à gauche, il faut utiliser `float: left`, et pour placer l'élément en mode flottant à droite, `float: right`.

Dans cet exemple, la div destinée à contenir un graphique ou une image est placée en position flottante à droite et le reste du paragraphe de texte va l'entourer :

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    float: left;
    margin: 0 15px 10px 10px;
    width: 550px;
    height: 150px;
    background-color: orange;
}

</style>
</head>
<body>


<p><div>Graphic</div>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus
imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae
scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec
congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet.
Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent
convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis
dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc
venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida
venenatis.Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio,
vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices
nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut
aliquet.Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus
imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae
scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec
congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut
aliquet.</p>

</body>
</html>
```



g. Propriété z-index

Il peut arriver que différents blocs s'empilent ou se superposent. Pour pallier ce petit inconvénient, il est possible d'agir sur l'ordre d'empilement d'un bloc grâce à la propriété `z-index`.

Plus la valeur de `z-index` est élevée, plus le bloc se trouvera en haut de la pile de superposition.

Voici un exemple d'utilisation de cette propriété : les deux div sont empilées. Au final, on ne voit que la première div, la div jaune :

```
<!DOCTYPE html>
<html>
  <head>
<style>

div {
      height: 200px;
      width: 200px;

}

div#div1 {
      position: absolute;
      z-index: 2;
      background-color: yellow;
}
div#div2 {
      position: absolute;
      z-index:1;
      background-color: transparent;
}


</style>
 </head>

  <body>
```

```
   <div id="div1">DIV1</div>
   <div id="div2">DIV2</div>

 </body>
</html>
```

Ici, `div1` a un z-index plus élevé que celui de la div orange, et c'est pour cela qu'elle est affichée en premier dans cette superposition.

Faites le test : inversez les valeurs des z-index pour les div, et c'est la div orange qui sera affichée en premier.

h. Centrage horizontal

Il est fort utile de savoir comment centrer des images, des textes ou des blocs. Les éléments pourront être centrés horizontalement.

Pour centrer horizontalement du texte ou des éléments en ligne, la propriété conseillée est `text-align`.

Ainsi, il faudra écrire :

```
element{
      text-align: center;
}
```

Pour centrer horizontalement des éléments de type `block` par rapport à l'élément qui les contient, il faut utiliser les propriétés `margin-left` et `margin-right` en leur attribuant la valeur `auto`.

```
div{
      margin-left: auto;
      margin-right: auto;
}
```

i. Centrage vertical

Il n'existe aucune balise unique destinée à centrer verticalement un élément. Pour y parvenir, il faut avoir recours à plusieurs propriétés existantes.

Pour centrer verticalement du texte ou des éléments en ligne, il faut déclarer une `line-height` de la même hauteur que l'élément.

Ainsi, il faudra écrire :

```
element{
      height: 200px;
      line-height: 200px;
}
```

Voici un exemple, le texte se retrouve centré verticalement par rapport à l'élément qui le contient ici (un élément

de type p) :

```
<!DOCTYPE html>
<html>
<head>
<style>

p{
      height: 200px;
      line-height:200px;
      background-color: orange;
}

</style>
</head>
<body>

<p>
Vertically centered text
</p>

</body>
</html>
```

Pour centrer verticalement des éléments de type `block`, il faut suivre les étapes ci-dessous.

Pour l'élément contenu, il faut utiliser la position `absolute` pour le sortir du flux.

Ensuite, il faut appliquer la propriété `top: 50%` pour le placer à 50 % du conteneur. À ce moment précis, la bordure du haut de cet élément est placée à 50 % de la bordure du haut du conteneur.

Pour centrer l'élément, il faut encore effectuer cette dernière étape : retrancher la moitié de la hauteur de l'élément à centrer avec la propriété `margin-top` (si l'élément a une hauteur de 200px, alors il faudra écrire `margin-top: -100px`).

Pour l'élément conteneur qui contient l'élément à centrer, il faut qu'il soit déclaré en position relative.

Ainsi, il faudra écrire :

```
container{
```

```
position: relative;
}
containedelement{
      position: absolute;
      top: 50%;
      height:200px;
      margin-top: -100px;
}
```

Voici un exemple :

```
<!DOCTYPE html>
<html>
<head>
<style>
div#div1{
      height:400px;
      width: 200px;
      background-color: orange;
      position: relative;

}
p{
      position: absolute;
      top: 50%;
      margin-top:-50px;
      background-color: red;
      height: 100px;

}
</style>
</head>
<body>
 <div id="div1">
   <p>
   Vertically centered text
   </p>
 </div>
</body>
</html>
```