SEANCE 3 : Interface, mise en page et structuration d'une application Web – PART 2 / pratique

1. OUTILS DE STRUCTURATIONS ET D'ORGANISATIONS DE LA MISE EN PAGE

1. Organisation des balises et mise en page

Pour composer une mise en page web. Nous allons utiliser 2 types de balises différentes :

- Les balises « blocks » :
Ces balises structurent le contenu dans des cadres que l'on positionne.
Chacun de ces blocks va accueillir les différents contenus (textes & images).
Chaque bloc utilise, 100% de la largeur écran disponible.

- Des balises « inline »
Généralement utilisées pour donner des effets de typographies. Les balises « inline » se placent les unes à la suite des autres.

NB : **L'organisation du code doit prendre en compte** :
- La sémantique, le sens des balises que l'on emlploie.

2. Le modèle des blocks

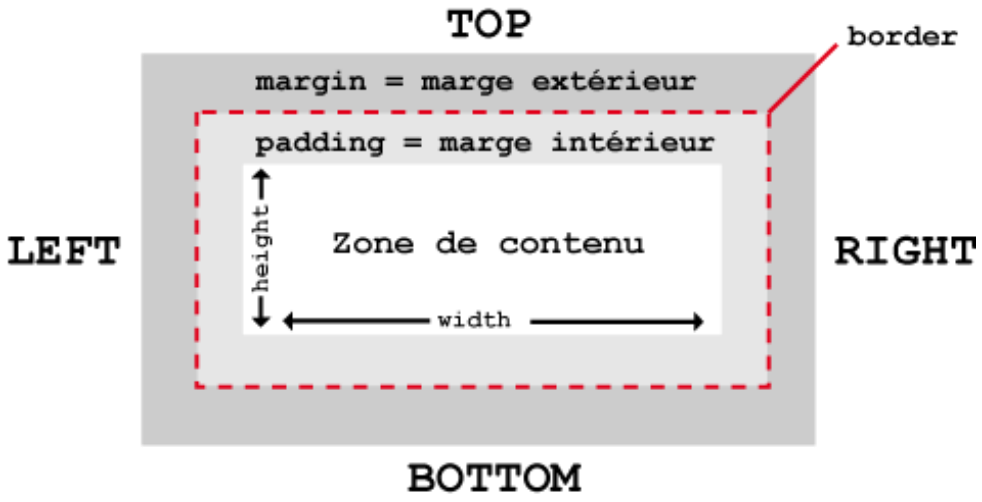
ATTENTION : Il est important de noter que ce modèle fonctionne pour l'ensemble des balises de types block, <p><h1> etc...
Dans nos exemples nous utiliserons une boite, , qui est de type block, neutre (non-sémantique), comme référence.

Le block est un élément de base du modèle de mise en forme visuel (visual formatting model). Il s'agit des spécifications concernant le rendu graphique effectué par les navigateurs.

Plus simplement lorsque le navigateur reçoit la page web à interpréter, il analyse la répartition des éléments « blocks » et des éléments « inline » et les positionne les uns par rapport aux autres en tenant compte de la taille écran de l'utilisateur.

Un block possède systématiquement une zone de contenu, et intègre les propriétés CSS suivantes :

- Des dimensions : « width » et « height » détermine la taille
- Marge intérieure : « padding ». Permet de chasser le contenu interne du bord du bloc
- Marge extérieure : « margin ». Crée une marge extérieure au block
- Des bordures : « border ». Peu utilisées, elles ont une vocation graphique.
- Un fond : « background », qui permet d'associer couleur et/ou image de fond.



Pour représenter graphiquement un block dans le navigateur, il convient au minimum d'y associer du contenu et/ou des propriétés CSS (dimensions, propriétés graphiques…).

Par défaut un block a la propriété graphique suivantes :

1. Transparent, sans fond
2. Peut avoir une marge intérieure et extérieure par défaut (margin et padding). Cela dépend des navigateurs. Exemple niveaux de titres <h1> et paragraphes <p> ont toujours une marge extérieure haute et basse).
3. Occupe 100% de la largeur de l'écran mais n'a pas de hauteur (la hauteur est également définie par le contenu placé à l'intérieur de la balise).
4. Sans bordures. Aligné en haut à gauche (100% largeur)

NB : Par défaut certains navigateurs attribuent des propriétés graphiques à certains blocks.
Par exemple un navigateur afficher un niveau de titre `<h1>` différemment d'un autre navigateur

3. Composition et principes d'imbrications

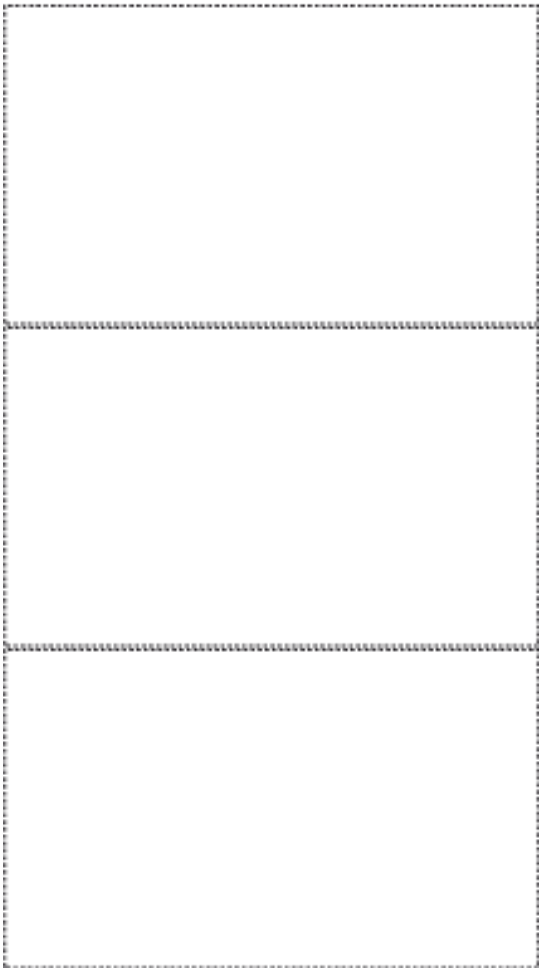
Pour créer une boite nous utilisons une balise ouvrante .

```
<div>
      Contenu à encadrer
</div>
```

Au final, ce seront les différentes propriétés CSS que nous associerons aux blocks qui permettront la création d'une mise en page élaborée et le positionnement des blocs.

```
<div>  contenu </div>            <!-- début et fin boite 1 -->
<div>  contenu </div>            <!-- début et fin boite 2 -->
<div>  contenu </div>            <!-- début et fin boite 3 -->
```

Boites superposées



Il convient donc d'opter rapidement pour une organisation visuelle du code.

```
<div>                                        <!-- début boite 1 -->
     <div>                                   <!-- début boite 2 -->
          <div>                              <!-- début boite 3 -->
               Contenu encadré par la dernière boite.
          </div>                             <!-- fin boite 3 -->
     </div>                                  <!-- fin boite 2 -->
</div>                                       <!-- fin boite 1 -->
```

Boites imbriquées

4. Unités de mesures

L'affichage du site dépend grandement de l'unité de mesure choisie par le développeur

Majoritairement il existe 2 unités de mesures pour développer une mise en page avec des blocs :

- Le pourcentage généralement noté « % ». Pourcentage de la taille écran. La taille de l'élément est définie en fonction de la taille écran de l'utilisateur. **Ex :** 50%

- Le pixel noté « px ». L'unité de mesure de l'écran. **Ex :** 500px

Les modèles de sites développés en pourcentage permettent d'utiliser l'intégralité de la largeur d'écran disponible. Cette unité de mesure est souvent utilisée pour faire « du design responsive » Peu importe la taille de l'écran, le contenu s'adaptera.

Le site web mobile des pages jaunes

Les 2 propriétés CSS qui permettent de définir les proportions d'un block s'intitulent « **width** » et « **height** ».

Il est important de prendre en considération que la hauteur, « height » est également définie par la hauteur du contenu placé à l'intérieur du bloc.

Ainsi, la plupart des blocks qui incluent le corps de texte n'ont pas de hauteur définie, puisque c'est le contenu texte, placé à l'intérieur, qui déterminera la hauteur.

```
<div style="width:250px;height:300px;">
     Contenu à encadrer
</div>
```

NB : Par défaut la largeur d'un block est toujours 100%.

NB : Il n'est pas possible de définir une hauteur en pourcentage. Excepté si le type de positionnement de l'élément est absolu « absolute ». Cf. suite du cours.

D'autres propriétés CSS permettent de définir le minimum et le maximum de ces dimensions :

- `min-width` : largeur minimale
- `min-height` : hauteur minimale
- `max-width` : largeur maximale

- `max-height` : hauteur maximale

Ces propriétés permettent d'adapter les dimensions d'un block en fonction de la taille écran disponible. C'est la raison pour laquelle on définira généralement une taille principale couplée à une taille minimum ou maximum.

```
<div style="width:100%;min-width:600px;">
     Contenu à encadrer
</div>
```

Dans cet exemple la largeur de la boite est de 100% de la taille écran. Toutefois si l'écran est inférieur à 600px, la largeur de la boite (100%), ne pourra pas être inférieure à 600px. Par conséquent des barres de scroll horizontales devraient apparaître dans le navigateur pour visualiser la totalité de la boite.

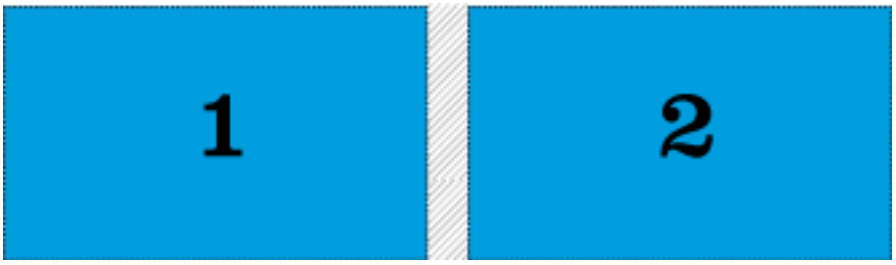
5. Marges extérieures (margin)

Les marges permettent de définir un écart entre les blocks et d'autres éléments adjacents dans la mise en page. Elles sont généralement utilisées pour chasser les blocs. Ce qui peut par exemple être le cas, lorsque l'on souhaite décoller le « footer » (pied de page du site) du bloc de texte principal. On pourrait, dans ce cas de figure, ajouter une marge basse au bloc de texte principal ce qui aurait pour effet de pousser systématiquement le « footer ».

Pour chaque bloc il est possible de définir indépendamment une marge haute, droite, basse ou gauche via les propriétés CSS « margin-top », « margin-right », « margin-bottom » et « margin-left ».

```
<div style="margin-top:5px;"> </div> <!-- Marge haute -->
<div style="margin-right:10px;"> </div> <!-- Marge droite -->
<div style="margin-bottom:15px;"> </div> <!-- Marge bas  -->
<div style="margin-left:20px;"> </div> <!-- Marge gauche -->
```

En fonction de leurs positionnements (droite, gauche…), la boite 1 chassera la boite 2 ou inversement



Bien entendu il est possible de coupler ces propriétés.

```
<div style="margin-top:5px; margin-bottom:15px;"> </div>
<!-- Marge haute et basse-->
```

Il existe une autre modalité d'écriture que l'on nomme « concaténée », qui permet d'associer les valeurs respectives de chacune des marges à une seule propriété CSS : « margin ». Ainsi nous allons

définir successivement la valeur des 4 marges que l'on va avoir, en suivant le sens des aiguilles d'une montre : haut, droite, bas, gauche

```
<div style="margin : 5px 10px 15px 20px;"> </div>
```

Dans cet exemple, la boite a comme propriété :
Une marge haute de 5px / une marge à droite de 10px / une marge en bas de 15px / une marge à gauche de 20px

Une autre syntaxe concaténée permet de définir les marges d'un block avec 2 valeurs, haut et bas, gauche et droite.

```
•     <div style="margin : 20px 5px;"> </div>
```

Dans cet exemple, la boite a comme propriété :
Une marge haute et basse de 20px / une marge à droite et à gauche de 5px.

NB : Il est possible d'utiliser des marges négatives en introduisant la valeur de la marge par un signe moins, «-».
Ex : margin-top:-50px ;

NB : Certains blocks ont par défaut des marges supérieures et inférieures. C'est le cas par exemple des niveaux de titres <h1>, et des paragraphes <p>. Il sera parfois nécessaire de supprimer ces marges, pour se faire il suffit de donner une valeur de 0 ou 0px à cette propriété.
Ex : margin:0px ;

6. Marges intérieures (padding)

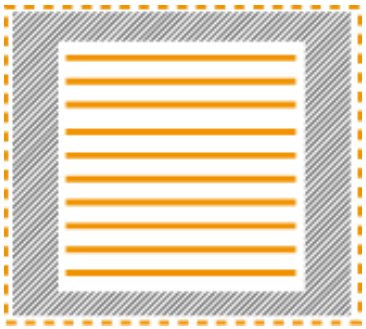
Il s'agit de la marge présente entre la zone de contenu et la bordure du bloc, on parle de marge intérieure. Elle est généralement utilisée pour décoller le texte du contour du bloc.

Cette propriété CSS s'intitule « padding », et obéit aux mêmes règles de compositions que « margin ».

```
<div style="padding-top:5px;"> </div>
<!-- Marge int. haute -->
```

```
<div style="padding:5px 10px;"> </div>
<!-- Marge int. Haute & basse 5px, droite et gauche 10px -->

<div style="padding:5px 10px 15px 20px;"> </div>
<!-- Marge int. haute droite bas gauche-->
```

Le padding permet de décoller une zone de texte du bord du block.



NB : Il n'est pas possible d'utiliser un « padding » négatif.

Attention, lorsque la taille de l'élément est fixée, le « padding » va augmenter la taille du block.

```
<div style="width:200px;height:150px;padding:15px"> </div>
```

Dans cet exemple, la largeur véritable de la boite est de 230px de large, et 180px de hauteur, et non plus 200px et 150px.

Une des solutions pour conserver la taille originale du bloc consiste à créer un conteneur avec ces dimensions et à placer le block avec le « padding » à l'intérieur.

```
<div style="width:200px;height:150px;">
     <div style="padding:15px"> </div>
</div>
```

Ainsi, nous conservons la taille de la boite qui est définie par le conteneur et nous pouvons employer un « padding ».

7. Bordures (border)

Il s'agit du contour du bloc, avant les marges extérieures, mais après le « padding ». Chaque bordure est composée :

- d'une épaisseur (valeur en pixel)
- d'un style (droit, pointillé, hachuré)
- d'une couleur

Comme pour les marges il est possible de définir successivement les côtés de la boite qui disposent d'une bordure.

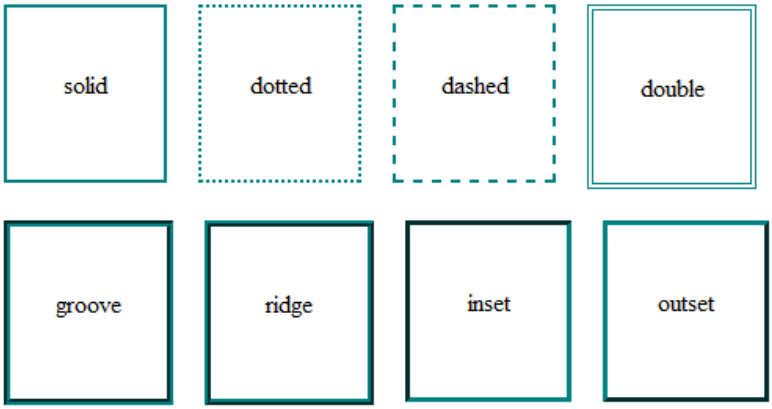
```
<div style="border-left:1px solid #ffcc00;"> </div>
<!-- Une bordure droite (trait continu) d'un pixel et de couleur
rouge -->
```

Il est possible de définir une bordure pour tous les côtés en utilisant la propriété CSS « border », « concaténée ».

```
<div style="border:2px dashed blue;"> </div>
<!-- Bordures X4 : 2 pixels hachurées bleues -->
```

NB : Il est important de prendre en considération que la bordure d'une boite s'ajoute à sa taille initiale.

Les styles de bordures sont nombreux. Voici les différents styles :



NB : De nombreux exemples de cette section utilisent des boites, . Mais il est indispensable de comprendre que le modèle des blocks fonctionne aussi bien avec une balise , qu'une balise <h1>, <p>, <footer>…

2. LES TYPES DE POSITIONNEMENTS

Il existe 2 types de positionnements :

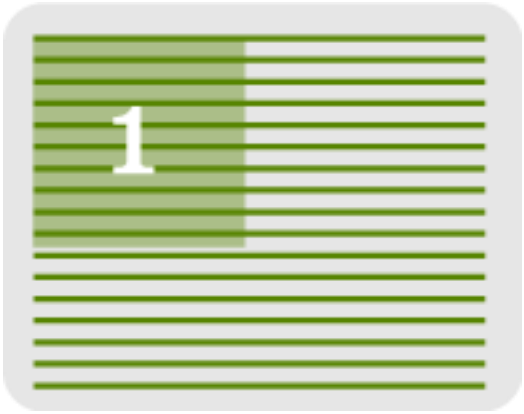
- **Statique :** La position d'un « block » par défaut.
Les « blocks » se superposent les uns par-dessus les autres ou les uns à côté des autres en obéissant aux règles définies par le flux de balises (l'ordre dans lequel les balises apparaissent).

- **Positionné :** L'élément sort du flux de balise et sera positionné avec des coordonnées.
On dit que cet élément est hors flux car il « survolera » les autres éléments de la page. Un peu comme si on crée un système de calque sous Photoshop. Dans ce cas de figure peu importe l'endroit où apparaît le code de ce block dans document HTML, il sera toujours positionné au même endroit.

Dans ce mode (positionné), nous distinguons 3 méthodes différentes : fixe, absolue, relative.

Elément statique. Les blocs se chassent les uns les autres. Ils sont dans le flux.

Elément positionné. Le bloc passe par-dessus la mise en page, il est hors flux.



MEMENTO : En mode statique, les blocks se superposent les uns par-dessus les autres et sont directement représentés en fonction de leur placement dans le flux de balises.

3. LE POSITIONNEMENT STATIQUE

1. Centrer des éléments en position statique

- **Centrer un bloc**

Pour centrer une boite ou un élément de type « block » il convient de coupler et de définir 2 propriétés CSS. Plus précisément d'associer les propriétés « margin- left » et « margin-right » à la valeur « auto ».

```
<div style="width:250px;height:300px;margin-left:auto;margin-
right:auto;">
     Contenu à encadrer
</div>
```

NB : Les boites qui n'ont pas de proportions mesurent par défaut 100% de la largeur de l'écran. **Il est donc indispensable de définir une largeur pour pouvoir observer l'effet de centrage.**

Mais rappelez-vous, il existe une autre modalité d'écriture que l'on nomme « concaténée », qui permet de regrouper les 2 propriétés « margin-left » et « margin-right » en une seule : « margin ».

```
<div style="width:250px;height:300px;margin:auto;">
      Contenu à encadrer
</div>
<!-- une valeur "auto" est associée au 4 bords. L'élément sera donc
centré -->
```

- **Centrer du contenu (text/image et autre…)**

On désigne par « contenu » l'ensemble des ressources textes / multimédia / vidéos / images etc. Dans la majorité des cas on peut utiliser directement la propriété CSS d'alignement « text-align » sur l'élément en question, ou sur l'élément parent (conteneur).

La propriété « text-align » peut prendre les valeurs « right », « left », « center », « justify ».

```
<p style="text-align:center">
     Contenu centré
</p>
<!-- ou -->

<div style="text-align:center">
     <p>Texte qui sera également centré</p>
</div>
```

- **Centrer une image <img>**

Une image est de type « inline ». C'est-à-dire qu'elle est considérée par le navigateur comme un élément placé à la suite d'autres éléments. Pour centrer un bloc image en CSS, `<img>` il convient de redéfinir son type avec la propriété CSS « display », puis d'associer une propriété CSS de centrage : « margin » de la manière suivante.

```
<img src="url" style="display:block;margin:auto;" />
```

NB : Il est également possible de centrer une image en la plaçant dans un conteneur ayant une propriété CSS de centrage de contenu : `text-align :center ;` Mais cela est généralement peu pratique et les cas d'applications sont restreints.

- **Centrer un bloc et gérer les marges hautes et basses**

En CSS, la propriété qui permet de centrer un élément et de gérer les marges est la même : `margin`.

Mais souvenez-vous, pour centrer un élément il suffit d'attribuer la valeur « auto » à la marge gauche et à la marge droite.

```
<div style="margin-left:auto;margin-right:auto;"> </div>
<!-- ou -->
<div style="margin:auto;"> </div>
<!-- plus simple… -->
```

En considérant ces principes d'écritures il serait donc possible de centrer et de marger le block de la manière suivante :

```
<div style="margin: 5px auto 15px auto;"> </div>
```

Ainsi, la boite dispose d'une marge haute de 5px, d'une marge basse de 15px et sera centrée dans la mise en page.

2. Créer des colonnes, coller des blocs

Il existe **2 méthodes** pour « coller », ou disposer plusieurs blocs les uns à la suite des autres… Chacune de ces méthodes permet d'arriver plus ou moins facilement au même résultat et présente des avantages et des inconvénients. A vous de choisir…

- **Méthode « float »**

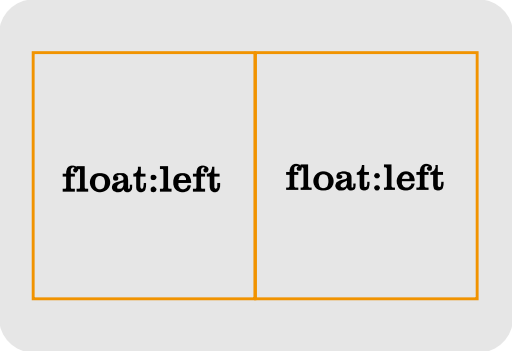
- La propriété CSS « float » doit être associée aux 2 éléments qui doivent être collés.
- Lorsqu'un élément est en « float » (flottant), il n'est plus intégré dans le flux de balises de la page, dans la structure. Il la survole en quelque sorte.
- L'élément qui suit une balise positionnée en float doit systématiquement avoir une propriété CSS «clear» avec la valeur « both », pour passer à la ligne : `clear:both;`

Dans le cas contraire (si vous n'utilisez pas la propriété « clear »), l'élément remontera au même niveau que le dernier élément « float » et sera positionné dessous, donc partiellement visible.

Ex :

```
<div style="width:250px;height:300px;float:left;"> collé </div>
<div style="width:250px;height:300px;float:left;"> collé </div>
```

Mise en page en colonne employant les propriétés CSS « float » et « clear »



ATTENTION : Lorsque 2 blocks sont collés dans un conteneur, mais qu'ils n'ont pas suffisamment de place pour le faire, ils se positionnent les uns en dessous des autres. Il convient donc de veiller à ce que le block parent, le conteneur de ces éléments, soit suffisamment large pour que les éléments puissent se coller.

- **« clear »**

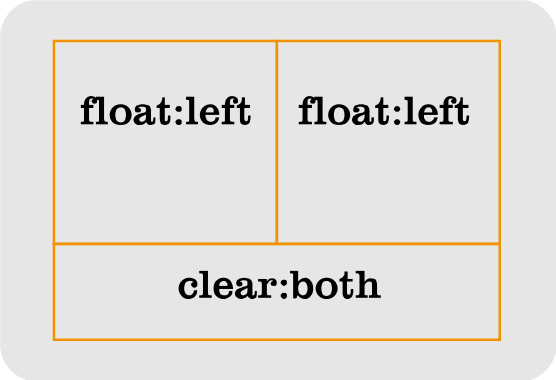
La propriété « clear » interdit à un « block » d'être adjacent à un « block » positionnée en « float ».

Ainsi « clear » place l'élément en dessous de tout bloc flottant qu'il soit à gauche ou à droite. C'est généralement la raison pour laquelle utiliser la propriété « float », implique systématiquement d'utiliser la propriété CSS « clear ».

Ex : (complet d'une d'utilisation correcte)

```
<div style="width:250px;height:300px;float:left;"> collé </div>
<div style="width:250px;height:300px;float:left;"> collé </div>
<div style="width:500px;height:100px;clear:both;"> à la ligne
</div>
```

Mise en page en colonne employant les propriétés CSS « float » et « clear »

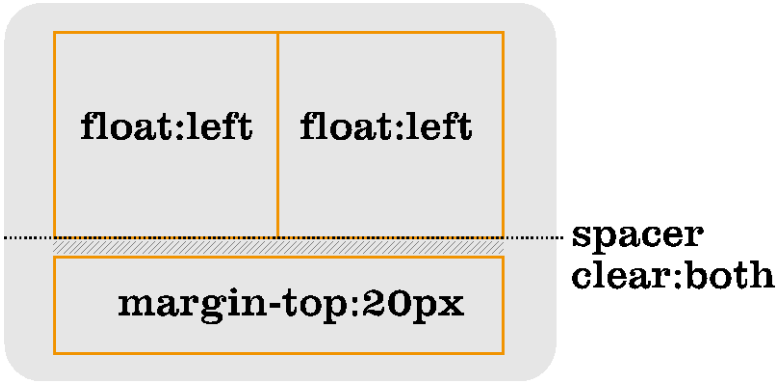


Par ailleurs, il n'est pas possible d'utiliser une marge haute ou une marge supérieure sur un élément ayant une propriété « clear ». Pour contrecarrer ce problème, nous utiliserons un « spacer ». Il s'agit d'une boite qui ne sera pas graphiquement représentée mais qui fera office de séparation entre le l'élément en « float » et l'élément qui suit.

Exemple de spacer :

```
<div style="width:250px;height:300px;float:left;"> collé </div>
<div style="width:250px;height:300px;float:left;"> collé </div>
<div style="clear:both;"></div>    <!-- séparateur ou spacer -->
<div style="width:500px;height:100px;margin-top:20px"> à la ligne
</div>
```

Mise en page en colonne avec « float » et un « spacer » en « clear »



- **Méthode « display:inline-block »**

La propriété CSS « display », permet de redéfinir le type d'élément (« block » ou « inline »).

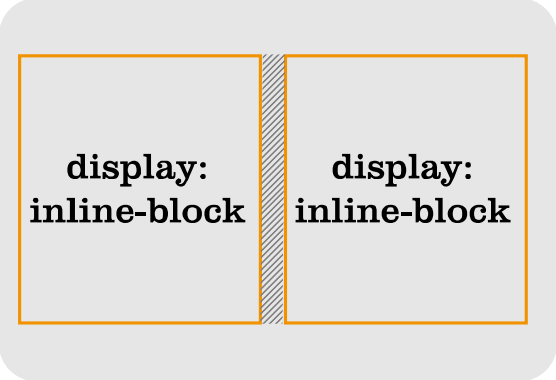
L'une de ses valeurs « inline-block » va permettre, comme son nom l'indique de conserver le type « block », mais de positionner les éléments en ligne. Cette méthode est souvent délaissée par les développeurs au profit de la méthode « float ».

Pour placer 2 blocks les uns à la suite des autres nous pouvons associer la propriété CSS « display » et sa valeur « inline-block »

```
<div style="width:250px;height:300px;display:inline-block;"> collé
</div>
```

```
<div style="width:250px;height:300px;display:inline-block;"> collé
</div>
```

2 éléments positionnés avec la propriété CSS « display »



Lorsque nous utilisons la propriété CSS « display », nous devons impérativement définir l'alignement vertical du contenu dans le block avec la propriété CSS « vertical-align ». Cette propriété peut prendre les valeurs « top », «middle », « bottom ».

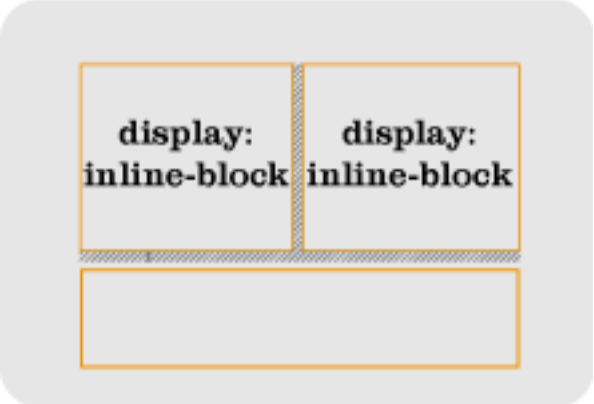
```
<div style="width:250px;height:300px;display:inline-block;vertical-
align:top;"> collé </div>

<div style="width:250px;height:300px;display:inline-block;vertical-
align:top;"> collé </div>

<!-- le contenu est aligné en haut pour les 2 blocks -->
```

Question: Mais quelle est la différence entre « display:inline », et «display:inline-block ; » ?
La valeur « inline » permet de redéfinir un élément comme étant de type « inline». Par exemple les balises qui permettent de mettre un texte en italique : texte en italique, sont de types « inline ». La particularité de ces balises est qu'elles n'ont pas de dimensions. C'est la taille du contenu placé à l'intérieur qui les détermine.

La valeur « inline-block » permet de conserver le type « block » mais de disposer les éléments les uns à la suite des autres en conservant le principe de dimensions.

Malheureusement le gros problème actuel c'est que l'emploi de « display:inline-block » crée des marges entre les blocks qu'il est parfois difficile à gérer.

« inline-block » crée une marge horizontale et verticale



En effet, à chaque fois que vous utilisez cette méthode un espace se crée entre les éléments adjacents et suivants. Cela provient du fait que les « blocks » définies comme étant « inline-block » doivent également l'être dans le code (écritures à la suite), ce qui n'est pas vraiment pratique… Il existe toutefois un « hack », une petite astuce, qui permet de contrecarrer ces effets. Il s'agit d'utiliser un commentaire pour annuler l'espace ou le saut de ligne. Le commentaire n'étant pas interpréter graphiquement par le navigateur, il va annuler le saut de ligne ou les espaces effectués entre les 2 balises. Comme si dans le code les écritures étaient à la suite.

```
<div style="display:inline-block;">contenu 1</div><!--
--><div style="display:inline-block;">contenu 2</div>
```

Même si cette méthode fonctionne, il est fréquent que la marge basse persiste, il va donc falloir employer une seconde astuce pour éradiquer le problème.

Il s'agit de placer les éléments dans un conteneur parent ayant une propriété CSS « font-size » définie à 0px ;

```
<div style="font-size:0px;">
    <div style="display:inline-block;">contenu 1</div><!--
    --><div style="display:inline-block;">contenu 2</div>
</div>
```

Cela implique a posteriori de redéfinir la taille des typographies « font-size » de chacun des blocks, car dans ce cas de figure, la taille est de 0px…

Exemple complet d'utilisation sans marge :

```
<div style="font-size:0px;">
    <div style="width:400px;height:300px;display:inline-
block;vertical-align:top;">contenu 1</div><!--
    --><div style="width:400px;height:300px;display:inline-
block;vertical-align:top;">contenu 1</div>
</div>
```

4. LE POSITIONNEMENT (positionné)

Il s'agit d'un mode de positionnement radicalement différent du positionnement statique.

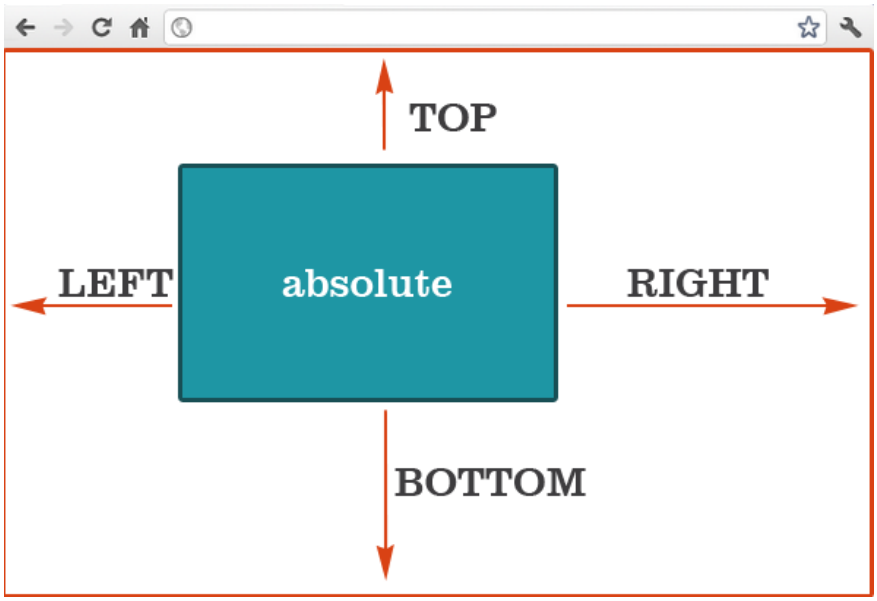
Le positionnement des éléments statiques s'opère en fonction des éléments adjacents. Il y a donc une contrainte de dépendance des éléments.

Lorsque nous utilisons la méthode « positionnée » l'élément sort du flux de balise et sera positionné avec des coordonnées. On dit que cet élément est hors flux car il « survolera » les autres éléments de la page. Un peu comme si on crée un système de calque sous Photoshop. Dans ce cas de figure, peu importe l'endroit où apparaît le code de ce block dans document HTML, il sera toujours positionné au même endroit.

C'est principalement le modèle de positionnement qui est adopté pour placer :
- Des menus déroulants, qui passent par-dessus les contenus de la page,
- Des pieds de pages (footer) fixes.
- Des éléments uniques sans avoir à créer de systèmes de conteneurs complexes.

Les éléments positionnés sont généralement nommés « calques ».

L'élément est positionné avec des coordonnées, tenant compte de la fenêtre du navigateur



Cette méthode fonctionne aussi bien sur des éléments de types « blocks » que sur des éléments de types « inline »

Il existe 3 modes distincts :

- Le positionnement absolu : il nous permet de placer un élément n'importe où dans la mise en page. Comme si on utilisait un système de grille. (en haut à gauche, en bas à droite, au centre, etc.).

- Le positionnement fixe : il nous permet de placer un élément par rapport à la fenêtre du navigateur et non par rapport à la mise en page. Ainsi, l'élément sera toujours visible même lorsque nous effectuons un « scroll » (un déplacement de la fenêtre), vers le bas. L'élément reste en place au même endroit.

- le positionnement relatif : permet de définir une position par rapport à un élément parent, ou par rapport à une position initiale

Pour définir l'un de ces 3 modes de positionnement nous utilisons la propriété CSS « `position` » à laquelle nous pouvons associer 3 valeurs : « `absolute` », « `fixed` », « relative ». A noter que lorsque l'on crée un block par défaut sa position est `static` : « `position:static` ».

- **Positionnement absolu (absolute)**

Nous allons définir l'emplacement de l'élément dans la mise en page en considérant les 4 côtés de la fenêtre du navigateur.

1. Dans un premier temps il convient de définir le type de positionnement avec la propriété CSS : `position:absolute;`

2. Ensuite, il convient de positionner l'élément par rapport à 1 bord vertical et 1 bord horizontal

Il existe 4 propriétés CSS pour positionner l'élément :
top : bord haut du document
right : bord droit du document
bottom : bord bas du document
left : bord gauche du document

Bien entendu il n'est pas possible définir un positionnement en fonction d'un bord gauche et d'un bord droit, il faudra choisir… Droite ou gauche…

```
<div style="position:absolute;top:0px;right:0px ;">contenu 1</div>
```

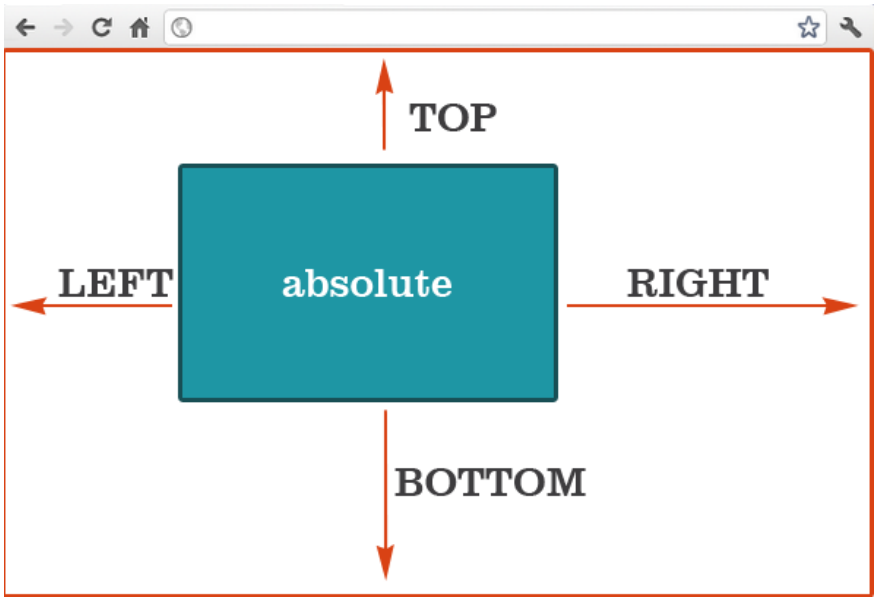
Dans cet exemple l'élément sera positionné en haut à droite de la mise en page.

Il est possible de définir une valeur en px ou en %

```
<div style="position:absolute;top:200px;left:50%;">contenu 1</div>
```

L'élément est positionné à 200px du bord et à 50% de la page. Attention cela ne va pas centrer l'élément puisque l'élément sera positionné par rapport à son bord gauche.

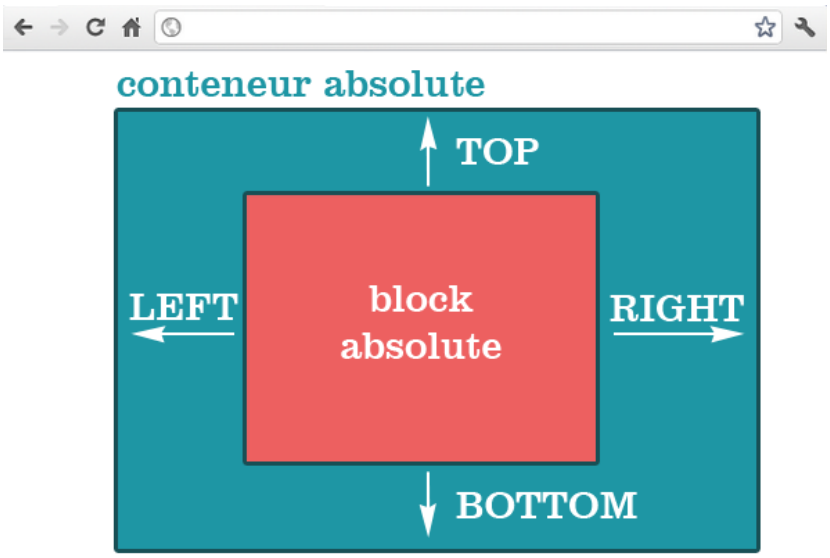
Un élément est positionné en fonction des bords de la mise en page



Mais cela ne s'arrête pas là !

Un block positionné en absolu, lui-même placé à l'intérieure d'un autre (positionné en absolu) sera positionné par rapport à ce dernier. Ce qui permet ainsi, de placer certains éléments sans avoir à utiliser des marges par exemple.

Block positionné en absolu à l'intérieur d'un autre block positionné en absolu



- **Positionnement fixe (fixed)**

Le block reste dans sa position fixe même si nous effectuons un « scroll » dans la page

Le principe de positionnement est exactement le même que pour le positionnement absolu.

A la différence que :

- Il faut définir la propriété CSS « `position:fixed;` »
- L'élément est positionné par rapport à la fenêtre du navigateur.

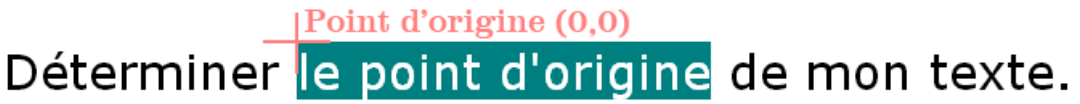
Attention : principe de positionnement dans un conteneur ne marche pas. C'est à diore que placé à l'intérieur d'un conteneur, une boite par exemple, l'élément ne sera pas positionné en fixe. C'est la raison pour laquelle nous plçons généralement ces éléments à la suite de la balise `<body>`. Couramment, ces positionnements fixes, nous permettent, par exemple d'afficher des popups.

- **Positionnement relatif (relative)**

Ce mode de positionnement est beaucoup moins répandu que les 2 autres. Il est généralement utilisé pour réaliser de petits ajustements. (picots / effets de textes)

Ce positionnement peut être utilisé à la fois sur des « block » et des « inline-block ».

On va repositionner l'élément par rapport à son point d'origine.

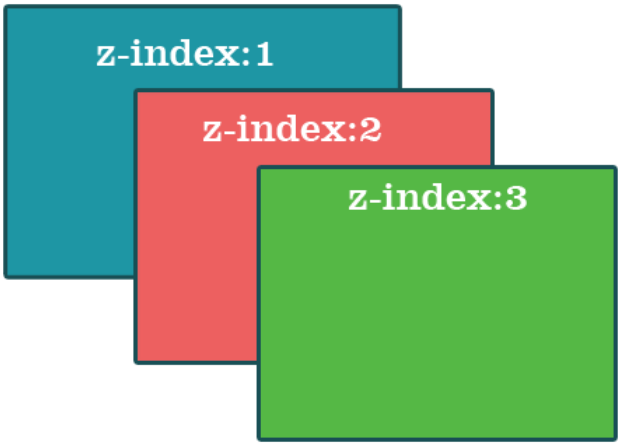


- **Définir l'ordre des calques**

Les éléments positionnés en absolu sont placés par-dessus le reste des éléments de la page.

Si vous placez deux éléments en absolu au même endroit, ils risquent de se chevaucher. Dans ce cas, utilisez la propriété CSS « z-index » afin d'indiquer quel élément doit apparaître au-dessus des autres.

L'élément ayant la valeur de z-index la plus élevée sera placé par-dessus les autres, comme le montre la figure suivante.



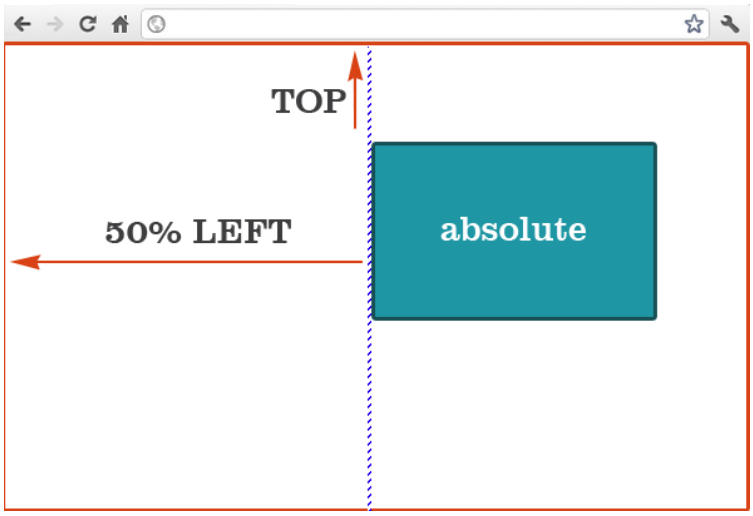

```
<div style="position:fixed;z-index:2">Contenu 1</div>
<div style="position:fixed;z-index:1">Contenu 2</div>
<!-- Contenu 1 est placé devant contenu 2 -->
```

- **Aligner des éléments absolus et fixes, horizontalement et verticalement**

Lorsque nous souhaitons aligner verticalement ou horizontalement des éléments positionnés en fixe ou en absolu, nous définissons les coordonnées de l'élément en pourcentage. Ainsi nous disposerons l'élément à 50% du bord haut ou du bord gauche par exemple.

```
<div style="position:fixed;top:10px;left:50%;"> </div>
```

Block positionné en absolu à 50% du bord gauche



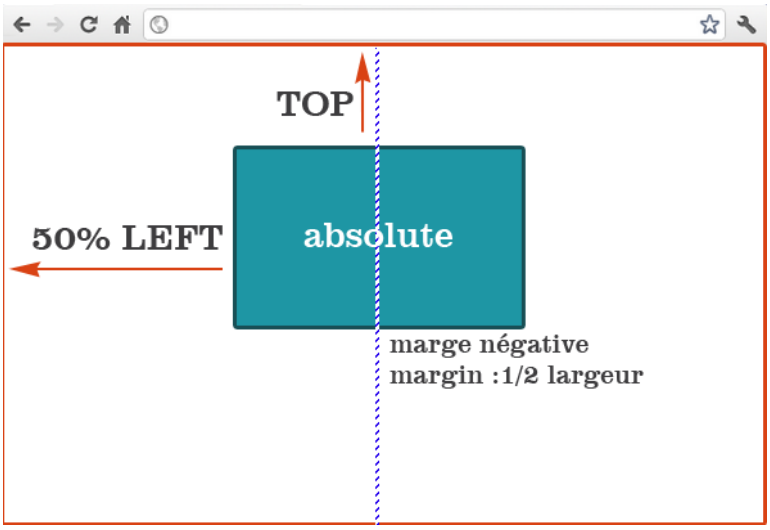
Toutefois nous pouvons observer que l'élément n'est pas centré, puisque c'est le bord gauche qui se positionne à 50% de la largeur de la fenêtre du navigateur.

Pour centrer l'élément nous allons ajouter une marge négative qui correspond à la moitié de l'une de ces dimensions. Soit la largeur, soit la hauteur. Cela dépend si le centrage est vertical ou horizontal.

```
<div style="position:fixed;top:10px;left:50%;width:300px;margin-left:-150px;"> </div>
```

Dans cet exemple nous ajoutons une marge négative à gauche de 150px, qui correspond à la moitié de 300px. En conséquence, l'élément sera décalé vers la gauche pour être vraiment centré.

Block positionné en absolu à 50% du bord gauche avec une marge négative pour le centrage



5. En résumé, guide des bonnes pratiques

- Il existe 2 unités de mesures pour définir les dimensions d'un site : le pixel et le pourcentage
- Il existe 2 types de balises : type « blocks » et type « inline »
- On utilise une balise en fonction du sens qu'elle apporte et non en fonction de son type (block ou inline)

- Chaque block a un modèle qui fait intervenir des marges, des dimensions , un design de fond et intègre éventuellement des contenus.
- Les éléments de la mise en page se positionnent selon le principe du flux de balises. Un élément peut être « hors flux » ou « dans le flux ».
- Il existe 2 méthode de positionnement :
« `static` » avec les propriétés CSS « `float` » & « `display:inline-block` » ou
« positionné » avec les méthodes « `relative, absolute, fixed` ».