TP

8

Le positionnement en CSS [a]

a. Support de TP inspiré de CSS avancées vers le html5 et CSS3, R Goetter éditions Eyrolles

8.1 Introduction

Ce Tp a pour objectif de consolider et d'élargir vos connaissances des différents schémas de positionnement classiques utilisés : le positionnement en flux, le positionnement absolu, fixé ou relatif et le positionnement flottant.
Lorsque naît le web en 1994, sa portée est tout d'abord extrêmement limitée et essentiellement destinée à un monde d'étudiants ou de scientifiques. Adaptées à des documents de type principalement textuel, où les éléments sont disposés de façon linéaire, empilés les uns sur les autres (titres, paragraphes, listes), les CSS sont dépassés dès leur apparition par un Web en pleine ébullition : gigantesque marché naissant, internet ne pouvait se réduire à de simples pages textuelles et frustrer ainsi tous les graphistes en quête de l'aspect le plus vendeur pour leur vitrine.
Très rapidement, l'utilisation de tableaux pour gérer le positionnement a été géré par les navigateurs (dès HTML 3.2). Bien que les tableaux, mais également iframe et positionnement absolu, offrent des solutions fonctionnelles aux problèmes de la mise en page, un tournant est observé après les années 2000. Le navigateur dominant de l'époque gère une syntaxe (issue de CSS1) qui sera utilisée massivement pour la mise en page : la propriété `float`. Cette propriété permet de s'affranchir du positionnement par tableaux et donc de ne plus avoir une séparation claire entre le contenu (le html) et la mise en forme (le CSS).
A l'heure actuelle, le schéma de positionnement basé sur la propriété `float` demeure le plus conseillé et le plus usité par les précheurs de la bonne parole du web accessible à tous. Nous allons toutefois revoir en détail les autres méthodes de placement d'un Web acessible à tous. Nous allons toutefois revoir en détail les autres méthodes de placement des éléments, tels que le positionnement absolu, relatif et fixe.

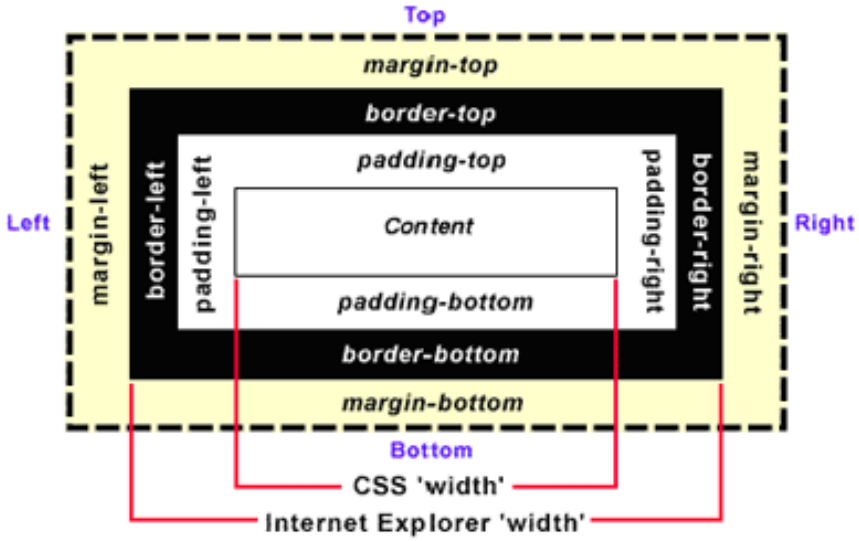
8.2 Modèle de boîte

8.2.1 Anatomie d'une boîte

Selon les spécifications du W3C, tout élément structuré par une balise HTML (ou plusieurs) se représente sous forme d'une boîte rectangulaire définie par diverses composantes que sont :

- la zone de son contenu, représentée par une largeur et une hauteur : en CSS, il s'agit des propriétés `width` et `height`
- un espace réservé à la bordure de la boîte (propriété `border`)
- une marge interne à la boîte (`padding`), se situant entre la zone de contenu et la bordure

Une dernière composante représente la marge externe (`margin`) et se situe hors de la boîte, au-delà de l'espace alloué à la bordure. Elle affecte le positionnement de l'élément puisqu'elle pousse sa boîte ou ses soeurs environnantes.
Toutes ces composantes doivent être prises en considération : contrairement à ce que beaucoup pensent, un élément n'occupe pas uniquement l'espace déterminé par sa valeur `width`, mais aussi celui de ses marges internes `padding` et ses bordures.



8.2.2 Dimensions des éléments

Le calcul de la largeur réelle d'un élément est souvent mal appréhendé car l'attribut `width` ne désigne pas la largeur entière mais uniquement celle de la partie de contenu.
Prenons par exemple un paragraphe (<p>) disposant des déclarations CSS suivantes

```
p {
  width: 100px;
  padding: 10px;
  margin: 5px;
  border: 1px solid #000;
  background-color: yellow;
}
```

Question 1
Quel est la largeur (totale) en pixels, de la boîte de paragraphe ?

Une bonne pratique de la conception web accessible à tous est de favoriser autant que possible la fluidité des éléments. Tout d'abord, il faut éviter de fixer la hauteur d'une boîte (car dépend de la taille des textes que l'internaute définit sur son navigateur). Préférez plutôt l'utilisation de taille min et max :

- `min-width` : largeur minimale (en pixel, pourcentages, `em` [1])
- `max-width`, `max-height`, `min-height`

Ces propriétés ne figent pas les dimensions de l'élément, mais se contentent de définir une valeur minimale ou maximale, quel que soit le contenu présent.
Il est d'ailleurs possible de cumuler plusieurs de ces propriétés :

```
body {
  width: 80%;
  min-width: 800px;
  max-width: 1200px;
}
```

Cet exemple permet de définir que le `body` aura une largeur fixe de 80% de la taille de l'écran, tout en fixant un minimum de 800px et un maximum de 1200px. Ce choix évite un affichage dégradé sur les petits écrans ou que le contenu, trop large, soit pénible à lire sur les très grandes surfaces.
La propriété `max-width` est particulièrement utile pour la gestion de contenus tels que `input`, `select` et `textarea` dont la quantité de contenu n'est pas connue à l'avance. Par exemple `max-width=100%` permet de restreindre l'élément désigné à ne pas déborder de son parent.

1. un bon tuto sur em : http://www.alsacreations.com/article/lire/563-gerer-la-taille-du-texte-avec-les-em.html

8.2.3 Exercice pratique : centrer horizontalement en CSS

Si une déclaration `text-align:center` appliquée à un bloc suffit de centrer un contenu textuel ou une image, l'alignement des éléments de type bloc est plus problématique. Dans cet exercice nous verrons comment centrer du contenu en utilisant uniquement les marges d'un bloc.

Question 2

- ➤ Créez le code HTML contenant un pargraphe au sein d'une balise `<div>`.
- ➤ Un paragraphe est par nature de type bloc, il occupe donc toute la largeur de son parent (ici body). Il faut donc lui attribuer une largeur fixe avant de le centrer. Fixer sa largeur à 200px. Fixez également une couleur au paragraphe et au body.
- ➤ Comme nous l'avons vu avant, l'alignement au centre du parent est défini à l'aide des marges externes. Fixez les bonnes valeurs à `margin-right` et `margin-left` pour que le paragraphe soit automatiquement centré.

8.2.4 Fusion de marges

Nous allons étudier un cas un peu plus complexe dans lequel des éléments frères ou parents on tous des définitions de marges.

Question 3

- ➤ Créez un code html contenant deux paragraphes frères
- ➤ Appliquez au premier paragraphe un fond rouge et un `margin-bottom` à 20px ;
- ➤ Appliquez uniquement au second paragraphe un fond vert et un `margin-top` à 30px ;
- ➤ A l'aide de firebug calculez la distance séparant effectivement ces deux paragraphes ?

La marge du premier paragraphe étant inférieure à celle du second, c'est celle du second qui s'applique. Les marges ne sont pas ajoutées !
Si deux blocs sont imbriqués on observe également le même comportement :

```
<div>
  <p> la marge haute va se repercuter sur le parent!</p>
</div>
```

```
div {
background-color: orange;
}
p{
  background-color:green;
  margin-top: 30px;
}
```

Par défaut le `margin-top` d'une div est de 0, elle est donc inférieur à celle du bloc enfant `p`. Toute la div sera donc décalée de 30px ;

8.3 Rendu par défaut et flux courant

Dans cette section nous allons voir comment se positionnent, en fonction de leur type, les éléments.

8.3.1 Le rendu des éléments

Tous les éléments d'une page web sont définis par la norme HTML en termes d'imbrications tolérées, mais il le sont également dans leur spécifications CSS au travers de la propriété `display` qui leur confère un certain nombre de caractéristiques :

- un rendu général sous forme de boîte
- une disposition par défaut (les uns sous les autres, à côté, en haut à gauche du parent ou sous le frère précédent)
- d'éventuelles marges internes ou externes
- la possibiité d'être redimensionnés, etc.

La propriété display possède plus de 20 valeurs. Les 4 plus utilisées sont :

- `block` (div, p, ul, h1, h2, ...) : Les éléments se placent toujours l'un en dessous de l'autre par défaut. Un élément bloc occupe automatiquement toute la largeur disponible dans son conteneur et peut être dimensionné à l'aide de propriété telles que `width, height, min-width, ...`
- `inline` (a, em, span, strong, img) : Les éléments se placent toujours l'un à côté de l'autre afin de rester dans le texte. Leur taille n'est pas définissable (par défaut), elle va être déterminée par le texte ou les éléments qu'ils contiennent. Certaines propriétés de marges peuvent être appliquées aux éléments comme les marges latérales.
- `none` (head) : l'élément est complètement retiré du flux, il n'est pas restitué par les agents utilisateurs.
- `inline-block` (input, select) : mêmes caractéristiques que les éléments `inline` mais peuvent être dimensionnés.

8.3.2 Le Flux

L'ordre dans lequel apparaissent les balises dans le code HTML est celui dans lequel les boîtes sont affichées et s'empilent dans le document. Ce schéma de positionnement par défaut se nomme le flux courant. La mise en place des différents éléments de la page s'organise par défaut selon le flux courant.
Les règles de positionnement dans le flux courant sont relativement simples et intuitives. Chaque élément :

- est situé sur le même plan que l'autre dans le flux
- se place le plus haut possible et le plus à gauche possible au sein de son parent
- est dépendant de l'élément frère précédent : deux éléments de type `block` s'empilent verticalement l'un sous l'autre, deux éléments de type `inline` se suivent sur la même ligne s'ils en ont la place

Par défaut, chaque bloc est donc dépendant de ses frères immédiats.

Question 4
Ecrire le code HTML et CSS permettant d'obtenir le rendu suivant. Observez à l'aide de firebug les balises de type `block` et `inline`.

8.3.3 Positionnement absolu

Un élément est positionné en absolu quand il est affublé de la déclaration `position:absolute`.

```
#info{
  position: absolute;
}
```

Positionner un élément en absolu le retire du flux :

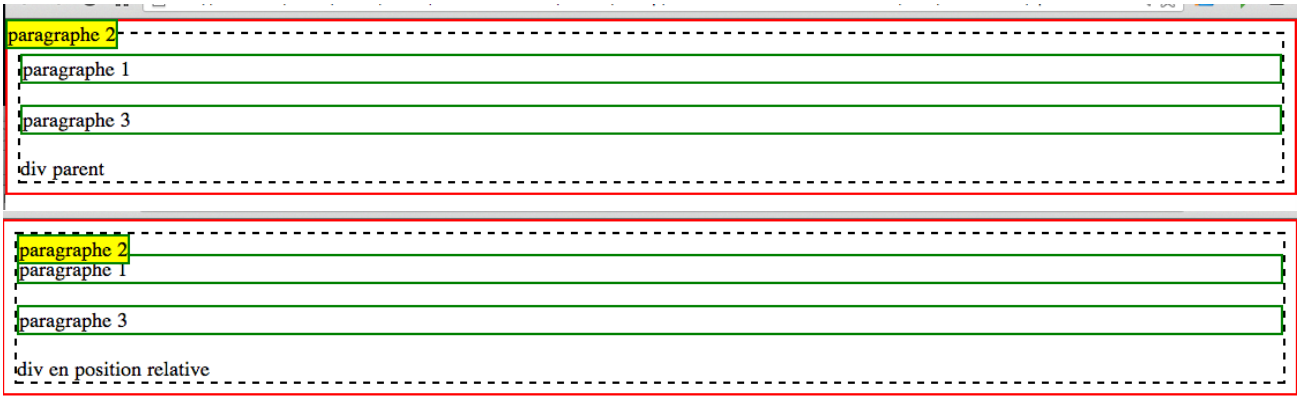
- ➤ L'élément se retrouve sur un autre plan, placé au dessus du niveau du flux (un peu comme un calque sous photoshop)
- ➤ Les éléments restants se réorganisent entre eux, dans le flux sans tenir compte de l'élément positionné en absolu hors de leur plan d'affichage

Le positionnement se fait ensuite en utilisant les propriétés `top, right, botton, left`

```
#info{
  position: absolute;
  top : 0; left : 0; /*coin superieur gauche de son referent*/
  top : 20px; left : 20px;  /*a 20 pixel de coin superieur gauche de son
      referent*/
}
```

Nous avons bien avancé sur le concept du positionnement absolu, mais il nous manque encore une donnée essentielle : qui est donc ce fameux référent mentionné partout ? L'erreur classique est de croire que le référent de tout élément positionné en absolu a comme référent l'écran. En fait, un élément positionné en absolu se place par rapport à son premier ancêtre positionné.
Le principe est le suivant : le bloc positionné en absolu remonte toute sa branche au sein de la hiérarchie dans le code HTML. Il vérifie si son parent est "positionné", c'est-à-dire s'il est muni de la propriété `position` à laquelle est associée une valeur. Si tel n'est pas le cas, il remonte d'une génération et ainsi de suite jusqu'à trouver un ancêtre positionné (en dernier recours `<html>` sera son référent).
Nous comprenons donc qu'il devient aisé d'indiquer une référence à un bloc absolu : il suffit d'appliquer une déclaration `position: absolute` `position: fixed` ou `position: relative` à cet élément de référence.

Question 5
Créez le code HTML et CSS permettant de reproduire les deux cas de figure illustrés dans les images ci-dessous. Jouez sur la valeur de l'attribut `position` de la balise `div` contenant les 3 paragraphes et observez les différences de positionnement absolu du second paragraphe.



8.3.4 Le positionnement relatif

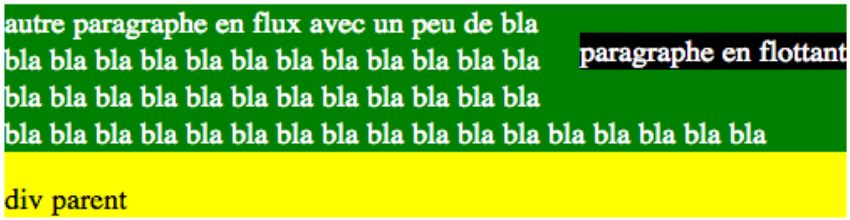
Un élément relatif se place par rapport à sa position classique dans le flux, puis est éventuellement décalé si au moins une des propriétés `top,right,bottom,left` est renseigné. La notion de relatif s'applique par conséquent à son placement initial dans le flux. Ici, les propriétés `top,right,bottom,left` servent à indiquer un décalage par rapport à la position initiale.

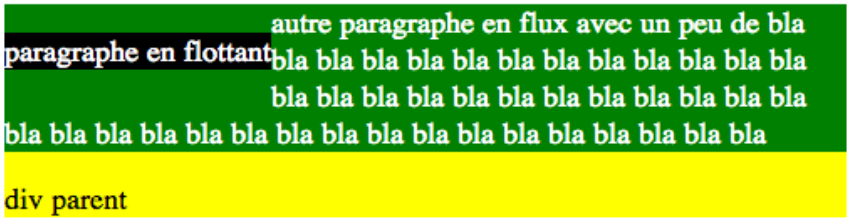
```css
strong{
  position: relative;
  //decalage de 10 vers le bas et a droite par rapport au flux courant
  top: 10px;
  left:10px;
}
```

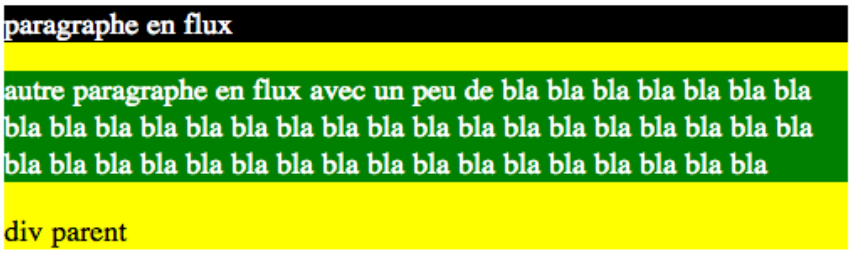
8.3.5 Le positionnement flottant

Par définition, un élément flottant est ôté du flux et placé à l'extrême gauche (`float: left`) ou droite (`float: right`) de son conteneur, tout en demeurant sur sa hauteur de ligne initiale dans le flux.
Dans les faits, le positionnement flottant s'avère complexe à utiliser car si l'élément est extrait du flux courant, il ne l'est que partiellement : les éléments précédant le bloc flottant ne sont pas affectés ; cependant, tous les éléments suivants se réorganisent dans le flux comme dans le cas du positionnement absolu, sauf ... leur contenu qui, lui, s'écoule autour du flottant en épousant sa forme.
il est par conséquent nécessaire d'être très attentif au modèle de boîte des éléments suivant le flottant : seule la composante de contenu de la boîte s'écoule autour de l'élément flottant qui la précède. En revanche, la boîte elle-même se repositionne dans le flux à la suite de la boîte en flux précédent. Vous suivez toujours ?

Question 6
Créez le code HTML et CSS permettant de reproduire les 3 cas de figure illustrés dans les images ci-dessous.
Jouez sur la valeur de l'attribut `float` de la balise `p`. Observez comment se comporte le second paragraphe.







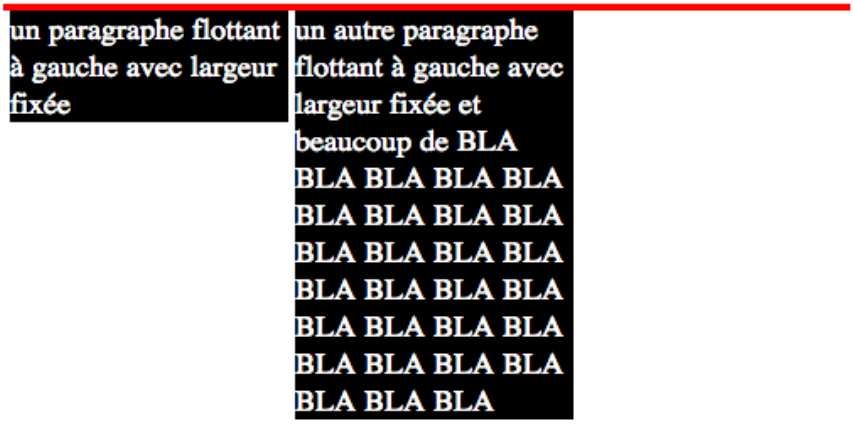
A l'instar du positionnement absolu, un élément flottant adopte par défaut la largeur qu'occupe son contenu. Si celui-ci est dense, elle est susceptible d'occuper toute la largeur du parent, c'est pourquoi il est souvent nécessaire de fixer une largeur au flottant via la propriété `width` ou `max-width`.
Lorsqu'un élément flottant est poussé dans la même direction qu'un autre élément flottant, il demeure sur le même plan et se cale à ses côtés (très utilisé pour les menus). Attention, gardez à l'esprit que le flottement est un positionnement hors flux et qu'il n'a plus de prise sur les blocs alentours en flux, à l'exception des contenus qui vont "épouser" les

flottants. Cela signifie qu'un parent ne contenant que des flottants n'occupera physiquement aucune surface à l'écran, ou encore que les objets flottants vont dépasser de leur conteneur.

Question 7
Soit le code HTML suivant :

```
<body>
<div>
  <p> un paragraphe flottant a gauche avec largeur fixee </p>
  <p> un autre paragraphe flottant a gauche avec largeur fixee et
      beaucoup de BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA
      BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA BLA </p>
</div>
</body>
```

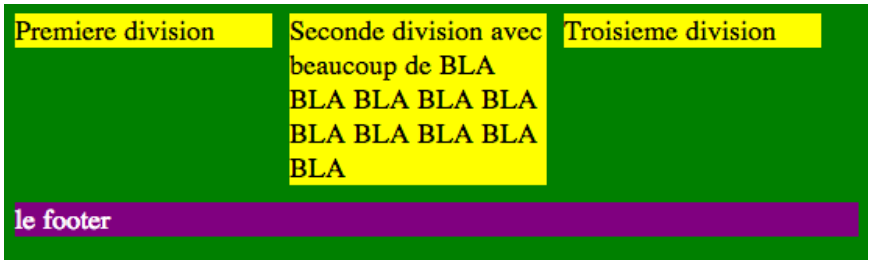
Créez le CSS permettant (comme sur l'image ci-dessous) d'illustrer le cas de figure pour lequel le parent `div` vide de contenu (excepté les paragraphes) n'occupera pas physiquement de place sur la page.



8.3.6 La propriété clear

La propriété `clear` est une sorte de passerelle entre deux plans du document : celui du flux courant et celui des flottants. elle interdit à un élément de se placer sur la même ligne qu'un bloc flottant et le force par conséquent à se caler directement en dessous de celui-ci. Elle autorise par ailleurs un positioner des flottants exclusivement à gauche (`clear: left`), à droite (`clear:right`) ou les deux simultanément (`clear:both`).

Question 8
Créez le code HTML et CSS permettant d'obtenir la même disposition que sur l'image ci-dessous.

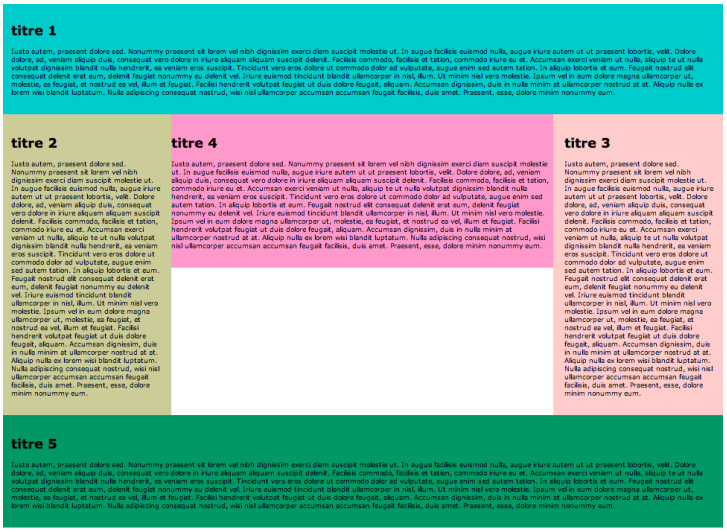


8.4 Exercices

Question 9
Créez le code HTML et CSS permettant d'obtenir la même disposition que sur l'image ci-dessous.

Question 10
Créez le code HTML et CSS permettant d'obtenir la même disposition que sur l'image ci-dessous.

Question 11
Créez le code HTML et CSS permettant d'obtenir la même disposition que sur l'image ci-dessous.



S'il vous reste du temps, vous pouvez aller vous mesurer à un exercice préparé par les développeurs de Alsacréation : `http://www.mammouthland.net/parisweb2007/`