



# Mise en boîte (partie 2/2)

## Informations sur le tutoriel

[Ajouter à mes tutoriels favoris](#) (7 fans)



Auteur : [M@teo21](#)

Licence :

[Plus d'informations](#)

## Popularité

Visualisations : 102 458 959

Appréciation 15

des lecteurs :15

60

888

## Publicité

## Historique des mises à jour

- *Le 08/08/2010 à 03:23:12*  
Correction conjugaison.
- *Le 03/08/2010 à 01:07:48*  
Correction orthographique, #2605
- *Le 31/07/2010 à 18:20:55*  
Correction orthographique, ticket n°2593

Dans la première partie de ce chapitre "Mise en boîte", nous avons fait le tour des propriétés CSS permettant de modifier l'apparence et la taille des blocks. Pour résumer rapidement, on a vu :

- Comment modifier la taille des blocks
- Comment modifier la bordure des blocks
- Comment modifier la marge des blocks

Dans cette seconde partie, nous allons apprendre à les positionner. Bah oui, il va bien falloir apprendre à dire : "Je veux que mon menu soit à gauche, que mon contenu soit à droite, qu'il y ait le logo de mon site en haut à droite à 12 pixels du bord droit" etc...

Autant que les choses soient claires : le positionnement en CSS, c'est pas franchement évident... Il y a la théorie, et il y a la pratique. Dans ce chapitre, nous verrons seulement la théorie (c'est un passage obligé), et dans le chapitre suivant nous passerons à la pratique (ce qui sera beaucoup plus concret pour vous)

Une chose est sûre, et vous devriez déjà vous mettre ça en tête, c'est que vous ne pourrez jamais avoir un site qui s'affiche pareil au pixel près sur tous les navigateurs. A la place, nous apprendrons à faire en sorte que notre site s'affiche "à peu près correctement" sur chacun de ces navigateurs, et ça sera déjà pas si mal 😊

Sommaire du chapitre :



- [Transformer un inline en block \(et vice-versa\)](#)
- [Les flottants](#)
- [Positionnement absolu, fixe et relatif](#)
- [Q.C.M.](#)

## Transformer un inline en block (et vice-versa)

Dans le genre, "je suis là juste pour vous embrouiller l'esprit", je crois que je vais gagner le premier prix là 😊  
Je vais vous apprendre ici à ~~repousser les lois de la physique~~ à modifier les lois du CSS (brrr...).

Les images ( `<img />` ) sont des balises de type inline, et vous préféreriez que ce soient des blocks ? Pas de problème !

Les titres ( `<h1>` ) sont de type block, et ça vous a toujours embêté, vous préféreriez que ça soient des balises inline ? Aucun problème !

Il existe une propriété CSS très puissante, elle s'appelle *display* (à manipuler avec précaution, sinon vous risquez de tout faire péter :p)

Cette propriété peut prendre les valeurs suivantes :

- **block** : la balise concernée deviendra de type block.
- **inline** : la balise concernée deviendra de type inline.

En fait, *display* peut prendre beaucoup d'autres valeurs (c'est ça d'être une propriété puissante ;)), mais je vous en fais volontairement grâce et je vous montre juste les 2 plus utilisées.

Pour transformer une balise inline en balise de type block, vous allez devoir taper :  
`display: block;`

Par exemple, si je veux que TOUTES mes images soient de type block, j'écrirai donc ceci :

Code : CSS

```
img
{
    display: block;
}
```

Attention, après avoir fait cela toutes vos images vont aller automatiquement à la ligne, comme le font les autres balises de type block.

Rien n'impose que TOUTES vos images soient de type block, hein. J'espère que vous l'avez compris depuis le temps : on peut très bien utiliser l'attribut "class" sur une balise pour qu'elle ait une présentation différente. Par exemple :

Code : HTML

```


```

Bien entendu, il faudra écrire le code CSS suivant pour que la deuxième image soit de type block :

Code : CSS

```
.imageblock
{
    display: block;
}
```

J'espère que je ne vous apprends rien sur le fonctionnement de l'attribut *class*, sinon retournez voir le premier chapitre de la partie II au triple galop 🤪

Et l'inverse alors ? Pour transformer une balise de type block en balise de type inline ?

Bah c'est pareil, sauf qu'on utilise *display:inline*; 😊

Je ne vous refais pas un exemple de code, je pense que vous êtes assez grands pour deviner tous seuls comment il faut faire 😊

Petite remarque en passant : on transforme généralement des inline en block, mais on fait plus rarement l'inverse. Il faut dire que les balises block ont pas mal d'avantages : on peut modifier leur taille, leur bordure, leurs marges etc... Bref, tout ça vous le savez déjà 😊

## Les flottants

La première technique que nous allons voir, ce sont les **flottants**. C'est un nom un peu bizarre je vous l'accorde, mais c'est en fait pas si sorcier que ça.

Pour que vous voyiez bien de quoi on parle, voici ce que nous allons apprendre à faire :



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

*Une image flottant à gauche*

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

*Une image flottant à droite*



Cela permet en gros d'"entourer" un élément (ici une image) par du texte. Ça permet d'avoir une jolie présentation facilement.

J'imagine que la question qui vous brûle les lèvres maintenant est : "Mais quelle est donc la propriété magique qui fait flotter ?!".

La réponse est... `float` ("flottant" en anglais). Cette propriété peut prendre 2 valeurs, que vous pourriez d'ailleurs avoir deviné tous seuls 😊 :

- `left` : l'élément flottera à gauche.
- `right` : l'élément flottera... à droite ! Oui bravo 😊

L'utilisation des flottants est très simple, tellement simple qu'on a parfois tendance à se compliquer la vie (et je parle en connaissance de cause). En fait, il n'y a que 2 choses à faire :

1. Vous appliquez un `float` à une balise.
2. Puis vous continuez à écrire du texte à la suite normalement.

On peut aussi bien utiliser la propriété *float* sur des balises block que sur des balises inline, vous allez voir...

## Faire flotter une image

Nous allons apprendre ici à faire flotter une image (qui est de type "inline" je vous rappelle). Voici le code XHTML que nous devons taper dans un premier temps :

Code : HTML

```
<p>
<p>Ceci est un texte normal de paragraphe, écrit à la suite de l'image mais
```

Une erreur courante que l'on a tendance à faire, c'est de mettre l'image flottante après le texte. Non, il ne faut pas : le flottant doit être avant le texte, qu'il flotte à droite ou qu'il flotte à gauche c'est pareil.

Vous essaieriez de mettre la balise `<img />` après le paragraphe, et vous verrez que ça ne marche plus 😊

Ce qu'il faut bien comprendre maintenant, c'est qu'on a juste besoin de modifier le CSS de l'image, et non pas du paragraphe. Le paragraphe n'a pas besoin d'être modifié, il sera automatiquement placé comme il faut. Voici le seul bout de code CSS qu'on a besoin de taper, qui permet de faire flotter l'image à gauche :

Code : CSS

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

[Essayer !](#)

Amusez-vous aussi à faire flotter l'image à droite, c'est tout bête : mettez `float:right`; et le tour est joué ! 😊

## Créer une lettrine (exercice)

Parfois, il m'arrive de faire un cauchemar. J' imagine que je suis en train de rédiger un long cours sur les flottants en CSS, et que tout le monde bâille, s'endort, bref s'ennuie. Brrr... Cette vision me fait froid dans le dos à chaque fois

Alors, pour éviter que le pire de mes cauchemars se produise, j'ai eu l'idée de vous faire travailler un peu sur un exercice amusant (du moins je l'espère :p).

Vous savez ce qu'est une lettrine ? Non.

Bon c'est pas gagné 😊

Une lettrine, c'est la première lettre d'un paragraphe écrite beaucoup plus grosse. On en voit tout le temps dans les journaux (si vous lisez le journal ;o). Voici ce que vous devez arriver à faire :

**B**onjour tout le monde !  
 commence par une lett  
 pas plus joli comme ça  
 trouve que ça invite à la lectur  
 ? Faire une lettrine en CSS, c

La seule consigne, c'est d'utiliser le code XHTML suivant, que vous n'avez pas le droit de changer :

Code : HTML



Votre consigne, c'est de créer un fichier CSS qui permettra de créer cette présentation sous forme de lettrine. Vous ne pouvez pas modifier le fichier XHTML, mais par contre sur le CSS tous les coups sont permis : vous pouvez utiliser les propriétés que vous voulez.

Au boulot ! 😊

...  
 ...  
 ...

Drrring ! Allez c'est l'heure de la correction !

Bien évidemment il fallait utiliser un *float:left*;, mais comment faire pour que la première lettre uniquement soit affectée ? Sans changer le XHTML ?

Si vous avez trouvés tous seuls qu'il fallait utiliser le pseudo-élément :first-letter (= "première lettre"), eh bien bravo ! C'était la seule vraie difficulté, après le reste coule de source et on peut s'amuser à mettre les propriétés qu'on veut.

Il faut penser en particulier à utiliser font-size, pour modifier la taille de la première lettre pour qu'elle fasse disons... 3 hauteurs de lignes. Pour cela, j'ai mis la valeur "3em" (= 3 hauteurs de lignes), mais si vous avez tapé "300%" c'est tout aussi bon ! Par contre, il vaut mieux éviter de mettre une valeur en pixels comme "50px" parce qu'on ne connaît pas la taille d'une ligne (ça peut changer).

Voici le code que j'ai fait pour réaliser la lettrine :

Code : CSS

```
p:first-letter /* Je veux que la première lettre de mes paragraphes... */
{
  float: left; /* Flotte à gauche */
  font-size: 3em; /* Fasse une hauteur de 3 lignes */
  font-family: Arial, Georgia, "Times New Roman", Times, serif; /* Soit m
  font-weight: bold; /* Soit écrite en gras (c'est plus voyant) */
  margin-right: 5px; /* Qu'il y ait une marge de 5px à droite pour que ça
```

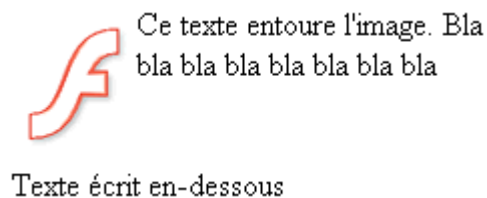
[Essayer !](#)

Ne vous arrêtez pas là ! Rien ne vous empêche d'écrire la lettrine en blanc sur fond noir, ou même d'utiliser une image de fond ! Laissez donc libre cours à votre imagination 😊

## Stopper un flottant

Quand vous mettez en place un flottant, le texte l'"entoure", ça on l'a compris. Mais comment faire si vous voulez qu'au bout d'un moment le texte continue **en-dessous** du flottant ? On pourrait faire plusieurs `<br />` à la suite, mais ça serait pas pratique ni très propre...

En gros, on aimerait pouvoir faire ça :



Il existe en fait une propriété CSS (sans blague) qui permet de dire : "Stop, ce texte doit être en-dessous du flottant et non plus à côté". C'est la propriété *clear* qui peut grosso modo prendre 3 valeurs :

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

Pour simplifier, on va utiliser tout le temps le `clear:both`, qui marche après un flottant à gauche et après un flottant à droite (ça marche à tous les coups quoi).

Pour illustrer son fonctionnement, on va prendre ce code XHTML :

Code : HTML

```
<p>
<p>Ce texte est écrit à côté de l'image flottante.</p>
<p class="dessous">Ce texte est écrit sous l'image flottante.</p>
```

Code : CSS

```
.imageflottante
{
    float: left;
}
.dessous
{
    clear: both;
}
```

[Essayer !](#)

Et voilà le travail 😊

On applique un `clear:both`; au paragraphe que l'on veut voir continuer sous l'image flottante, et le tour est joué !

## Positionnement absolu, fixe et relatif

Hormis les flottants, il existe 3 façons de positionner un block en CSS :

- **Le positionnement absolu** : il nous permet de placer un block n'importe où sur la page (en haut à gauche, en bas à droite, tout au centre etc...)
- **Le positionnement fixe** : c'est pareil que le positionnement absolu, mais cette fois le block reste toujours visible, même si on descend plus bas dans la page. C'est un peu comme *le background-attachment:fixed*; (si vous vous en souvenez encore ;))
- **Le positionnement relatif** : c'est le plus compliqué des trois. En gros, ça nous permet de déplacer un block par rapport à sa position normale.

Comme pour les flottants, le positionnement absolu, fixé et relatif fonctionne aussi sur des balises de type inline comme `<img />`

Toutefois, vous verrez qu'on l'utilise bien plus souvent sur des balises block que sur des balises inline.

Il faut d'abord faire son choix entre les 3 modes de positionnement disponibles. Pour cela, on utilise la propriété CSS `position` à laquelle on donne une de ces valeurs :

- **absolute** : positionnement absolu.
- **fixed** : positionnement fixe.
- **relative** : positionnement relatif.

Nous allons étudier chacun de ces positionnements un à un.

### Le positionnement absolu

Vous savez que, pour effectuer un positionnement absolu, il faut taper :  
`position:absolute;`

Mais ça ne suffit pas ! 💡

On a dit qu'on voulait un positionnement absolu, mais encore faut-il dire **où on veut que le block soit positionné sur la page**.

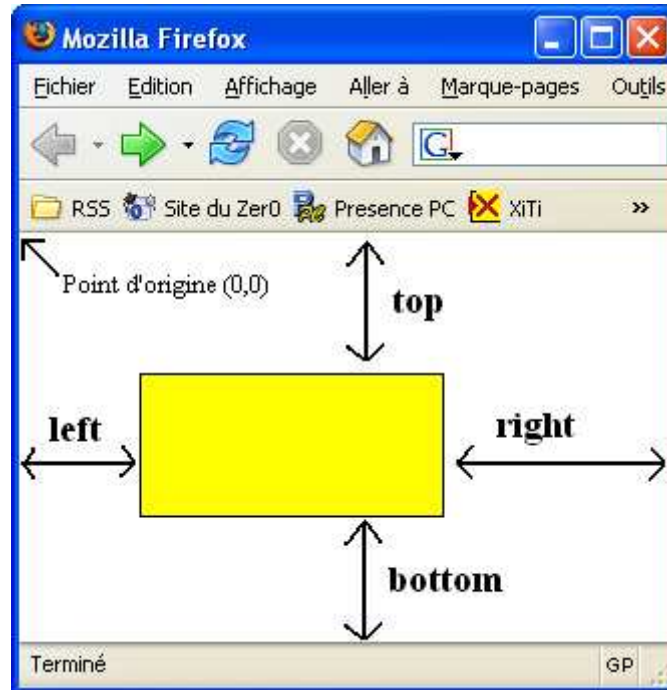
Pour faire cela, on va réutiliser 4 propriétés CSS que vous commencez certainement à bien connaître :

- **left** : position par rapport à la gauche de la page.
- **right** : position par rapport à la droite de la page.

- **top** : position par rapport au haut de la page.
- **bottom** : position par rapport au bas de la page.

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

Si ce n'est pas très clair pour certains d'entre vous, ce schéma devrait vous aider à comprendre :



Avec ça, vous devriez être capables de positionner correctement votre block 😊

Il faut donc utiliser la propriété *position* et au moins une des 4 propriétés ci-dessus (top, left, right ou bottom).

Si on écrit par exemple :

```
position: absolute;
right: 0px;
bottom: 0px;
```

... cela signifiera que le block doit être positionné tout en bas à droite (0 pixels par rapport à la droite de la page, 0 par rapport au bas de la page).

Pour le coup, on va utiliser la balise universelle <div> (de type block). Je vais rédiger un paragraphe, et je vais mettre dans le block <div> deux ou trois mots :

#### Code : HTML

```
<p>
Ceci est un paragraphe normal.<br />
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer vel libero.
</p>

<div>Block positionné à droite</div>
```

[Essayer !](#)

Rien d'extraordinaire, vous avez un paragraphe suivi d'un block div qui contient un peu de texte. Jusque là, rien de bien excitant.

Maintenant, ajoutons un peu de CSS pour positionner où on veut notre block div :

#### Code : CSS



```
div
{
  background-color: yellow;
  border: 1px solid green;

  position: absolute;
  left: 35px;
  top: 50px;
}
```

[Essayer !](#)

J'ai mis un fond jaune et une bordure pour qu'on repère mieux le block.

Comme vous pouvez le constater, le block se met au-dessus du paragraphe. Il le masque. Ca, c'est un des grands avantages et défauts du positionnement absolu : il n'y a aucun contrôle, vous pouvez mettre votre block vraiment où vous voulez sur la page, mais il faut faire attention à ce qu'il ne masque pas de texte.

Voici un autre exemple pour vous montrer qu'on peut vraiment mettre le block n'importe où :

Code : CSS

```
div
{
  background-color: yellow;
  border: 1px solid green;

  position: absolute;
  right: 50%;
  bottom: 20px;
}
```

[Essayer !](#)

Une petite précision technique pour ceux qui veulent aller plus loin (les autres fermez les yeux) : si vous positionnez un block A en absolu, et qu'à l'intérieur de ce block vous positionnez un autre block B en absolu, ce positionnement-là ne se fera plus par rapport au coin en haut à gauche de la fenêtre mais par rapport au coin en haut à gauche du block A.

Ah... Les joies du positionnement CSS 😊

## Le positionnement fixe

Le fonctionnement est **exactement le même** que pour le positionnement absolu, sauf que cette fois le block reste fixe, même si on descend plus bas dans la page.

Au lieu de faire *position:absolute;*, on va taper *position:fixed;* dans notre CSS. Comme tout à l'heure, on s'aide de top, left, right et bottom pour placer notre block où on veut sur la page.

Le *position:fixed;* est particulièrement utile pour faire un menu qui reste toujours visible :

Code : CSS

```
div
{
  background-color: yellow;
  border: 1px solid green;
  width: 150px;
  height: 250px;

  position: fixed;
  right: 40px;
  top: 60px;
}

p
{
  width: 300px;
}
```

[Essayer !](#)

## Le positionnement relatif

Plus délicat, le positionnement relatif peut vite devenir une grosse prise de tête si on n'y fait pas très attention. Personnellement, je ne l'utilise pas beaucoup. Le positionnement relatif sert généralement à faire des "ajustements".

Prenons par exemple un texte important, situé entre 2 balises <strong>. Pour commencer, je le mets sur fond rouge pour qu'on puisse mieux le repérer :

Code : CSS

```
strong
{
  background-color: red; /* Fond rouge */
  color: yellow; /* Texte de couleur jaune */
}
```

[Essayer !](#)

Cette fois, le schéma que je vous ai montré tout à l'heure pour les positions absolue et fixe ne marche plus. Pourquoi ? Parce que l'origine a changé : le point de coordonnées (0, 0) ne se trouve plus en haut à gauche de votre fenêtre comme c'était le cas tout à l'heure. Non, cette fois l'origine se trouve en haut à gauche de la position actuelle de votre élément. Tordu n'est-ce pas ? Bah oui, c'est une position relative. Ce schéma devrait vous aider à comprendre où se trouve l'origine des points :


  
 Pas de doute, **ce texte est important** si on veut comprendre corre

Donc, si vous faites un *position:relative* et que vous appliquez une des propriétés *top*, *bottom*, *left*, *right*, le texte sur fond rouge va se déplacer par rapport à la position où il se trouve.

Je vous l'avais dit que c'était tordu 🤔 En fait, il faut faire des tests pour bien comprendre.

Prenons donc un exemple : je veux que mon texte se décale de 55 pixels vers la droite et de 10 pixels vers le bas. Je vais donc demander à ce qu'il soit décalé de 55 pixels par rapport au "bord gauche", et de 10 pixels par rapport au "bord haut" :

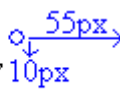
### Code : CSS

```
strong
{
    background-color: red;
    color: yellow;

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

[Essayer !](#)

Le texte s'est décalé par rapport à sa position initiale, comme ceci :

Pas de doute,  **ce texte est important** si on veut com

Voilà le principe. A vous de voir si vous avez besoin de l'utiliser, maintenant vous savez comment ça marche 😊

## Q.C.M.

Quelle propriété CSS permet de transformer un inline en block et inversement ?

- ☐ position
- ☐ display
- ☐ blockstyle

Si je mets display:block; à la balise span, que va-t-il se passer ?

- ☐ Un bug dans la matrice
- ☐ Ce n'est pas possible
- ☐ Ca va fonctionner comme une balise div

Lequel de ces positionnements n'existe pas ?

- ☐ absolute
- ☐ active
- ☐ fixed

Si je fais flotter une image à droite avec un float:right, et que j'applique au paragraphe qui suit un clear:left, que va-t-il se passer ?

- ☐ Le paragraphe continuera sous l'image
- ☐ Le paragraphe continuera au-dessus de l'image
- ☐ Le paragraphe continuera à gauche de l'image

Si je veux que mon bloc soit positionné de façon absolue en haut à droite, avec un décalage de 10 pixels,

comment dois-je procéder ?

- ☐ position:absolute; bottom:10px; right:10px;
- ☐ position:absolute; top:10px; right:10px;
- ☐ position:absolute; top:10px; left:10px;

Je veux procéder à un ajustement. Je souhaite que mes titres soient tous décalés de 5 pixels vers la gauche et de 4 pixels vers le bas. Quels valeurs top, bottom, left ou right utiliser avec mon position:relative ?

- ☐ top:4px; right:5px;
- ☐ bottom:4px; left:5px;
- ☐ top:4px; left:5px;

Si un bloc est positionné de façon absolue, où se trouve l'origine des points ?

- ☐ En haut à droite de la page
- ☐ En haut à gauche de la page
- ☐ En haut à gauche du bloc

Même question pour un bloc positionné de façon relative.

- ☐ En haut à droite de la page
- ☐ En haut à gauche de la page
- ☐ En haut à gauche du bloc

Correction !

[Statistiques de réponses au QCM](#)

---

Comme quoi, être attentif en cours de maths ça peut servir : au moins vous savez ce que c'est l'origine d'un repère (le point de coordonnées 0, 0).

Dès qu'on parle de positionnement absolu, il y a forcément une origine... Cela nous permet de placer un block (ou même une balise inline comme img) n'importe où sur la page !

Avec ces connaissances-là, vous êtes prêts pour la suite...

D'ailleurs, la suite du programme c'est quoi déjà ?... Ah oui ! La suite, c'est un TP (= Travaux Pratiques)

En effet, le chapitre suivant ne va rien vous apprendre de nouveau, mais il va pourtant beaucoup vous intéresser : je vais vous montrer comment on crée le design d'un site de A à Z avec ce qu'on a appris ! 😊

Eh oui, vous savez tout ce qu'il faut, encore faut-il maintenant savoir s'y prendre... Quand vous êtes prêts, on y va 😊