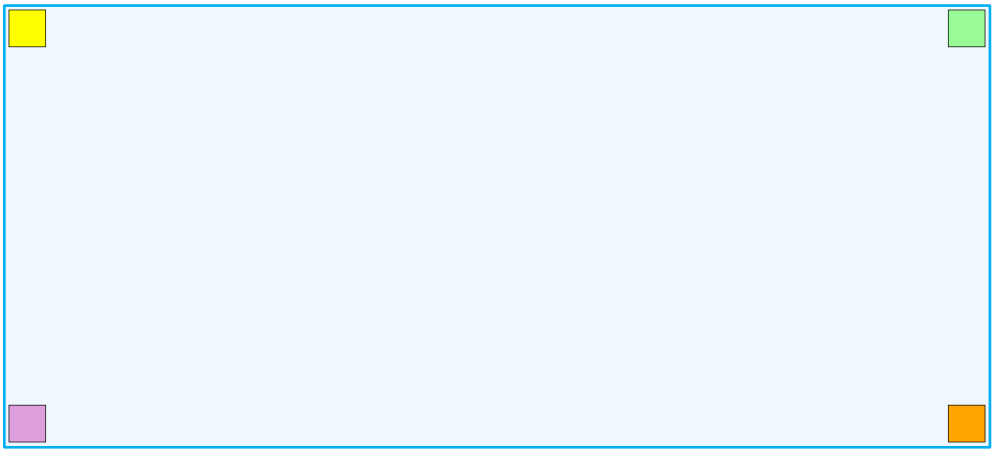
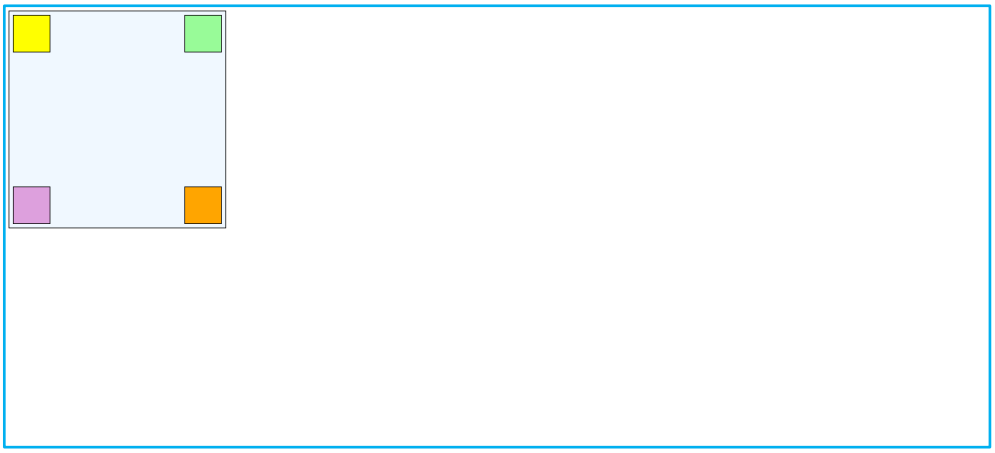
Exercice 1 :

Faites tous les exemples cités en annexe.

Exercice 2 :

Réalisez les pages suivantes :

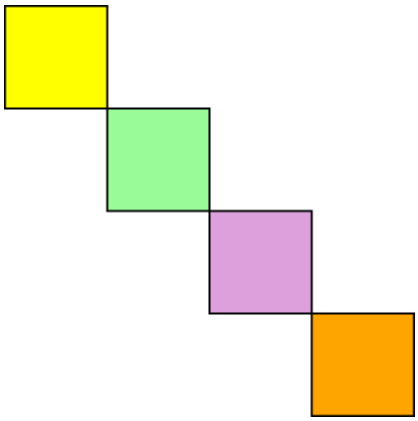




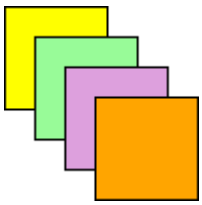
Couleurs : AliceBlue, LightGoldenRodYellow, PaleGreen, Plum, Orange

Exercice 3 :

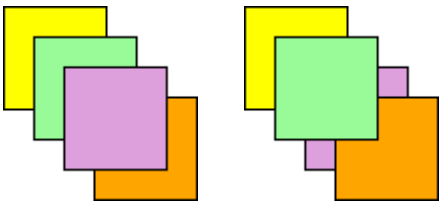
1. Réalisez la page suivante :



2. Améliorez la page comme suit :



La division survolée doit être placé au-dessus des autres :



Exercice 4 :

1. Observez bien chacune des figures suivantes répondez à la question suivante : Quelles sont les propriétés qui ont été appliquées aux différents éléments de l'image pour obtenir le rendu final ?

Figure 1

Figure 2

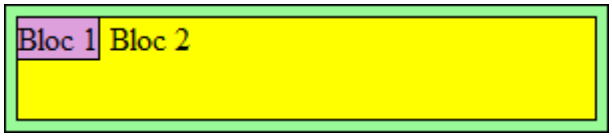


Figure 3

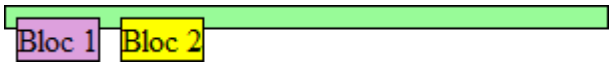


Figure 4

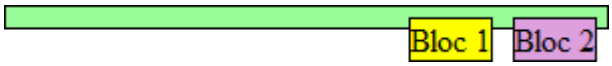


Figure 5

Figure 6

Figure 7

Figure 8

2. Réalisez les pages précédentes.

Annexe

Modèle de boîte

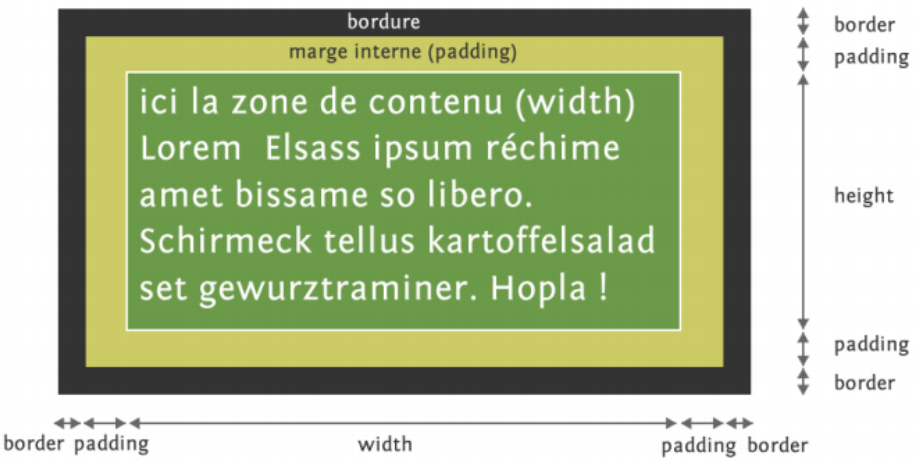
Anatomie d'une boîte

Selon les spécifications 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 ;
- 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 sœurs 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 et ses bordures.



Dimensions des éléments

Le calcul de la largeur "réelle" d'un élément est souvent mal appréhendé par les débutants, car il peut sembler illogique au premier abord. En réalité, c'est le terme de width qui est source de confusion, car il ne désigne pas la largeur de la boîte entière, mais uniquement celle de la partie de contenu. Le terme de content-width ou inner-width eût été moins trompeur.

Avec un peu d'expérience, jongler avec ces composantes devient très vite intuitif et les erreurs de dimensions se dissipent.

Prenons par exemple un paragraphe disposant des déclarations CSS suivantes :

```
<html>
    <head>
        ...
    </head>
    <body>
        <p>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit ...
        </p>
    </body>
</html>
```

```
html
{
    border:1px solid green;

}
body
{
    border:1px solid blue;
}
p
{
    width:200px;
    padding:10px;
    margin:5px;
    border:5px solid black;
    background-color:yellow;
}
```

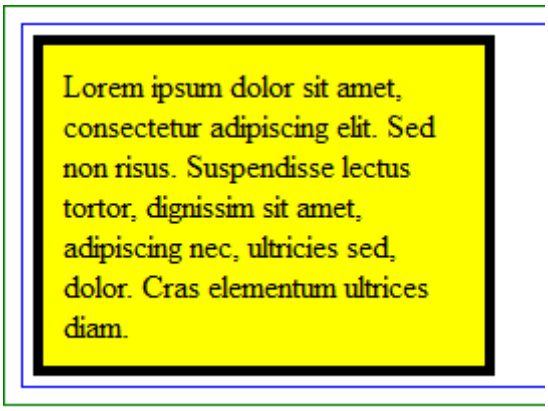


Figure 1 : Une boîte de 230 de large

La largeur de la boîte de paragraphe est dans cet exemple de… 230 pixels. En effet, les composantes à prendre en compte sont width, padding et border. La marge externe (margin) n'entre pas dans le calcul

de la largeur de l'élément. La propriété padding exprime les marges tout autour de l'élément, il faut donc la compter de chaque côté. Il en est de même pour la propriété border. Au final, nous additionnons 200px (width) + 20px (padding droit et gauche) + 10px (border droite et gauche) pour parvenir à une largeur réelle de 230 pixels.

Le mode Quirks de Microsoft

Le modèle de boîte proposé par le W3C est adopté par l'ensemble des navigateurs du marché, pour le plus grand bonheur des concepteurs web que nous sommes. Sachez cependant que ce ne fut pas toujours le cas, puisque du temps de l'apogée d'Internet Explorer 5.0 et 5.5, Microsoft avait décidé d'adopter un modèle de boîte différent de celui proposé par le Consortium. Ce modèle de boîte assimilait les composantes de marges internes (padding) et les bordures (border) à la zone de contenu (width).

Reprenons notre paragraphe précédent et les styles associés. La largeur de la boîte de paragraphe dans ce modèle Microsoft sera tout simplement de 200 pixels. Cela signifie que la zone de contenu sera réduite à 170 pixels.

Ce mode de calcul, où les propriétés visibles (espaces et bords) sont incluses dans le calcul de la boîte, est appelé mode Quirks. C'est un modèle de boîte ancien et spécifique à Microsoft. Fort heureusement, depuis la version IE6, ce mécanisme propriétaire disparaît au profit du modèle de boîte dit Standard, où l'interprétation des dimensions des boîtes est la même que pour les autres navigateurs.

Exercice pratique : centrer horizontalement en CSS

Dès les premières moutures du langage CSS, les alignements horizontaux sont pris en compte au sein des spécifications.

Si une déclaration text-align:center appliquée à un bloc suffit à centrer horizontalement le contenu textuel ou les images au sein de celui-ci, l'alignement des éléments de type bloc est généralement plus problématique. Pourtant, l'une des particularités du modèle de boîte, et plus précisément de ses marges externes, permet de centrer simplement les blocs.

Prenons dans l'exemple d'un paragraphe que nous souhaiterions centrer horizontalement au sein d'un élément div.

Puisqu'un paragraphe est de type bloc par nature, il occupe forcément toute la largeur de son parent. Il est par conséquent inutile de tenter de le centrer sans lui attribuer de largeur. Notre première mission consiste donc à intervenir sur cette composante de boîte. Choisissons une largeur de 200 pixels :

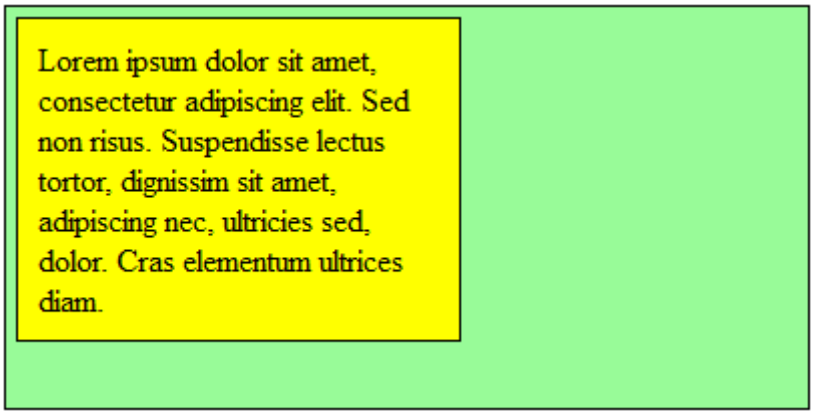


Figure 2 : Paragraphe de 200 pixels prêt à être centré

```
div
{
    width:400px;
    height:200px;
    background-color:PaleGreen;
    border:1px solid black;
}
p
{
    width:200px;
    padding:10px;
    margin:5px;
    border:1px solid black;
    background-color:yellow;
}
```

L'alignement au centre du parent est défini à l'aide de marges externes latérales automatiques, c'est-à-dire en appliquant la valeur auto :

```
p
{
    width:200px;
    padding:10px;
    margin-top:5px;
    margin-bottom:5px;
    margin-right:auto;
    margin-left:auto;
    border:1px solid black;
    background-color:yellow;
}
```

À ce stade, le paragraphe large de 200 pixels est centré horizontalement au sein de son conteneur. C'est aussi simple que cela et cette méthode est reconnue depuis Internet Explorer 6.

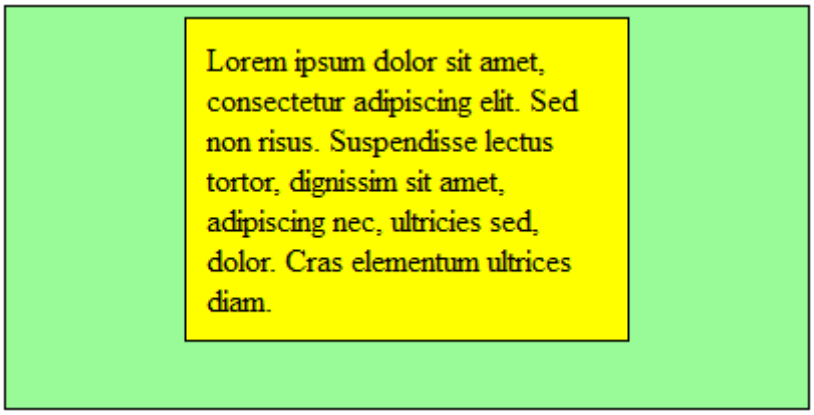


Figure 3 : Des marges automatiques et c'est centré !

Nous pouvons optimiser le code en employant la syntaxe raccourcie de la propriété margin, par exemple :

```
margin:5px auto;
```

Dans cet extrait, nous affectons les marges externes ainsi : 5px en haut et en bas, auto à droite et à gauche.

Notez que les marges automatiques appliquées en haut et en bas n'ont aucun effet sur l'alignement vertical d'un élément.

Fusion de marges

La fusion de marges est une particularité du modèle de boîte qui concerne les marges externes verticales (margin-top et margin-bottom) des éléments de type bloc positionnés dans le flux.

Lorsque deux ou plusieurs éléments sont adjacents, qu'ils soient frères ou imbriqués (Parent-Enfant), leurs marges verticales se combinent pour n'en former qu'une seule : la plus grande des deux.

- Un élément *Ancêtre* est un élément qui contient un élément ou une hiérarchie d'éléments.
- Un bloc *Parent* est un élément contenant directement un autre bloc.

  Par exemple, un div contenant un paragraphe p. Attention : si ce paragraphe contient lui-même des éléments (ex: strong), div ne sera pas Parent de l'élément strong mais uniquement son Ancêtre. Le Parent est donc l'Ancêtre immédiat.
- Un bloc contenu directement dans un autre bloc est dit *Enfant* de cet élément. Par exemple, les éléments li sont enfants de leur conteneur ul.

- Les éléments ayant le même élément Parent sont appelés *Frères*.

Le concept n'étant pas simple à imaginer, voyons immédiatement un exemple.

```
<p>La marge inférieure de ce paragraphe fusionne...</p>
<p>... avec la marge supérieure de celui-ci</p>
```

```
p
{
    width:400px;
    background-color:PaleGreen;
    margin-bottom:20px;
}
p + p
{
    width:400px;
    background-color:yellow;
    margin-top:30px;
}
```

Dans notre exemple, nous définissons une marge basse de 20 pixels aux paragraphes et une marge haute de 30 pixels au second. Nous nous attendons par conséquent à un écart de 50 pixels entre ces deux éléments. Or il n'en est rien : en vertu de la fusion de marges, seule la marge la plus élevée, c'est-à-dire 30 pixels, sera retenue :

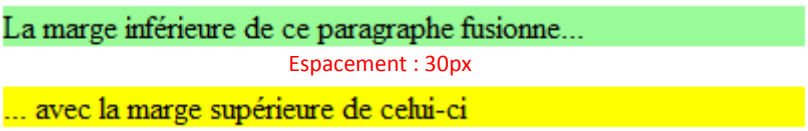


Figure 4 : Fusion de marges entre frères

Plus déroutant encore, ce mécanisme intervient également dans le cas de boîtes imbriquées, c'est-à-dire entre un parent et son premier ou son dernier enfant, par exemple un paragraphe p au sein d'un div.

Le principe demeure identique, à l'exception notable que les marges concernées sont :

- la marge supérieure du parent et la marge supérieure du premier enfant ;
- la marge inférieure du parent et la marge inférieure du dernier enfant.

Parmi ces combinaisons, la marge externe la plus grande est retenue et s'applique au-dessus (ou en dessous) du parent.

Observez l'exemple qui suit :

```
<div>
</div>

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

```
div
{
    margin-top:0px;
    width:400px;
    height:50px;
    background-color:PaleGreen;
}
p
{
    width:300px;
    background-color:yellow;
    margin-top:30px;
}
```

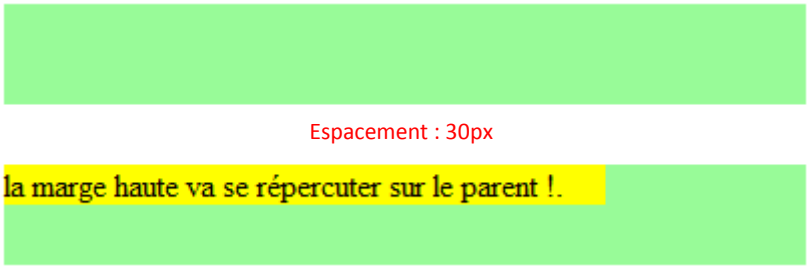


Figure 5 : Fusion de marges entre un parent et son enfant

Cette situation illustre un comportement courant, dû à la fusion de marges entre un parent et son enfant : la marge supérieure de div (non renseignée donc nulle) fusionne avec la marge supérieure de p (30 pixels). Au final, la marge externe supérieure du bloc div prend pour valeur… 30px et tout le bloc est décalé vers le bas.

Vous seriez alors sans doute tentés d'appliquer une règle :

```
div { margin-top:40px; }
```

Mais celle-ci n'aura aucun effet tant que la valeur sera inférieure à celle du paragraphe.

Le mécanisme de fusion de marges a été instauré pour harmoniser le rythme vertical, par exemple pour mieux répartir une suite de paragraphes ou de liens, de façon à ce que l'espace entre chaque élément

demeure identique. Dans certains cas, il est souhaitable de s'affranchir de ce phénomène, qui peut perturber l'affichage d'un site web.

Dans les cas suivants, la fusion des marges ne s'applique pas entre un parent et son enfant :

- Lorsqu'une bordure est appliquée sur le parent. Une simple bordure haute ou basse suffit et il est possible de rendre cette bordure invisible avec une déclaration de type border-top: 1px solid transparent, qui est reconnue depuis IE7. Attention, toutefois, à ne pas oublier que la largeur de la bordure compte dans le calcul des dimensions de la boîte.
- Lorsqu'une marge interne (padding) haute ou basse est appliquée au parent. Là aussi, un simple padding-top: 1px fonctionne, mais doit être pris en compte dans la hauteur du parent. Cette astuce fonctionne sur tous les navigateurs.
- Lorsqu'un contenu orphelin (non balisé) est introduit avant le premier enfant : n'importe quel texte brut, ou caractère, empêche alors la fusion de marges.
- …

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 les autres éléments 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.

Contrairement à ce que l'on peut penser, un système aussi rudimentaire que le positionnement en flux permet d'aller très loin dans la mise en page d'un document.

- Il offre une souplesse et une fluidité inégalées : puisque chaque élément est directement dépendant de ses voisins, l'ensemble de la page se réorganise automatiquement lorsqu'un nouvel élément s'ajoute, lorsqu'un élément est retiré, ou lorsque la longueur (ou la taille) des contenus des blocs varie.
- Grâce à une bonne connaissance des composantes du modèle de boîte, on peut faire interagir chaque élément avec son entourage ou son contenu en jouant sur ses valeurs de marges internes, de marges externes, de largeur de contenu, ou de taille de bordures. On peut ainsi

dimensionner un bloc et le "positionner" à 100px en haut et à 300px à gauche de son parent, simplement en lui affectant des marges externes (ou des marges internes au parent).

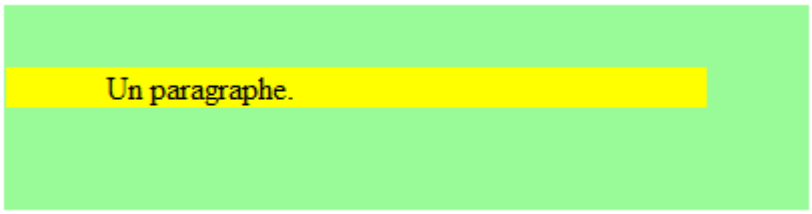


Figure 6 : Un bloc positionné en flux grâce à ses composantes de boîte

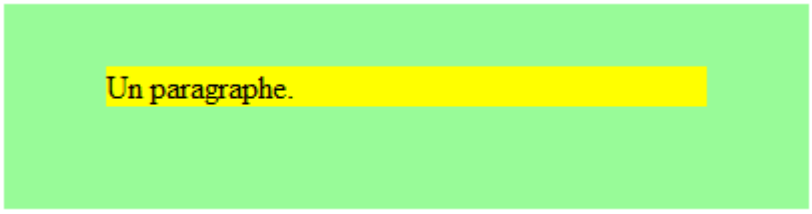


Figure 7 : Un autre bloc positionné en flux grâce à ses composantes de boîte

- …

Dans la suite, nous allons nous intéresser à de nombreux schémas de positionnement. Certains s'écarteront volontairement du modèle de flux courant, ils sont appelés "positionnement hors flux". Retenez que dans la plupart des cas, il demeurera judicieux de rester dans le flux pour éviter tout écueil.

Positionnement absolu

Souvent employé pour l'agencement des blocs, mais rarement à bon escient ou de façon maîtrisée, le positionnement absolu représente à la fois un extraordinaire moyen d'étendre les limites imposées par le flux courant et une hantise constante du concepteur web ne sachant pas toujours où vont se placer les éléments. Et pourtant, le mécanisme décrivant ce schéma de positionnement est loin d'être une loterie aléatoire ; il est au contraire parfaitement défini et, comble de chance, entièrement compatible avec l'ensemble des navigateurs toutes générations confondues.

Un élément est positionné en absolu, vous l'auriez sans doute deviné, lorsqu'il est affublé de la déclaration position:absolute.

Sortir du flux

Cette évidence étant énoncée, ajoutons aux particularités de ce positionnement qu'il retire l'élément concerné du flux selon ce schéma.

- L'élément se retrouve sur un autre plan, placé au-dessus du niveau du flux.

- 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.

```
<p>Paragraphe 1</p>
<p id = "p2">Paragraphe 2</p>
<p>Paragraphe 3</p>
```

```
p
{
    background-color:yellow;
    width:300px;
    border:1px solid black;
    margin-bottom:5px;
    margin-top:5px;
}
#p2
{
    background-color:Pink;
}
```

Figure 8 : Trois paragraphes en flux

```
p
{
    background-color:yellow;
    width:300px;
    border:1px solid black;
    margin-bottom:5px;
    margin-top:5px;
}
#p2
{
    background-color:Pink;
    position:absolute;
    top:10px;
    left:30px;
}
```

Figure 9 : Le paragraphe 2 est retiré du flux

On dit des éléments héritant de la propriété position qu'ils sont "positionnés". Aux éléments positionnés peuvent s'appliquer les quatre propriétés top, right, bottom et left. Il est inutile d'essayer de les affecter à un élément en flux, elles n'auront aucune conséquence.

En revanche, elles sont idéales pour placer l'élément (ou plus précisément son coin supérieur gauche) par rapport aux bords de son bloc référent. Voici quelques exemples :

```
#info
{
    position:absolute;

    /* le coin supérieur gauche du bloc #info se place
    dans le coin supérieur gauche de son référent */
    top:0; left:0;

    /* le coin supérieur droit du bloc #info
    se place dans le coin supérieur droit du référent */
    top:0; right:0;

    /* dans le coin inférieur gauche du référent */
    left:0; bottom:0;

    /* à 20 pixels du coin supérieur gauche du référent */
    top:20px; left:20px;

    /* à 20 % du coin supérieur droit du référent */
    top:10%; right:10%;

    /* le coin supérieur gauche du bloc #info
    se place au centre de son référent */
    top:50%; left:50%;
}
```

Nous avons à présent bien avancé sur le concept du positionnement absolu, mais il nous manque une donnée essentielle : qui est donc ce fameux "référent" mentionné partout ? Plus prosaïquement, par rapport à quoi se positionne un élément en absolu ?

C'est là que beaucoup de concepteurs web se fourvoient : souvent, la réponse qui m'est apportée est "par rapport aux coins de l'écran" ou "selon le document web". Même si ce cas de figure peut survenir, il ne s'agit pourtant que d'une exception dans la règle.

L'étiquette "élément positionné" prend toute sa signification lorsqu'on énonce la formule magique décrivant ce mécanisme de positionnement : *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 avec une valeur autre que static ou s'il ne dispose pas de la propriété position. 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, si l'élément positionné absolument ne rencontre aucun ancêtre positionné, son référent est l'élément racine html.

Nous comprenons, grâce à cette règle, 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.

```
<div>
    <p>Paragraphe 1</p>
    <p id = "p2">Paragraphe 2</p>
    <p>Paragraphe 3</p>
</div>
```

```
div
{
    background-color:PaleGreen;
    width:500px;
    padding-bottom:10px;
    padding-top:10px;

    position:relative;
}
p
{
    background-color:yellow;
    width:300px;
    border:1px solid black;
    margin-bottom:5px;
    margin-top:5px;
}
#p2
{
    background-color:Pink;
    position:absolute;
    top:0px;
    right:0px;
}
```

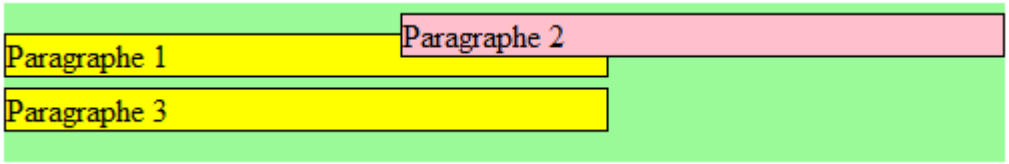


Figure 10 : Positionnement absolu avec une référence

Un mode de rendu particulier

Le positionnement absolu ne répond pas seulement à un mode de placement caractéristique, mais se double également de certaines spécificités de rendu :

- Sa boîte devient dimensionnable quel que soit son type de rendu initial. Cela signifie qu'un élément HTML de rendu inline par défaut peut bénéficier des propriétés width, height, etc. qui auraient été inopérantes dans le flux.
- L'élément occupe par défaut exactement la largeur de son contenu, c'est-à-dire qu'un bloc qui ne contient qu'un seul mot n'occupera que la surface de ce mot, tandis qu'un élément contenant de longs paragraphes s'étendra sur toute la largeur du référent.
- Les marges externes des éléments positionnés en absolu ou de leurs enfants ne sont pas sujettes au phénomène de fusion de marges évoqué précédemment.

La profondeur : z-index

Tous les éléments de la page se placent selon une composante horizontale (axe x), verticale (axe y) et de profondeur (axe z).

La propriété z-indexa pour valeur un nombre entier qui représente l'ordre d'empilement de l'élément. Plus cet entier est grand, plus la couche sera haute dans la pile. La valeur 0 de z-index représente la couche de base des éléments dans le flux. Une valeur négative est permise afin de passer un élément derrière tous les plans visibles, mais est encore relativement mal interprétée par les navigateurs.

Notez que seuls les éléments "positionnés" peuvent être affectés par cette propriété d'empilement z-index.

```
<div>
    <p>Paragraphe 1</p>
    <p id = "p2">Paragraphe 2</p>
    <p>Paragraphe 3</p>
    <p id = "p4">Paragraphe 4</p>
    <p>Paragraphe 5</p>
    <p>Paragraphe 6</p>
</div>
```

```css
#p2
{
    background-color:Pink;
    position:absolute;
    top:0px;
    right:0px;
}
#p4
{
    background-color:OldLace;
    position:absolute;
    top:15px;
    right:10px;
}
```

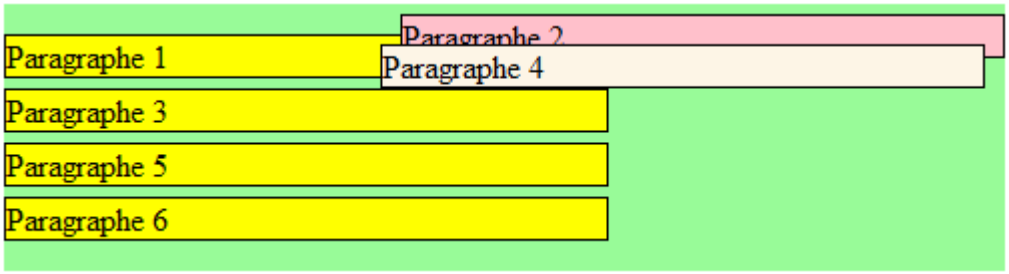


Figure 11 : Sans z-index explicite

```css
#p2
{
    background-color:Pink;
    position:absolute;
    top:0px;
    right:0px;

    z-index: 2;
}
#p4
{
    background-color:OldLace;
    position:absolute;
    top:15px;
    right:10px;

    z-index: 1;
}
```

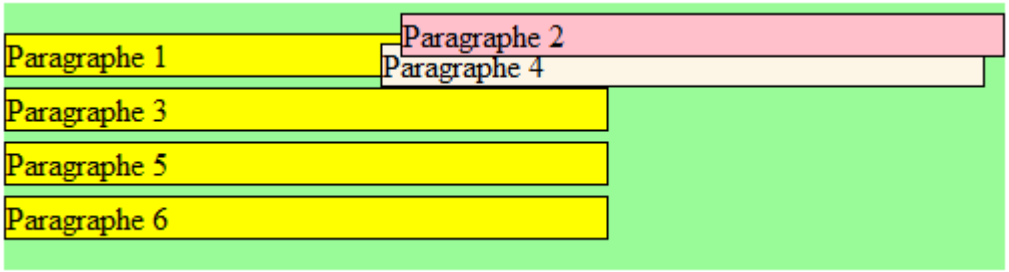


Figure 12 : z-index appliqué à des blocs positionnés

Dans la plupart des cas de figure classiques, la compréhension de l'empilement est intuitive, mais elle l'est moins dès que le contexte est plus complexe. Voici les règles à retenir :

- Sans z-index, les éléments s'empilent suivant leur ordre d'apparition dans le code HTML. Le premier se place sous les suivants.
- Lorsqu'une propriété z-index est appliquée à plusieurs éléments positionnés au sein d'un même référent, celui bénéficiant de la valeur la plus forte apparaît par-dessus les autres.
- Lorsque les éléments positionnés dépendent de plusieurs référents qui eux-mêmes disposent d'une valeur de z-index, c'est le poids du z-index référent qui devient prioritaire.

Étirer un élément

L'une des possibilités originales offertes par les propriétés top, right, bottom et left est de pouvoir cumuler les paires contraires (left et right ou top et bottom) pour littéralement étirer l'élément positionné en absolu au sein de son référent.

Ainsi, un paragraphe positionné doté de la double déclaration left:0px et right: 0px va s'étendre sur toute la largeur de son référent au lieu de n'occuper que la largeur de son contenu.

L'intérêt peut ne pas sembler évident au premier abord, puisqu'une largeur de type width:100% suffit à déployer le bloc sur toute la surface libre. Cependant, rappelez-vous que la propriété width ne représente que la composante de contenu d'une boîte et que les autres unités (bordures, marges internes et externes) sont parfois nécessaires et vont s'ajouter à cette valeur de 100% provoquant un conflit : le bloc peut dépasser de son référent, ou pire, interagir avec d'autres éléments avoisinants.

Dans le cas où l'élément absolu est "dimensionné" à l'aide de ses paires contraires, il devient possible d'occuper 100% de la largeur tout en bénéficiant de padding et de border :

```css
#header
{
    position:absolute;
    left:0;right:0; /* s'étend sur toute la largeur  */
    padding:20px; /* le padding est inclus à l'intérieur  */
    background-color:yellow;
}
```

Cette astuce fonctionne également pour le pair top et bottom dans le cas d'une expansion verticale et elle permet certaines fantaisies, telles que centrer un élément dans la page sans lui imposer de dimensions :

```css
#popup
{
    position:absolute;
    top:25%; bottom:25%; /* top et bottom  */
    left:25%; right:25%; /* left et right  */
    padding:10px;
    background-color:yellow;
}
```

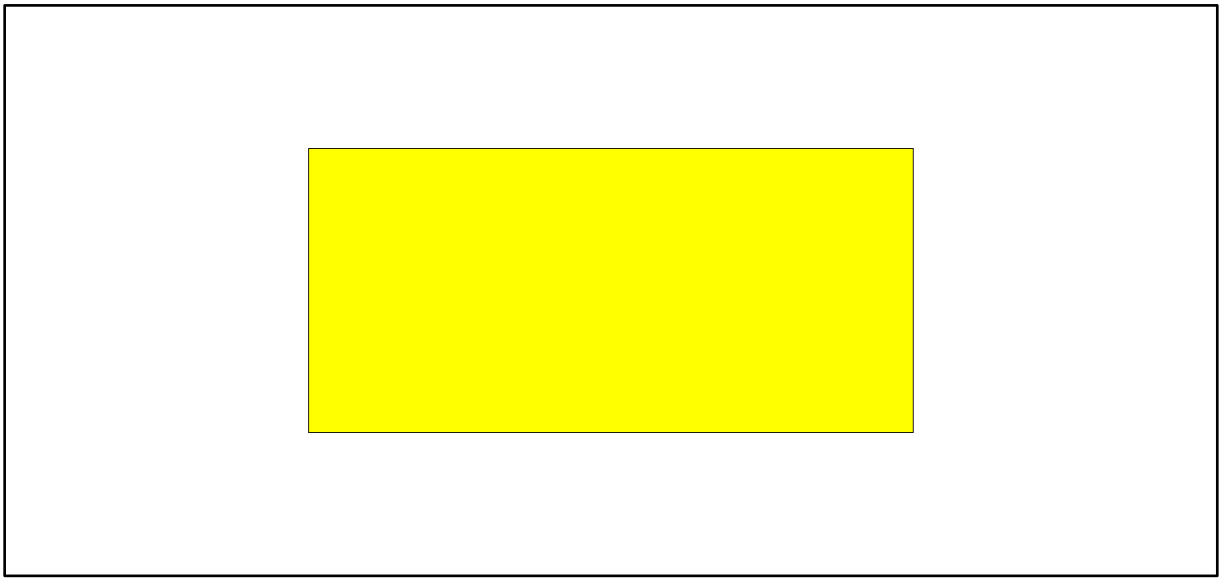


Figure 13 : Un bloc div centré horizontalement et verticalement par rapprot à la page

Un dernier détail a son importance : cette technique fonctionnelle sur tous les navigateurs récents n'est comprise qu'à partir d'Internet Explorer 7.

Positionnement fixé

Autant le positionnement absolu est célèbre, autant il est bien plus rare d'entendre parler du positionnement fixé. Pourtant les similitudes entre les deux schémas sont nombreuses :

- Un élément est dit fixé lorsqu'il bénéficie de la déclaration position:fixed.
- Puisqu'on lui applique une propriété position, l'élément est dit "positionné" (il sert donc de référent aux éléments positionnés en absolu).
- Un bloc fixé sort du flux et se place dans un autre plan, laissant les éléments en flux se réorganiser entre eux.
- La boîte d'un élément fixé devient dimensionnable, à l'instar des éléments absolus.

Le positionnement fixé se distingue principalement de son cousin absolu de par sa particularité de rendu : un élément fixé demeure ancré à l'écran même lorsque l'utilisateur fait défiler le contenu à l'aide des ascenseurs (scrollbars).

En termes de localisation, et lorsque les propriétés top, right, bottom et left sont précisées, l'élément fixé est toujours positionné par rapport à la partie visible de la fenêtre du navigateur, quel que soit son ancêtre, fût-il positionné.

Ce type de positionnement se révèle particulièrement séduisant dans la conception d'un menu de navigation qui demeurera figé à l'écran même si le contenu de la page défile :

```
#nav
{
    position:fixed;
    top:0;
    right:0;
    width:200px;
    background:yellow;
}
```

Quelques réserves sont à prendre en compte avant d'employer ce schéma de positionnement : tout d'abord, il n'est reconnu qu'à partir d'Internet Explorer 7. Par ailleurs, il peut dans certains cas poser de lourds problèmes d'ergonomie : pensez qu'un élément positionné à 800 pixels du haut de l'écran devient totalement invisible et inaccessible sur de petits écrans de notebooks par exemple, dont la résolution est de 600 pixels de hauteur. Ce type de positionnement est d'ailleurs inactivé sur certains smartphones, notamment ceux sous environnement Safari Mobile (iPhone), Opera Mini ou Android.

Positionnement relatif

Troisième et petit frère de la famille, le positionnement relatif porte à mon sens très mal son nom, tout simplement parce qu'il ne représente en aucune manière un positionnement tel qu'il a été défini jusque-là, mais un simple décalage.

Un élément positionné relativement se place par rapport à sa position classique dans le flux, puis est éventuellement décalé si au moins une des propriétés top, right, bottom ou left est renseignée. La notion de "relatif" s'applique par conséquent à son emplacement initial dans le flux.

Contrairement aux schémas absolus et fixés, les propriétés top, right, bottom et left ne servent plus à indiquer un emplacement, mais un décalage par rapport à la position initiale.

```
<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non
    risus. Suspendisse lectus tortor, <span>dignissim</span> sit
    amet, adipiscing nec, ultricies sed, dolor. Cras elementum
    ultrices diam.
</p>
```

```
p
{
    width:200px;
    padding:10px;
    margin:5px;
    border:5px solid black;
    background-color:yellow;
}
span
{
    background-color:Plum;
}
```

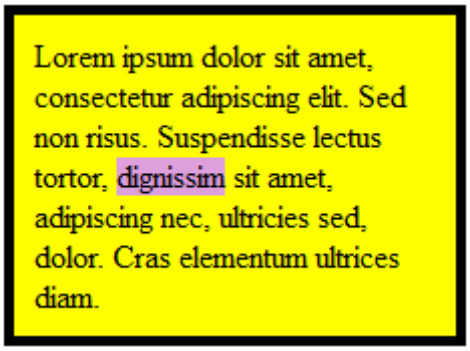


Figure 14 : dignissim en flux classique

```
span
{
    background-color: Plum;

    position:relative;
    left:4px;
    top:5px;
}
```

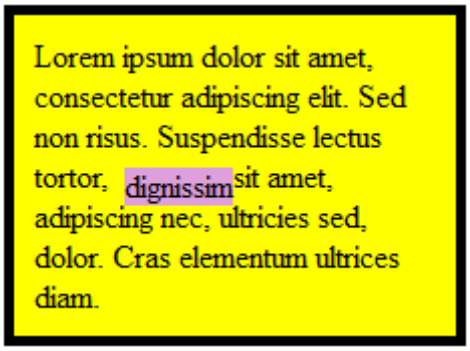


Figure 15 : dignissim enpositionnement relatif

Un élément relatif demeure dans le flux et continue à interagir avec les autres éléments voisins.

D'ailleurs, l'espace laissé par un élément décalé en relatif ne peut pas être occupé par d'autres éléments, car il est toujours censé l'occuper.

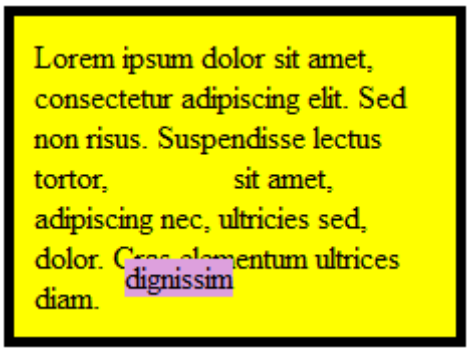


Figure 16 : dignissim en positionnement relatif (left : 4 et top : 50)

Un élément positionné relativement conserve une particularité due à sa propriété CSS : il est dit "positionné". Le seul fait d'être positionné en relatif, sans adjonction de valeurs pour top, right, bottom et left en fait un référent dans le flux, parfait pour les contenus positionnés en absolu. C'est d'ailleurs son usage principal.

Positionnement flottant

Un usage détourné de son objectif initial

Les spécifications initiales ne semblent pas avoir prévu l'usage de cette propriété pour positionner les éléments, comme que nous le faisons actuellement (même si rien n'indique que cela soit interdit non plus) : tous les exemples illustrant le flottement désignent des images ou des portions de texte à pousser à droite ou à gauche dans un élément.

Un positionnement à part

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.

Cela se corse légèrement par la suite, 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 précédente.

```
<div>
    <p>
        Paragraphe en flux
    </p>
    <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
        non risus. Suspendisse lectus tortor, dignissim sit amet,
        adipiscing nec, ultricies sed, dolor. Cras elementum
        ultrices diam.
    </p>
</div>
```

```css
div
{
    border:1px solid black;
    background-color:PaleGreen;
    width:500px;
    padding:5px;
}
p
{
    border:1px solid black;
    background-color:yellow;
    margin:5px;
}
p:first-child
{
    background-color:Plum;
}
```

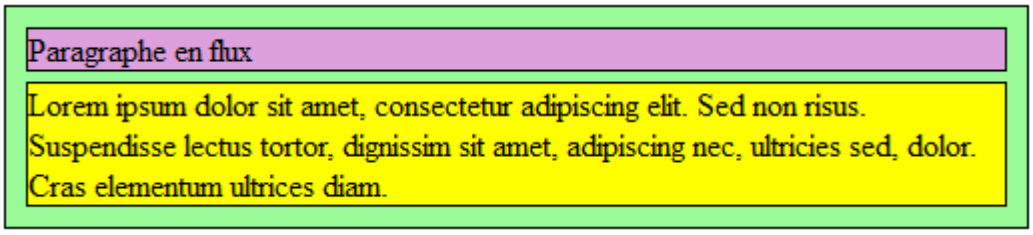


Figure 17 : Avant flottement (deux paragraphes en flux)

```css
div
{
    border:1px solid black;
    background-color:PaleGreen;
    width:500px;
    padding:5px;
}
p
{
    border:1px solid black;
    background-color:yellow;
    margin:5px;
}
p:first-child
{
    background-color:Plum;
    float:left;
    width:150px;
}
```

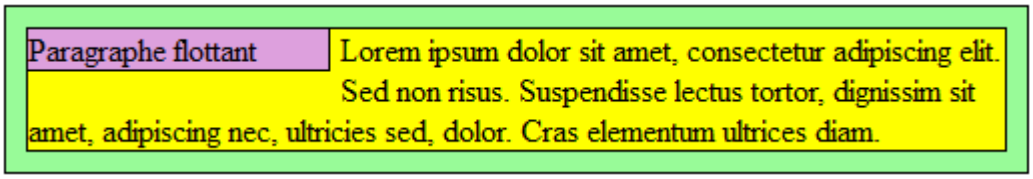


Figure 18 : Après flottement (premier paragraphe flottant à gauche)

Des blocs côte à côte

À 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.

```
<div>
    <p>
        Paragraphe flottant à gauche + width: 150px.
    </p>
    <p>
        Un autre paragraphe flottant à gauche + width: 150px. Lorem
        ipsum dolor sit amet, consectetur adipiscing elit. Sed non
        risus. Suspendisse lectus tortor, dignissim sit amet,
        adipiscing nec, ultricies sed, dolor. Cras elementum
        ultrices diam.
    </p>
</div>
```

```
div
{
    border:1px solid black;
    background-color:PaleGreen;
    width:500px;
    padding:5px;
}
p
{
    border:1px solid black;
    background-color:yellow;
    margin:0px;
    float:left;
    width:150px;
}
```

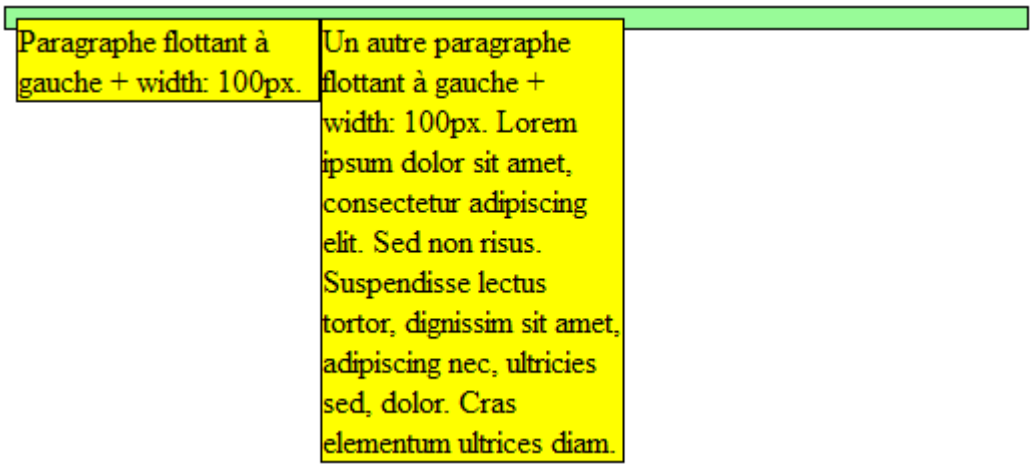


Figure 19 : Deux blocs côte à côte (float: left)

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. Il s'agit d'une méthode courante de mise en forme de blocs côte à côte tel qu'un menu de navigation et la zone de contenu.

Attention, toutefois, au caractère hâtif de cette disposition adjacente "magique" qui s'écarte de l'application originale de la propriété float : 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, puisque seuls des contenus en flux sont susceptibles de meubler l'espace dans le plan occupé par le flux courant. Cela est simple à démontrer en appliquant une couleur de fond au conteneur.

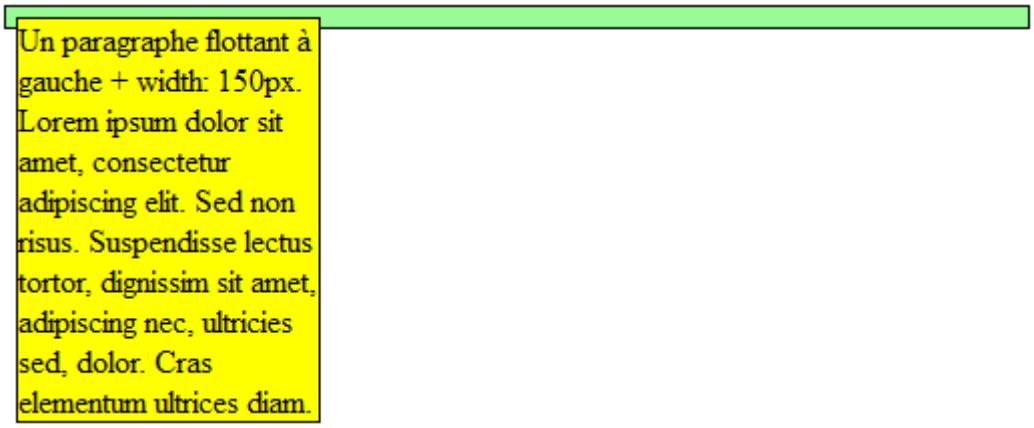


Figure 20 : Parent vide

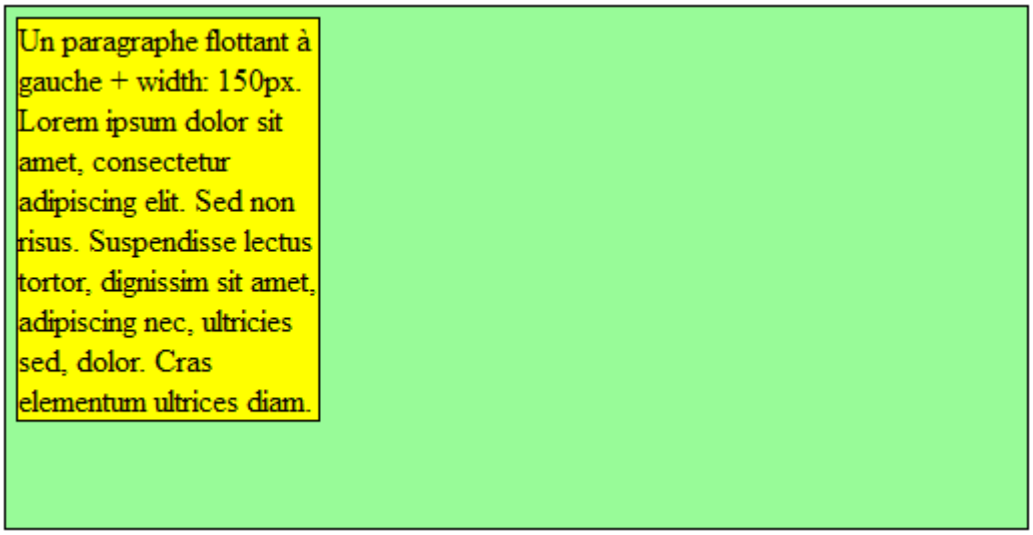


Figure 21 : Ajout de hauteur pour "voir" le parent

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 nettoyage des flottants exclusivement à gauche (clear:left), à droite (clear:right) ou les deux simultanément (clear:both).

```
<div>
    <p>
        Paragraphe flottant à gauche + width: 150px.
    </p>
    <p>
        Un paragraphe en flux. Lorem ipsum dolor sit amet,
        consectetur adipiscing elit. Sed non risus. Suspendisse
        lectus tortor, dignissim sit amet, adipiscing nec, ultricies
        sed, dolor. Cras elementum ultrices diam.
    </p>
</div>
```

```css
div
{
    border:1px solid black;
    background-color:PaleGreen;
    width:500px;
    padding:5px;
}
p
{
    border:1px solid black;
    background-color:yellow;
    margin:5px;
}
p:first-child
{
    background-color:Plum;
    float:left;
    width:150px;
}
p:last-child
{
    clear:left;
}
```

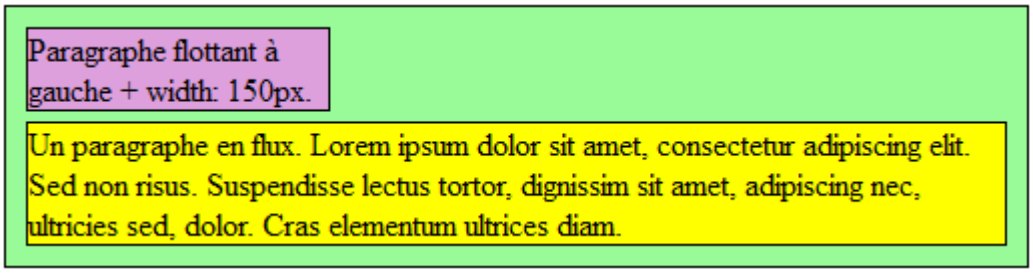


Figure 22 : Clear:left sur le second élément

Cette fonctionnalité offre un certain nombre d'avantages, puisqu'elle autorise un objet en flux à interagir avec les flottants précédents. Cela permet, par exemple, d'empêcher des "dépassements" de flottants de leur parent, ou de positionner un élément toujours en bas du bloc flottant le plus long comme le montre l'exemple suivant :

```html
<div id="bloc1">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
</div>
<div id="bloc2">
    Proin porttitor, orci nec nonummy molestie, enim est
    eleifend mi, non fermentum diam nisl sit amet erat.
    Duis semper. Duis arcu massa, scelerisque vitae,
    consequat in, pretium a, enim. ...
</div>
<div id="bloc3">
    Ut velit mauris, egestas sed, gravida nec, ornare ut, mi.
</div>
<div id="footer">
    Pied de page placé sous le bloc le plus long
</div>
```

```css
div
{
    border:1px solid black;
    background-color:PaleGreen;
    width:600px;
    padding:5px;
}
#bloc1, #bloc2, #bloc3
{
    float:left;
    width:150px;
    margin:0 10px 0 0;
    background-color:yellow;
}
#footer
{
    clear:both;
    background-color:Plum;
    width:590px;
}
```

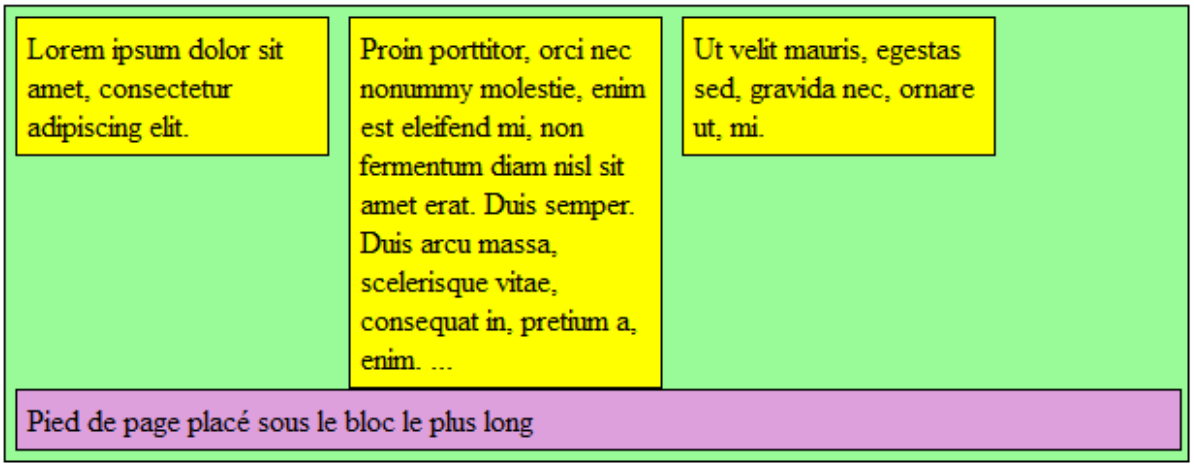


Figure 23 : Une conception en trois colonnes grâce à clear