

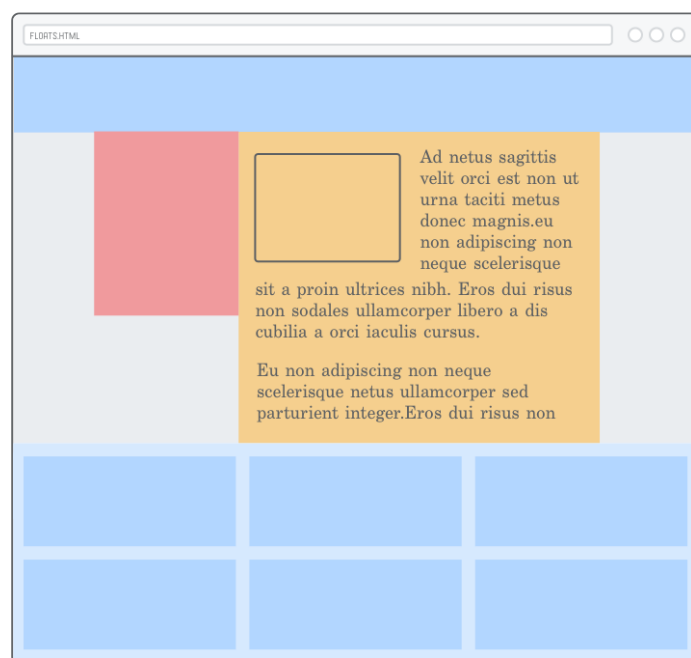
# Float et Mise en forme

"Float" vous permet de mettre des éléments de niveau bloc côte à côte au lieu d'au-dessus de l'autre. Il nous permet de construire toutes sortes de mises en page, y compris les barres latérales, pages multi-colonnes, des grilles et des articles de type magazine avec le texte qui s'écoule autour d'une image.

Les mises en page à base de *float* tendent à être remplacées par *Flexbox* dans les sites web modernes. Depuis plus d'une décennie, les *float* ont servi de base pour la majorité des sites web ce qui signifie que vous aurez certainement les rencontrer à un moment donné dans votre carrière.

Peut-être plus important encore, le caractère limité des *float* en fait une introduction plus douce aux mises en page CSS que Flexbox. Au lieu d'être submergé par toutes les possibilités de Flexbox, nous aurons l'occasion de nous concentrer davantage sur le *processus* de création d'une mise en page sophistiquée de page web.

Ce chapitre illustre les float CSS avec un exemple de projet assez simple.



Tout d'abord, créez un nouveau dossier appelé **floats**, puis ajoutez une nouvelle page web appelée `floats.html` avec le balisage suivant:

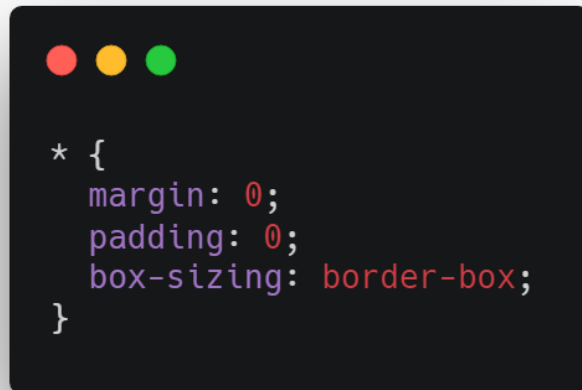
```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Floats</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <section class="page">
    <nav class="menu">Menu</nav>
    <aside class="sidebar">Sidebar</aside>
    <article class="content">Content</article>
    <footer class="footer">Footer</footer>
  </section>
</body>
</html>
```

Cela nous donne la structure de base de la plupart des sites web.

Nous avons un menu de navigation, une barre latérale, le contenu principal de la page et un pied de page.

Vous ne verrez pas grand chose quand vous ouvrez *floats.html* dans un navigateur parce que les éléments vides sont de taille “zéro”.

Nous allons corriger cela dans la section suivante.



```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}
```

Assurez-vous également de créer une feuille de style nommée *styles.css* qui réinitialise le comportement du modèle de boîte - *box-model* - par défaut, comme indiqué ci-dessus.

## Mise en page html par défaut

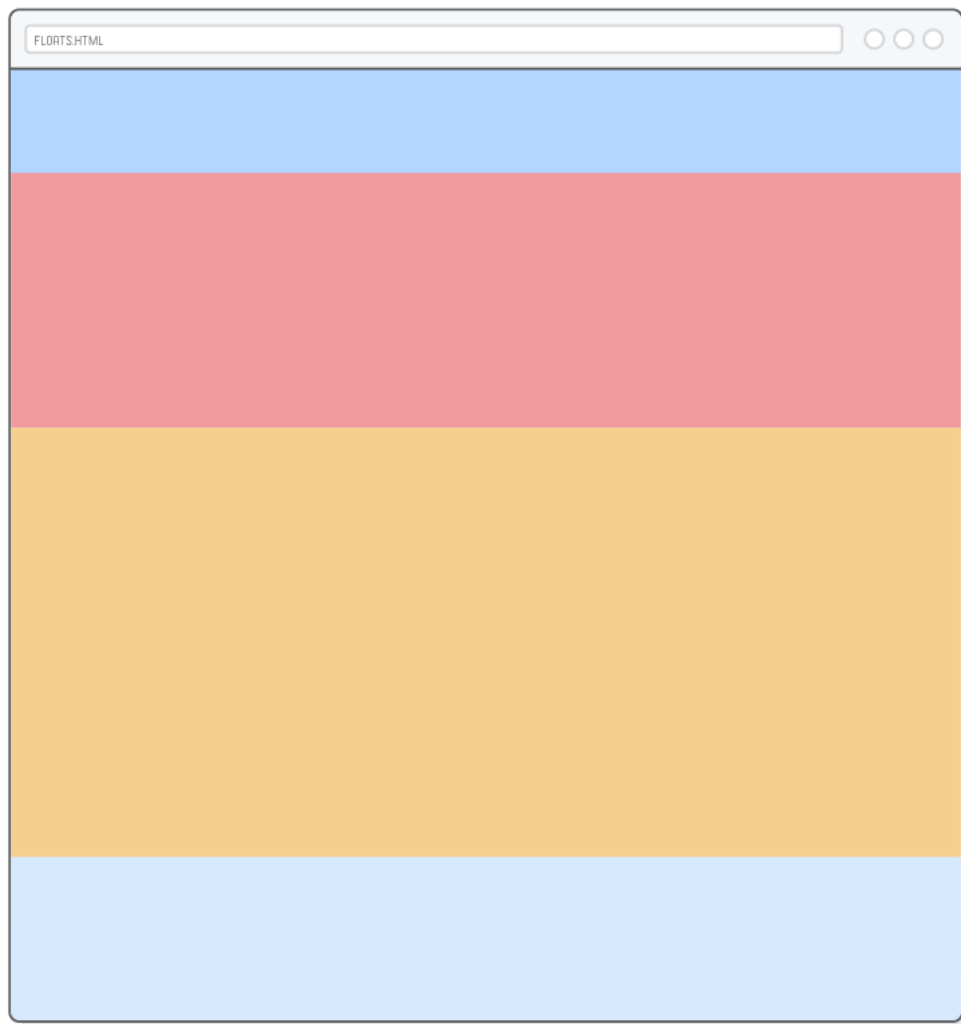
Les *float* modifient la disposition par défaut d'une page Web, nous devrions probablement commencer par examiner ce qu'est exactement ce comportement "par défaut".

Nous pouvons obtenir un meilleur rendu pour notre page d'exemple en ajoutant des couleurs de fond et des hauteurs explicites à chacun de nos éléments `<div>`. Ajoutez ceci à *styles.css*:



```
.menu {  
  height: 100px;  
  background-color: #B2D6FF;  
  /* Medium blue */  
}  
  
.sidebar {  
  height: 300px;  
  background-color: #F09A9D;  
  /* Red */  
}  
  
.content {  
  height: 500px;  
  background-color: #F5CF8E;  
  /* Yellow */  
}  
  
.footer {  
  height: 200px;  
  background-color: #D6E9FE;  
  /* Light blue */  
}
```

Cela nous donne un joli arc-en-ciel, ce qui n'est pas ce que nous recherchons, bien que cela montre quelques concepts utiles.



L'élément important ici est que chaque élément de niveau bloc remplit 100% de la largeur de ses éléments parents (`<div class="page">` dans ce cas) et ils apparaissent verticalement l'un après l'autre. Encore une fois, nous sommes essentiellement limités à une mise en page à une seule colonne.

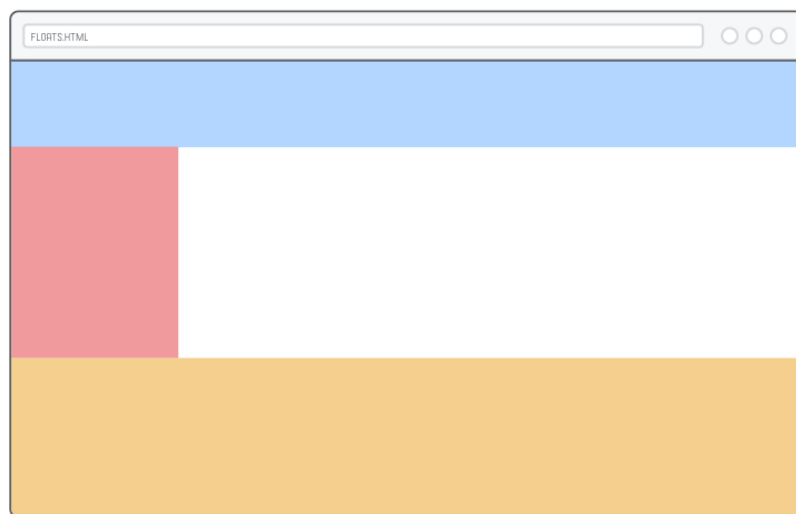
En règle générale, vous souhaitez laisser la hauteur de ces cases être déterminée automatiquement en fonction du contenu qu'ils contiennent. Cependant, nous sommes plus concernés par le contrôle des mises en page donc nous ne traiterons pas beaucoup de contenu réel. C'est pourquoi nous avons besoin des propriétés de hauteur explicites.

Il vaut la peine de jeter un coup d'oeil à ce qui se passe lorsque nous réduisons la largeur d'un élément. Mettez à jour notre règle `.sidebar` pour qu'elle corresponde à ce qui suit:

```
.sidebar {  
  width: 200px;  
  height: 300px;  
  background-color: #F09A9D;  
}
```

L'élément de barre latérale devient plus étroit, mais le reste des cases restent dans la même position. Tous les blocs sont toujours positionnés verticalement l'un après l'autre.


C'est le comportement que nous allons changer avec les *float*.



## Faire flotter un élément

La propriété *float* nous permet de contrôler la position horizontale d'un élément. En faisant «flotter» la barre latérale vers la gauche, nous demandons au navigateur de l'aligner sur le côté gauche de la page.

Faites flotter notre barre latérale avec la ligne suivante:



```
.sidebar {  
  float: left;  
  width: 200px;  
  height: 300px;  
  background-color: #F09A9D;  
}
```

Cependant, cela n'aligne pas seulement la barre latérale - il indique également aux éléments environnants qu'ils peuvent circuler autour de la barre latérale au lieu de commencer en dessous. C'est comme si la barre latérale est à l'intérieur du bloc *.content*, donc tout balisage HTML dans *.content* sera enveloppé dans la boîte de la barre latérale.

Cela nous donne une présentation de style magazine:



Vous pouvez également faire flotter les éléments à droite, comme indiqué ci-dessous (gardons notre barre latérale flotter à gauche cependant).

Ou, si vous remplacez une déclaration *float*, vous pouvez l'annuler avec la valeur *none*. Ce sont les valeurs les plus courantes pour la propriété *float*.



Nous disposons maintenant de tous les outils nécessaires pour aligner les éléments au niveau des blocs: *float* pour les alignements gauche / droite et *margin auto* pour l'alignement central.



N'oubliez pas que ceci s'applique uniquement aux éléments bloc. Les éléments en ligne sont alignés avec la propriété *text-align*.



### LEFT ALIGN

FLOAT: LEFT;



### CENTER ALIGN

MARGIN: 0 AUTO;



### RIGHT ALIGN

FLOAT: RIGHT;

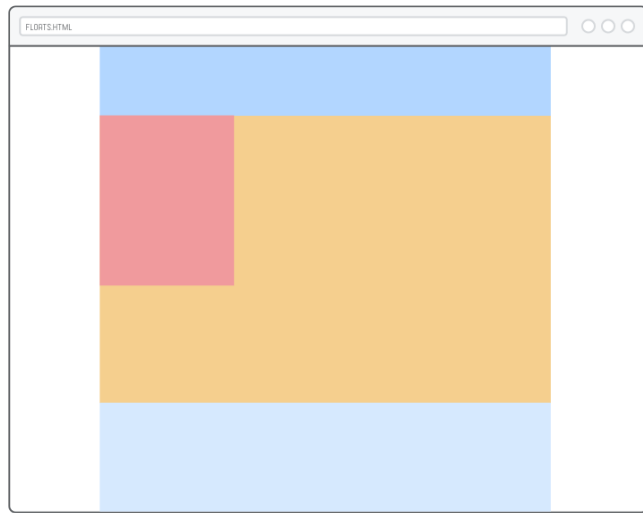
## Flotter à l'intérieur d'un parent

Les boîtes flottantes sont toujours alignées à gauche ou à droite de leur élément parent. Dans notre exemple, le parent de la barre latérale est `<section class="page">`, qui est aussi large que la fenêtre du navigateur. C'est pourquoi notre sidebar flotte à l'extrême gauche de la page.

Changeons cela en donnant à notre page une disposition à largeur fixe. Encore une fois, la technique de centrage automatique des marges est pratique. Ajoutez ceci à *styles.css*:

```
.page {  
  width: 900px;  
  margin: 0 auto;  
}
```

Maintenant, nous pouvons voir que *.sidebar* flotte à gauche du conteneur *.page*, opposé au bord de la fenêtre du navigateur.



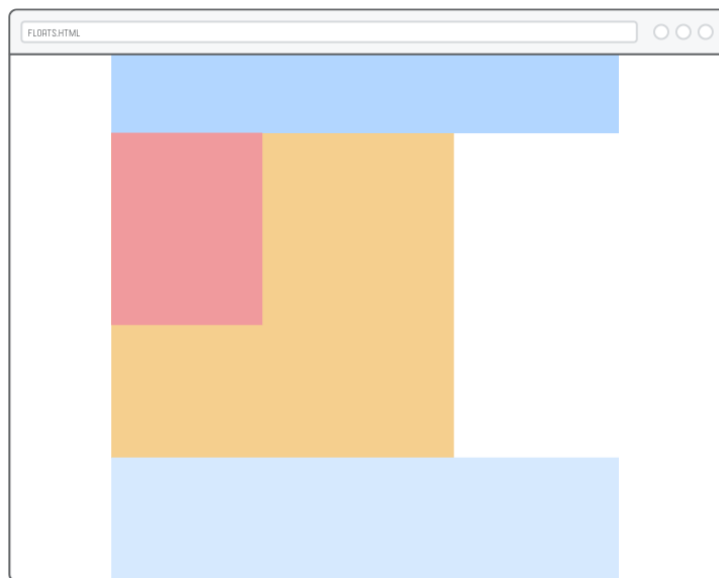
Le positionnement des éléments imbriqués de conteneur comme ceci c'est ainsi que vous construisez des dispositions sophistiquées de site Web. Ici, nous avons commencé avec *.page* pour centrer tout, puis nous avons aligné à gauche une barre latérale à l'intérieur de cette page centrée. Les choses peuvent devenir beaucoup plus complexes, mais notre exemple simple démontre la vérité universelle des mises en page CSS: tout est une boîte à l'intérieur d'une boîte à l'intérieur d'une autre boîte.

## Float multiples

Examinons un peu plus notre float de style magazine actuel en ajoutant une largeur explicite à notre bloc `.content`:

```
.content {  
  width: 650px;  
  /* Nouveau */  
  height: 500px;  
  background-color: #F5CF8E;  
}
```

Cela démontre clairement que notre barre latérale est en fait dans le bloc `.content`: si vous prenez une capture d'écran, vous aurez une image de 650 pixels de large opposée à 850 pixels (notre barre latérale est de 200 pixels de large).

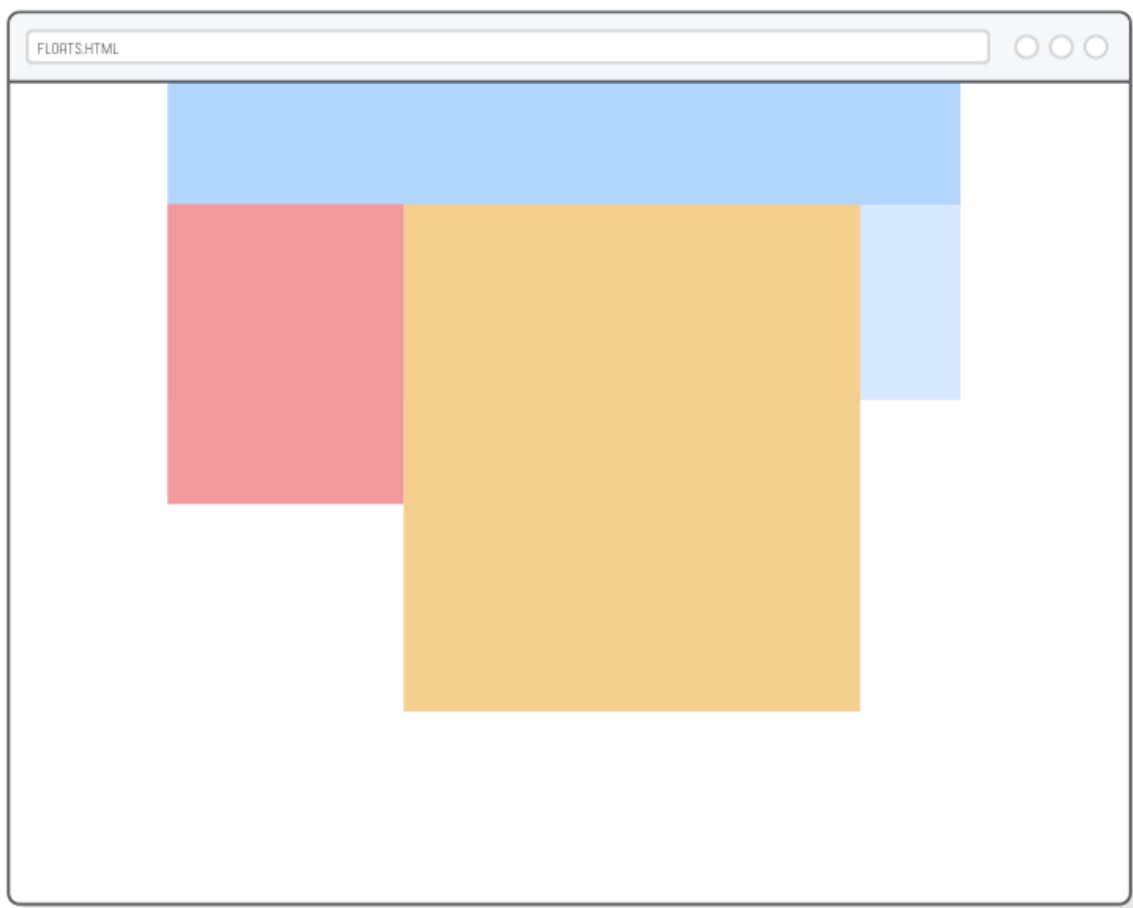


Ce type de comportement flottant est possible pour les images (ce que nous verrons plus tard), mais pour la mise en page, nous voulons que le bloc de contenu soit à côté de la barre latérale au lieu de couler autour de lui. Pour cela, nous devons dire au bloc de contenu de flotter à gauche, aussi. Ajoutez une ligne supplémentaire à la règle `.content`:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains CSS code for the `.content` class.

```
.content {  
  float: left;  
  /* Nouveau */  
  width: 650px;  
  height: 500px;  
  background-color: #F5CF8E;  
}
```

Lorsque vous faites flotter plusieurs éléments dans la même direction, ils s'empilent horizontalement, un peu comme l'algorithme de disposition verticale par défaut, sauf qu'ils sont pivotés de 90 degrés. Le code ci-dessus fait apparaître l'ensemble de notre bloc de contenu sur la droite de la barre latérale au lieu de l'entourer.



Cela nous donne un véritable contrôle sur l'alignement horizontal de nos boîtes de bloc. Essayez de jouer avec les valeurs `float` pour les deux `.sidebar` et `.content`, et vous verrez que nous avons déjà quelques dispositions distinctes à notre disposition:



**SIDEBAR** LEFT  
**CONTENT** LEFT



**SIDEBAR** LEFT  
**CONTENT** RIGHT



**SIDEBAR** RIGHT  
**CONTENT** LEFT

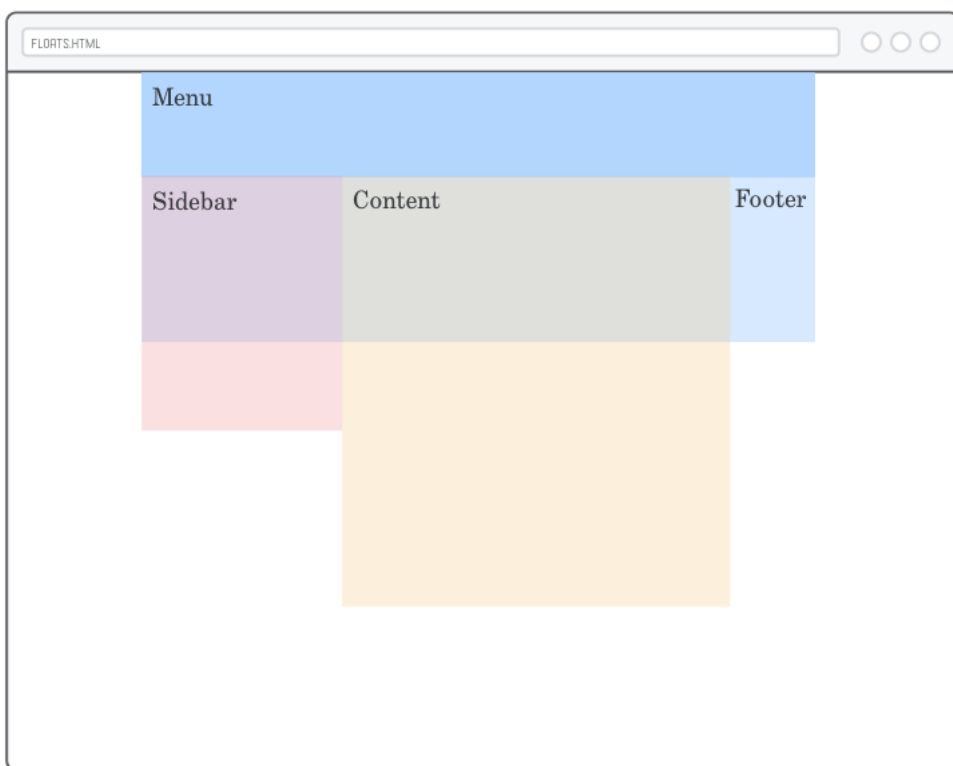


**SIDEBAR** RIGHT  
**CONTENT** RIGHT


Assurez-vous que les deux sont flottant à gauche. Cela fait ce que l'on attend pour notre mise en page pour la barre latérale et le contenu des blocs, mais malheureusement notre élément `.footer` est à la rue...

## Après un float

Vous avez probablement remarqué que notre pied de page apparaît en haut à droite, directement au-dessous de `.menu`. En effet, les boîtes *float* sont retirées de l'écoulement normal de la page. La hauteur de nos éléments flottants ne contribuent pas à la position verticale du pied de page, il se colle tout simplement en dessous du dernier élément qui *n'a pas* été déclaré *float*.



Nous pouvons voir plus clairement en ajoutant une bordure rouge autour de notre élément `.page`:



```
.page {  
  width: 900px;  
  margin: 0 auto;  
  border: 1px solid red; /* Nouveau */  
}
```

Remarquez comment la bordure est seulement autour de la classe `.menu` et des éléments `.footer`. C'est comme si les éléments flottants n'étaient pas encore là. Il y a deux façons de résoudre ce problème: la compensation d'un *float -clear-* ou en cachant le débordement *-overflow-*.

## Compensation *-clear-* de float

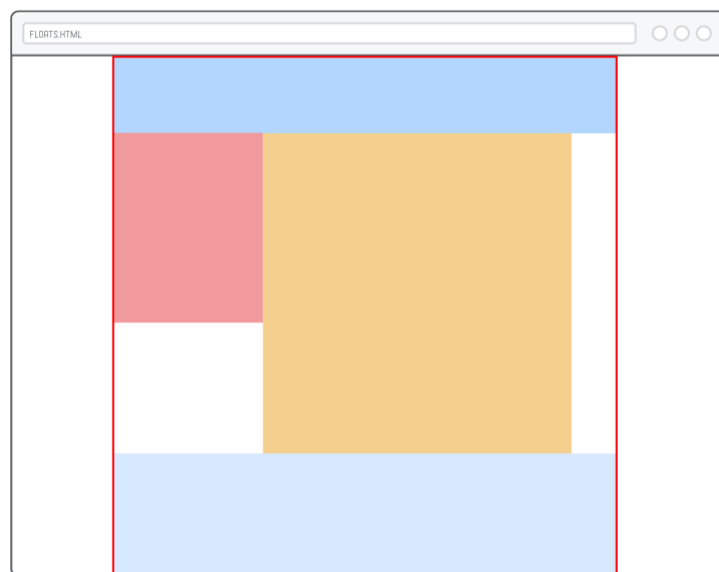
"Compenser" un float est quand nous disons à un bloc d'ignorer les *float* qui se présentent devant lui. Au lieu de couler autour de la boîte flottante, un élément *clear* apparaîtra toujours après des *float*. Il va forcer la boîte à faire un retour dans le flux vertical par défaut de la page.

Nous pouvons utiliser la propriété *clear* pour faire de notre `.footer` menu déroulant au bas de la page:

```
.footer {  
  clear: both;  
  /* Nouveau */  
  height: 200px;  
  background-color: #D6E9FE;  
}
```

Habituellement, vous voulez effacer les *float* à la fois gauche et droite comme nous l'avons fait ici, mais vous pouvez choisir d'effacer seulement l'un ou l'autre avec la valeurs *left* ou *right*.

Notez que la bordure rouge enveloppe maintenant tout le pied de page, ce qui indique que les éléments flottants ont en effet pris en compte toute la hauteur du conteneur *.page*




Selon le type de mise en page que vous essayez de créer, il s'agit d'une solution parfaitement acceptable. Nous pourrions nous arrêter ici, mais nous allons explorer le comportement des floats un peu plus en



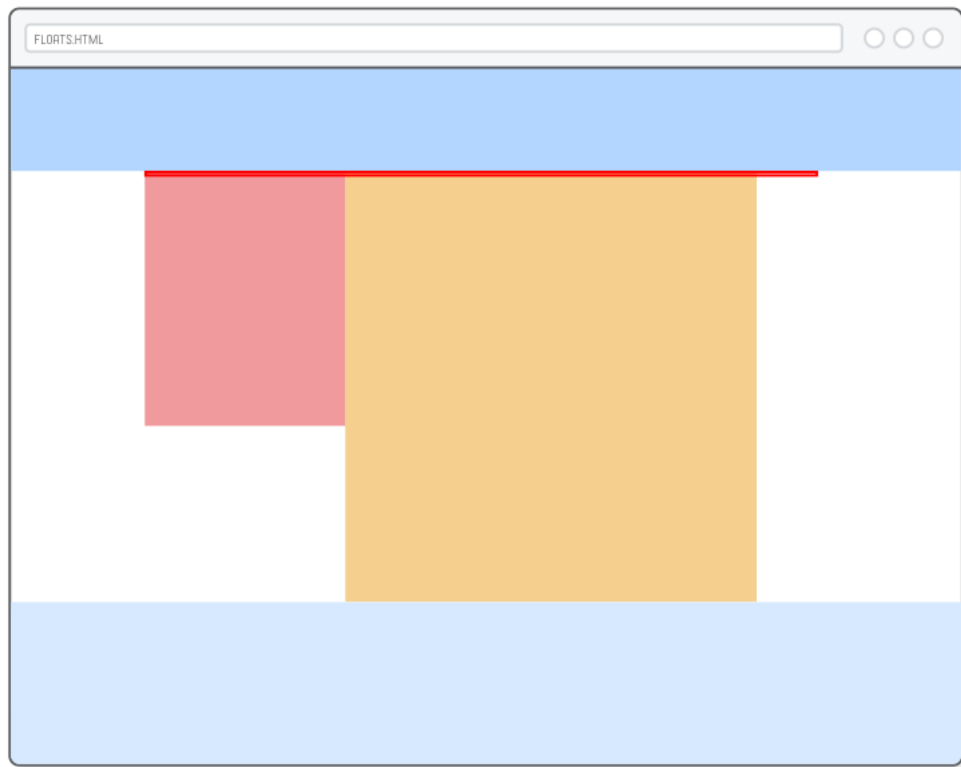
transformant notre page en une mise en page complète avec des couleurs de fond remplissant toute la fenêtre du navigateur.

Regardez ce qui se passe lorsque nous mettons le pied hors de l'élément `.page`. Changer l'élément `<body>` pour correspondre à ce qui suit:



```
<body>
  <nav class="menu">Menu</nav>
  <section class="page">
    <aside class="sidebar">Sidebar</aside>
    <article class="content">Content</article>
  </section>
  <footer class="footer">Footer</footer>
</body>
```

Ce faisant les classes `.menu` et `.footer` sont hors de notre largeur fixe `.page`, ils sont à la pleine largeur de la fenêtre, ce qui est exactement ce que nous voulons pour une mise en page à fond perdu. Cependant, remarquez comment `.page` a encore une fois la hauteur zéro malgré le fait que le pied de page efface toujours la barre latérale et le contenu des blocs.

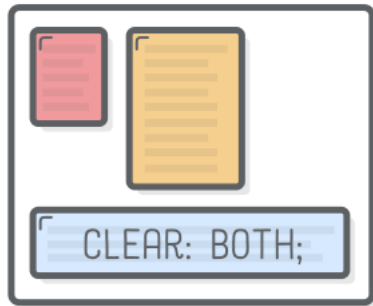


Encore une fois, les seuls éléments `.page` flottent, ils ne comptent pas pour sa hauteur. En d'autres termes, le déplacement du footer à l'extérieur du conteneur `.page` a cassé notre solution de `clear`.

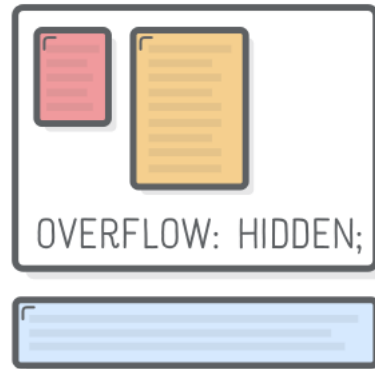
## Cacher les débordements

Le nettoyage des `float` ne règlent que le problème de hauteur lorsqu'il y a un élément à l'intérieur de l'élément conteneur auquel nous pouvons ajouter une propriété `clear`. Maintenant que notre pied de page est en dehors de `.page`, nous avons besoin d'une nouvelle façon de faire des éléments flottants pour contribuer à la hauteur de leur conteneur.

## CLEARING WITH CHILD ELEMENT



## CLEARING WITH PARENT ELEMENT



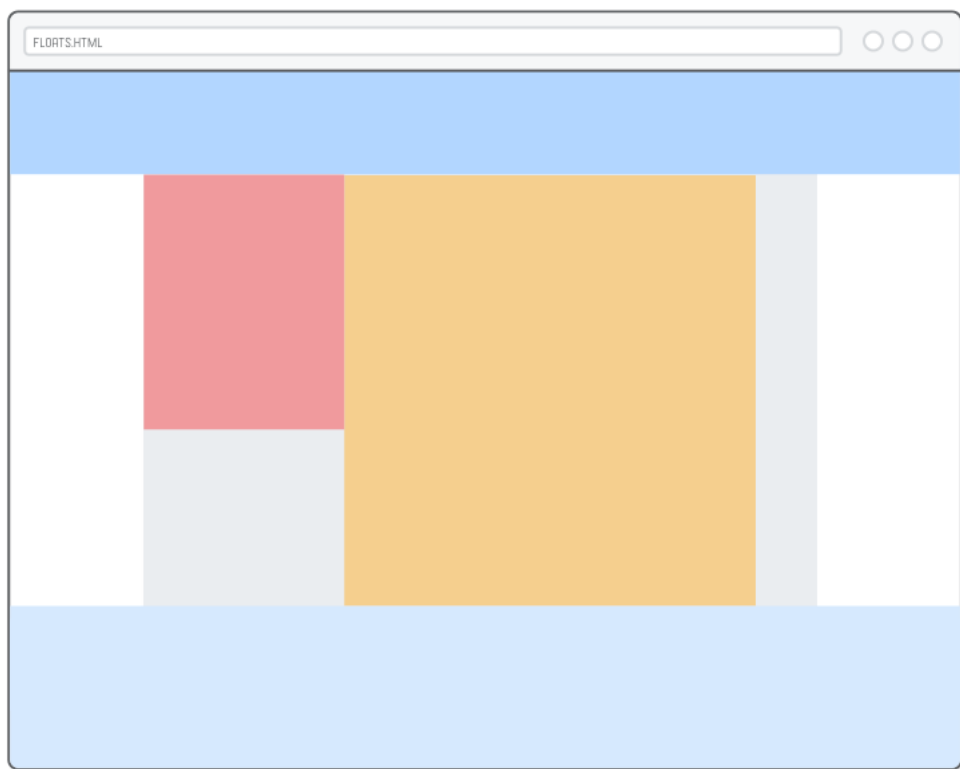
La solution est la propriété CSS *overflow*. En ajoutant une déclaration *overflow: hidden* à une div conteneur, nous lui disons de prendre en compte la hauteur de tous les éléments flottants qu'il contient.

Voilà comment nous pouvons ajouter une couleur de fond à notre élément `.page` et comment il sera réellement rendu:

```
.page {  
  width: 900px;  
  margin: 0 auto;  
  overflow: hidden;  
  /* Nouveau */  
  background-color: #EAEDF0;  
  /* Nouveau */  
}
```

Vous devriez maintenant être en mesure de voir un fond gris clair pour `.page` au lieu du blanc par défaut. Ce n'est pas encore une mise en page complète (nous allons aborder que dans la section suivante).

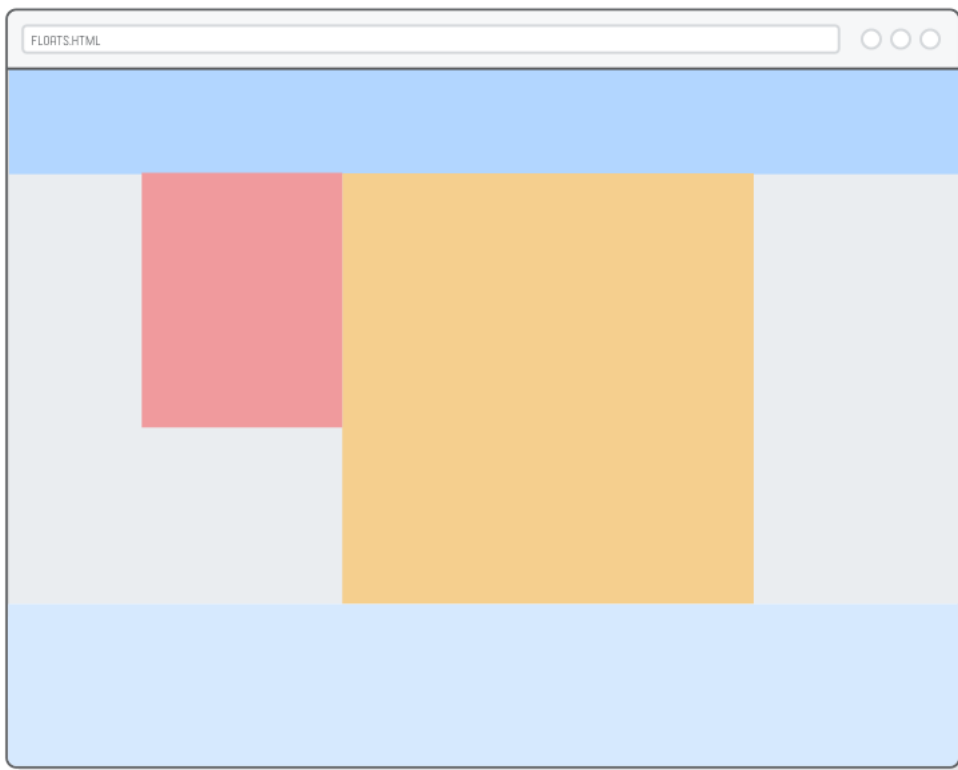
L'important ici est le comportement de `overflow: hidden`. Sans cette propriété, nous ne serions pas en mesure de voir l'arrière - plan de conteneur `.page`, car il aurait une hauteur nulle.



Pour résumer, lorsque vous avez un élément HTML non flottant supplémentaire au bas d'un conteneur `div`, utilisez la solution `clear`. Sinon, ajoutez une déclaration `overflow:hidden` à l'élément conteneur. L'idée sous-jacente pour les deux options est que vous avez besoin d'un moyen de dire au navigateur d'incorporer les éléments `float` dans la hauteur de leur élément conteneur afin que leur arrière-plan puissent apparaître.

# Mise en page complète

Ensuite, nous voulons que notre fond `.page` remplisse toute la fenêtre du navigateur sans changer l'alignement de notre barre latérale ou des blocs de contenu. Le problème est que notre `.page` centre tout, nous ne pouvons pas l'utiliser pour un arrière-plan plein, car le centrage nécessite une propriété de largeur explicite.



Il est temps d'ajouter un autre conteneur `div`. Mettre une boîte autour de `.page` permet de continuer à centrer les éléments tout en nous donnant une place pour définir une propriété de couleur d'arrière-plan `background-color`. Changer notre élément `<body>` pour qu'il corresponde à ce qui suit:

```
<body>
  <nav class="menu">Menu</nav>
  <div class="container">
    <section class="page">
      <aside class="sidebar">Sidebar</aside>
      <article class="content">Content</article>
    </section>
  </div>
  <footer class="footer">Footer</footer>
</body>
```

N'oubliez pas que le comportement de rendu de bloc par défaut est que les éléments remplissent la largeur de leur conteneur. Donc, nous devrions pouvoir déplacer notre déclaration *background-color* vers une règle *.container* pour obtenir un fond complet:

```
.page {
  width: 900px;
  margin: 0 auto;
}

.container {
  overflow: hidden;
  background-color: #EAEDF0;
}
```

Comme dans la section précédente, nous avons encore besoin de la ligne *overflow: hidden* pour forcer le *.container* à prendre en compte la hauteur des éléments flottants. Sans elle, nous ne verrions pas notre couleur de fond parce que *.container* aurait la taille zéro.

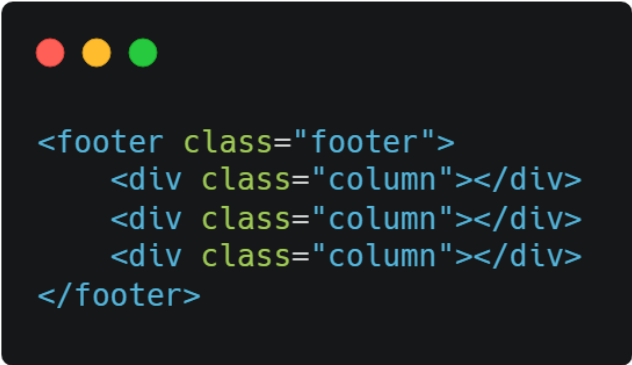
Cela nous donne trois éléments imbriqués `<div>` et `<section>` juste pour la mise en page: un “wrapper” `.container` pour la couleur de fond, un `.page` de largeur fixe pour le centrage de tous les éléments, et finalement les blocs `.sidebar` et `.content` alignés à gauche.

Ce genre de nidification et d'alignement est assez typique de la plupart des dispositions de site Web.

## Float pour des colonnes de largeur égale

Jusqu'à présent, nous avons vu la mise en page de la barre latérale, une mise en page à largeur fixe, et une mise en page complète. Float peut également être utilisé pour créer des mises en page multi-colonnes. Cela fonctionne exactement comme pour nos `.sidebar` et `.content`, en plus grand...

Nous allons ajouter trois colonnes de largeur égale à notre pied de page. Mettre à jour l'élément `<footer>`, comme suit:

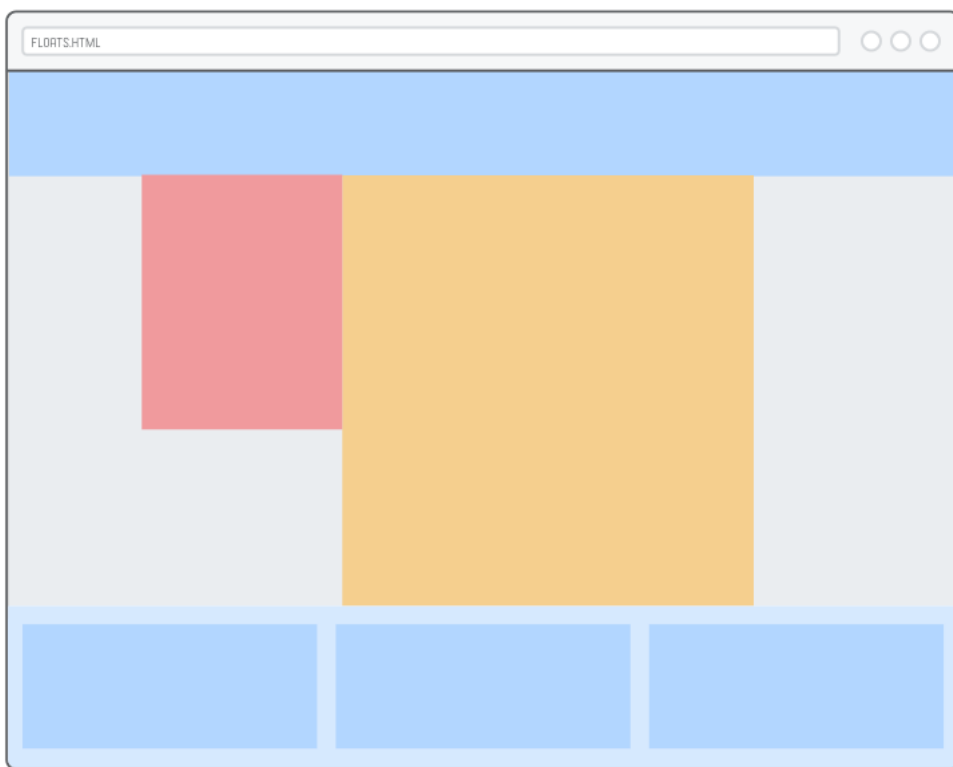


```
<footer class="footer">
  <div class="column"></div>
  <div class="column"></div>
  <div class="column"></div>
</footer>
```

Nous pouvons styler chacune de ces colonnes , tout comme nous l'avons fait pour le reste de notre page. Ajoutons une nouvelle règle à `styles.css`:

```
.column {  
  float: left;  
  width: 31%;  
  margin: 20px 1.15%;  
  height: 160px;  
  background-color: #B2D6FF;  
}
```

Ceci est la première fois que nous avons utilisé des valeurs de pourcentage au lieu de valeurs explicites de pixels. Les pourcentages sont en CSS par rapport à la largeur de l'élément parent. Le résultat est que les trois colonnes vont se redimensionner automatiquement à un tiers de la fenêtre du navigateur. Redimensionner la fenêtre du navigateur, et vous verrez nos colonnes se développer et se rétrécissent en conséquence. Ceci est le début de la conception responsive .

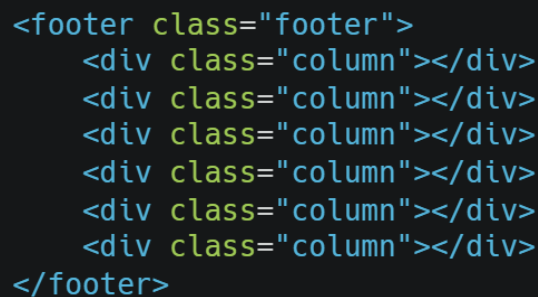




Quoi qu'il en soit, ne perdons pas de vue la thèse centrale de ce chapitre: les floats permettent d'empiler les objets horizontalement plutôt que verticalement. En modifiant la largeur des éléments que nous flottons, nous pouvons obtenir toutes sortes de mises en page différentes, des barres latérales à plusieurs colonnes en grilles.

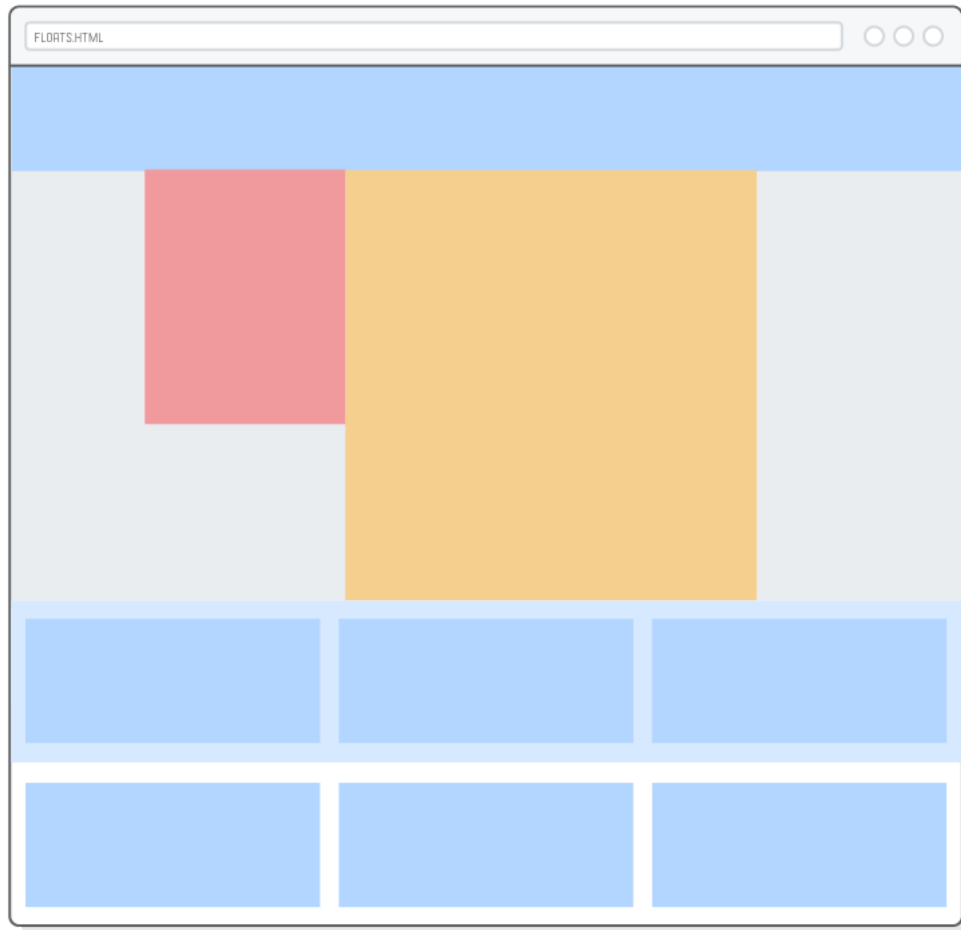
## Des floats pour une grille

Vous voulez une grille dans le bas au lieu de 3 colonnes? Pas de problème! Quand il n'y a pas assez de place pour empiler un élément flottant horizontalement, il saute à la ligne suivante. Tout ce que nous devons faire est d'ajouter quelques éléments `.column`:



```
<footer class="footer">
  <div class="column"></div>
  <div class="column"></div>
  <div class="column"></div>
  <div class="column"></div>
  <div class="column"></div>
  <div class="column"></div>
</footer>
```

Voilà une grille ! Enfin presque...



Notre arrière-plan du `.footer` est trop court. Heureusement, nous savons déjà comment résoudre ce problème. Remplaçons la hauteur explicite du pied de page avec une nouvelle propriété *overflow: hidden* pour faire en sorte qu'il puisse accueillir un certain nombre d'éléments de la grille:

```
.footer {  
  overflow: hidden;  
  background-color: #D6E9FE;  
}
```

Vous pouvez utiliser cette même technique pour faire des grilles de n'importe quelle taille. Par exemple, la création d'une galerie de photos avec un groupe de vignettes est simplement une question de mettre les

éléments de grille dans `.page` au lieu du pied de page et d'y ajouter des éléments `<img>`. Mais, encore une fois, rappelez-vous que flexbox est un moyen plus moderne de créer ces types de mises en page.

## Une brève note sur les conventions de nommage

Le nom de la classe `.column` n'est pas le plus juste. Ce scénario est un bon exemple de pourquoi nous voulons éviter les noms de classe qui se rapportent à l'apparence. "Colonne" n'est pas terrible parce que le contenu qu'il contient ne doit pas nécessairement être rendu dans plusieurs colonnes (par exemple, pour une disposition mobile, il n'y aurait probablement qu'une seule colonne). Un meilleur nom serait quelque chose comme `.footer-item`, pour élément de *footer*.

## Float pour le contenu

Il y a deux aspects à définir pour une disposition de page web. Vous avez votre structure de page globale, sur laquelle nous avons travaillé tout au long de ce chapitre. Il s'agit de choses comme où va se placer la sidebar, quel taille aura votre menu de navigation, etc... L'autre aspect des mises en page est le style des composants HTML individuels (votre contenu réel) qui sont à l'intérieur de cette structure de page globale.

Le processus de ce dernier est le même, il est juste imbriqué dans le premier. Nous allons ajouter du contenu factice à notre élément `.content` que nous ayons quelque chose à afficher:

```
<div class="container">
  <section class="page">
    <aside class="sidebar"></aside>
    <article class="content">
      
      <p>Ad netus sagittis velit orci est non ut urna taciti metus donec magnis hendrerit adipiscing mauris
        suspendisse ac scelerisque nascetur vestibulum parturient sed mi a dolor eu non adipiscing non
        neque scelerisque netus.
      </p>
      <p>Egestas at aliquam a egestas accumsan cum elementum consectetur conubia tristique eu et vitae
        condimentum in ante consectetur suscipit a a dui vestibulum gravida morbi sagittis.
      </p>
      <p>Ligula taciti vel primis sit a tincidunt habitant parturient.</p>
    </article>
  </section>
</div>
```

Nous avons une image et plusieurs paragraphes que nous pouvons styler tout comme nos *div* structurales.

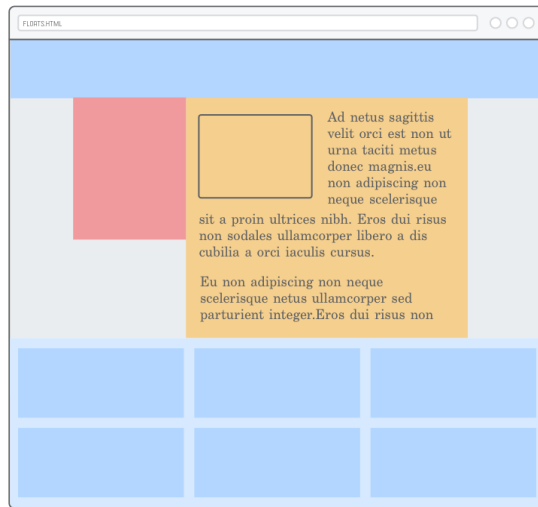
Par exemple, nous allons créer une mise en page de type magazine en faisant flotter l'image et de laisser le flux de texte autour d'elle.

Ajouter un couple plus de règles à notre feuille de styles:



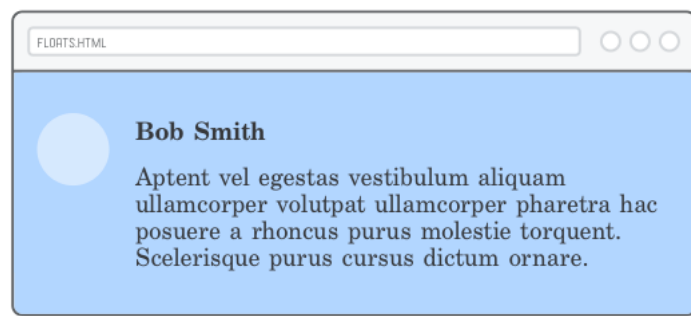
```
.content {  
    padding: 20px;  
}  
  
.article-image {  
    float: left;  
    width: 300px;  
    height: 200px;  
    margin-right: 20px;  
    margin-bottom: 20px;  
}  
  
p {  
    margin-bottom: 20px;  
}
```

Remarquez comment nous avons un *float* à l'intérieur d'un *float*, et tout fonctionne très bien. Habiller un site web est un processus récursif: vous construisez une structure de haut niveau pour travailler, alors vous remplissez avec votre contenu. Si la mise en page est plus complexe, il peut y avoir besoin d'une couche ou deux d'imbrication supplémentaire, mais l'idée est la même.



## Cacher le débord..... overflow:hidden (pour le contenu)

Vous trouverez des exemples de dispositions imbriqués partout. Pour notre dernier exemple, envisager un fil de commentaire utilisateur de plus classique. Vous avez une image qui est flotté à gauche avec un titre et un texte à côté de lui:



Essayons créer cela dans notre pied de page. Dans votre élément favori `.column`, ajouter ce qui suit:

```
<div class="column">
  <div class="avatar"></div>
  <h3 class="username">Rémi Martin</h3>
  <p class="comment">Aptent vel egestas vestibulum aliquam ullamcorper volutpat ullamcorper pharetra hac posuere a
    rhoncus purus molestie torquent. Scelerisque purus cursus dictum ornare a phasellus. A augue venenatis
    adipiscing.
  </p>
</div>
```

Et les règles CSS correspondantes:

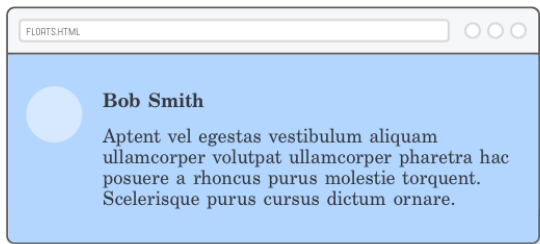
```
.avatar {
  float: left;
  width: 60px;
  height: 60px;
  margin: 25px;
  border-radius: 40px;
  background-color: #D6E9FE;
}

.username {
  margin-top: 30px;
}

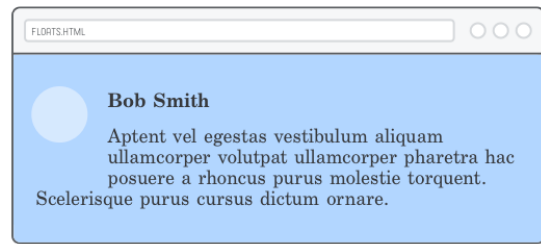
.comment {
  margin: 10px;
  font-size: 14px;
  overflow: hidden;
  /* Important */
}
```

Cela met en évidence un autre cas d'utilisation pour notre truc *overflow: hidden*. En l'ajoutant dans notre classe *.comment* cela fait en sorte que le texte soit "effacé horizontalement" (ce qui n'est pas un terme technique) de l'image flottante.

Sans elle, la dernière ligne du texte de *.comment* passe sous l'image.



**WITH HIDDEN OVERFLOW**



**WITHOUT HIDDEN OVERFLOW**

En d'autres termes, *overflow: hidden* brise la mise en page de type magazine de la section précédente, mais d'une manière très utile.

## La technique du clearfix

La méthode Clearfix utilise un pseudo-sélecteur CSS intelligent *:after* pour effacer les float. Plutôt que de définir la propriété *overflow* sur le parent, vous lui appliquez une classe supplémentaire que l'on nommera *clearfix* sur laquelle on appliquera le CSS suivant:

```
.clearfix:after {  
  content: ".";  
  visibility: hidden;  
  display: block;  
  height: 0;  
  clear: both;  
}
```

Ceci ajoutera un petit peu de contenu invisible après l'élément parent qui effacera le ou les *float*.