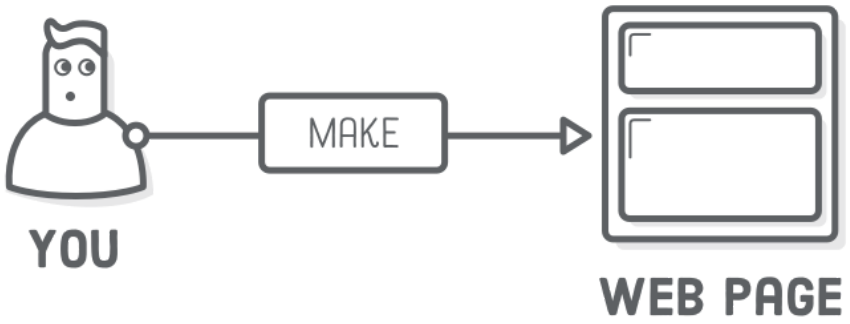
Introduction

L’apprentissage HTML et CSS est difficile, mais ne devrait pas l’être. Nous allons à travers ce cours, choisir un bon éditeur de texte (ce qui est étonnamment important) pour la construction de nos pages web de qualité professionnelle et partir de zéro ou presque.



Notre objectif est de le rendre aussi facile que possible pour les étudiants le fait de devenir des développeurs web professionnels.

Html, css & javascript

HyperText Markup Language (HTML), Cascading Style Sheets (CSS) et JavaScript sont les langages qui font le web. Ils sont très étroitement liés, mais ils sont aussi conçus pour des tâches très spécifiques. Comprendre comment ils interagissent sera un long chemin qu’il vous faudra parcourir pour devenir un développeur web. Nous allons développer cela au fur et à mesure du déroulement de ce cours, l'essentiel étant de retenir ceci :

- HTML est pour ajouter un sens à du contenu brut en utilisant des balises.
- CSS est pour le formatage du contenu balisé.
- JavaScript est pour rendre interactif le contenu et le formatage.

Pensez le HTML comme étant le texte et les images dans une page Web, le CSS comme la page qui est effectivement affichée, et le JavaScript comme des événements, des comportements qui peuvent manipuler HTML et CSS.

Par exemple, vous pouvez baliser un terme particulier de texte comme un paragraphe avec ce HTML.

```
<p id="un-paragraphe">Ceci est un paragraphe.</p>
```

Ensuite, vous pouvez définir la taille et la couleur de ce paragraphe avec quelques CSS :

```
p{
font-size: 20px;
}
```

Et, si vous voulez obtenir un effet saisissant, vous pouvez ré-écrire ce paragraphe lorsque l'utilisateur clique avec un peu de JavaScript :

```
var p=document.getElementById('un-paragraphe');
p.addEventListener('click', function( event ){
  p.innerHTML = 'Click !';
});
```

Comme vous pouvez le voir, HTML, CSS et JavaScript sont des langages totalement différents, mais ils se réfèrent tous les uns aux autres en quelque sorte. La plupart des sites comptent sur tous les trois, mais l'aspect de chaque site est déterminé par le HTML et la CSS.

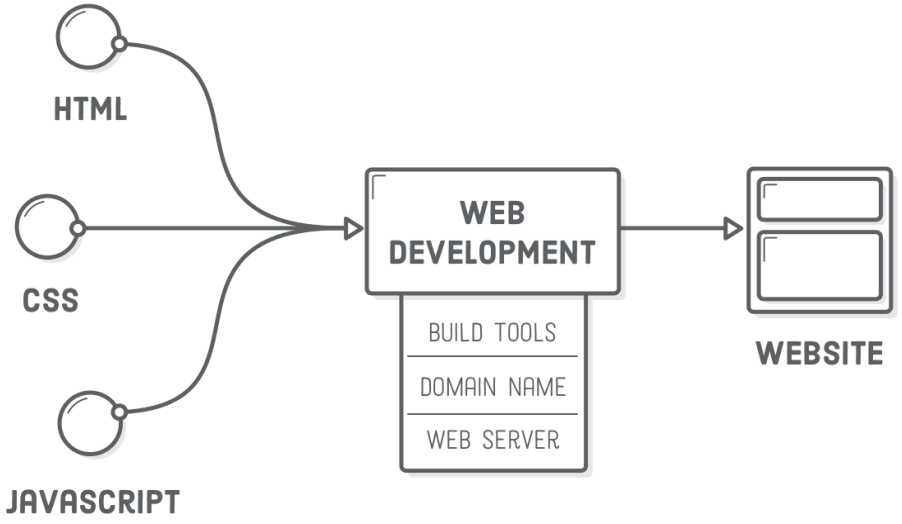
Les langages et le "développement web"

Malheureusement, la maîtrise de HTML, CSS et JavaScript est seulement une condition préalable pour devenir un développeur web professionnel. Il y a un tas d'autres compétences pratiques dont vous aurez besoin pour exécuter un site Web:

- Organiser le HTML dans des modèles réutilisables;
- Préparer un serveur web;
- Déplacer de fichiers à partir de votre ordinateur local sur votre serveur web;
- Revenir à une version précédente lorsque vous ratez quelque chose;
- Configurer un nom de domaine pour votre serveur;
- ...

Faire face à ces complexités implique la mise en place de divers «environnements» pour organiser vos fichiers et gérer la construction et le déploiement de votre site Web. Tout cela est orthogonal au HTML, CSS et au code JavaScript qui composent un site Web.

Ce cours se concentre entièrement sur les langues de HTML et CSS, pas la mise en place de ces environnements sous-jacents.



Mais pour ne pas être dépassés, acquérir la maîtrise de HTML et CSS est une première étape importante en vue de devenir un développeur web accompli.

Outillage

Pour ce cours, un éditeur de texte et un navigateur web décent est tout ce dont vous aurez besoin. Votre travail de base est d'écrire du code dans votre éditeur de texte, puis l'ouvrir dans un navigateur Web pour voir à quoi il ressemble. Comme vous commencez à créer vos propres sites Web, vous finirez par ajouter d'autres outils à votre boîte à outils, mais il est important de commencer à minima de bien apprendre les rudiments de HTML et CSS.

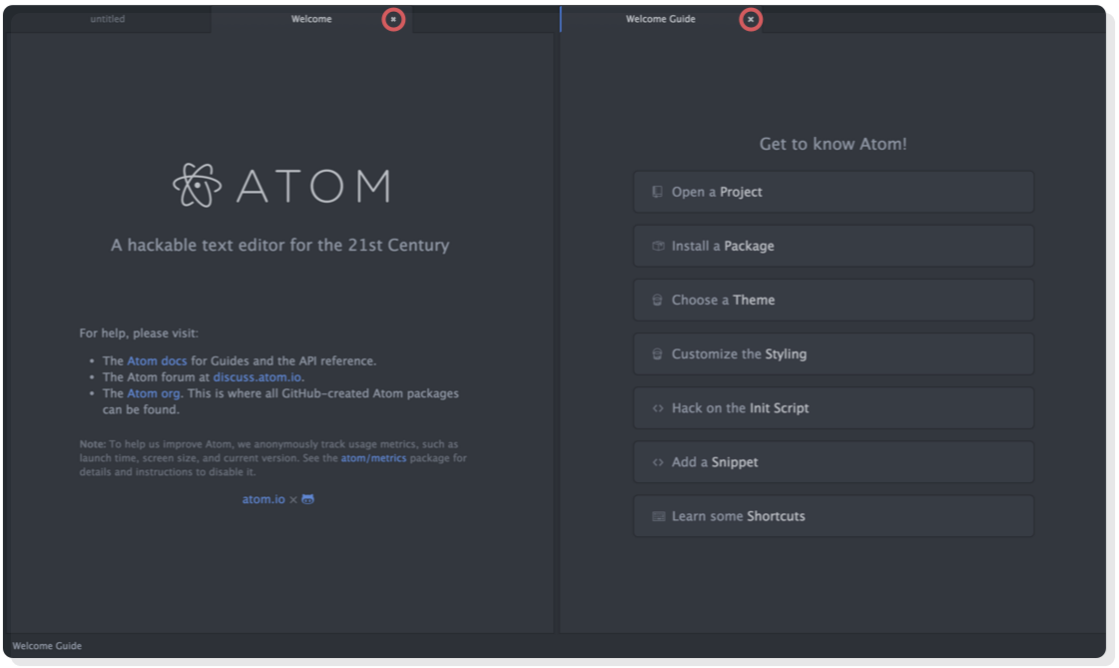
Cela dit, il vous faut prendre le temps de vraiment maîtriser votre éditeur de texte. Les meilleurs viennent avec des fonctionnalités qui vous permettent d'écrire du code plus rapidement que vous pouvez le faire normalement, avec des fonctions comme l'auto-complétion les balises, naviguer dans votre texte et naviguer dans votre système de fichiers. Tirer pleinement parti de votre éditeur de texte est une partie nécessaire de l'apprentissage du HTML et du CSS.

La seule condition réelle pour un bon navigateur Web est qu'il se mette à jour et soit d'un usage courant. Chrome et Firefox sont les favoris parmi les développeurs web. Safari est bien si vous utilisez OS X, aussi. Nous vous suggérons fortement de ne pas créer des sites Web avec Internet Explorer, avec Edge c'est mieux mais moins bien qu'avec les concurrents. Cependant le développement web professionnel exige souvent de tester le code sur tout ces navigateurs et ce de la façon la plus efficace.

Atom l'éditeur de texte

Nous recommandons Atom comme éditeur de texte. Il est simple, même pour les débutants, fournit toutes les fonctionnalités utiles que nous avons mentionnés ci-dessus, et est disponible pour tous les principaux systèmes d'exploitation. Il est également configurable à l'infini, ce qui deviendra important quand vous identifierez les tâches répétitives que vous pourrez automatiser.

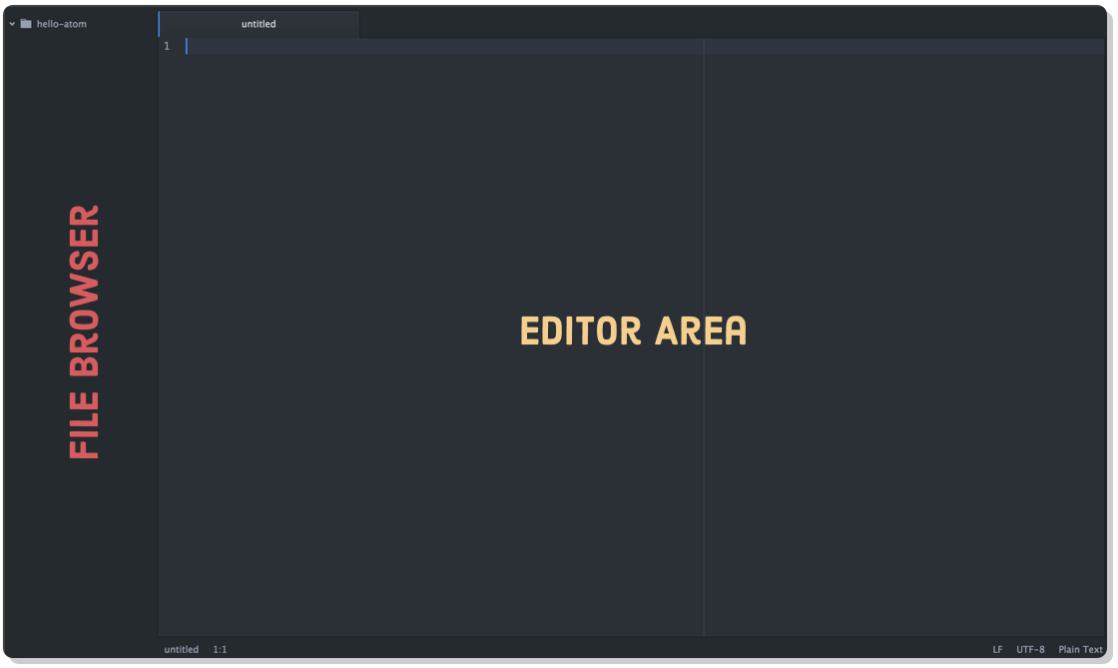
Une fois que vous l'avez téléchargé, ouvrez ATOM afin que nous puissions avoir un aperçu de ses principales caractéristiques. Vous devriez voir deux volets avec différents écrans d'accueil:



Nous ne voulons pas de ces écrans d'accueil, pour les fermer tous les deux en cliquant sur le x icône dans leurs onglets correspondants. Vous pouvez également utiliser les raccourcis Cmd + W (Mac) ou Ctrl + W (Windows / Linux) pour les fermer (raccourcis sont grands, les utiliser quand vous le pouvez). Vous devez avoir qu'un seul onglet untitled.

Création d'un projet

Chaque site Web que vous créez dans Atom est un «projet» , qui est essentiellement un dossier sur votre système de fichier qui contient un tas de fichiers HTML et CSS. Nous allons explorer Atom en créant un projet pour de faux et en ajoutant quelques fichiers texte à elle. Cliquez sur *Fichier* > *Ouvrir* dans la barre de menu pour ouvrir une fenêtre de dialogue de fichier, puis sélectionnez **Nouveau dossie**r pour créer un nouveau dossier. Appelez-le **hello-atom** et cliquez sur **Ouvrir**.



Vous devriez maintenant voir une barre latérale à gauche de l'interface qui dit **hello-atom** en haut à côté d'une petite icône de dossier. Ceci est notre navigateur de fichiers. Bien sûr, il ne sera pas montrer quoi que ce soit jusqu'à ce que nous ajoutons quelques fichiers, alors faisons-le suivant.

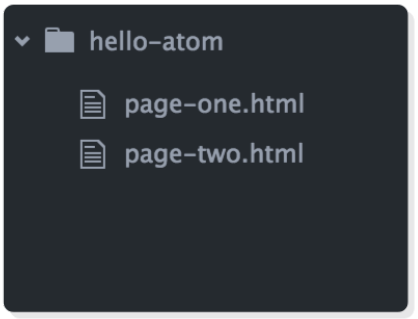
Création de fichiers

Ajouter un texte arbitraire à l'onglet *untitled*, puis appuyez sur Cmd + S (Mac) ou Ctrl + S (Windows, Linux) pour enregistrer le fichier. Appelez le **page-one.html**. Après l'enregistrement, vous devriez le voir apparaître dans le navigateur de fichiers Atom.

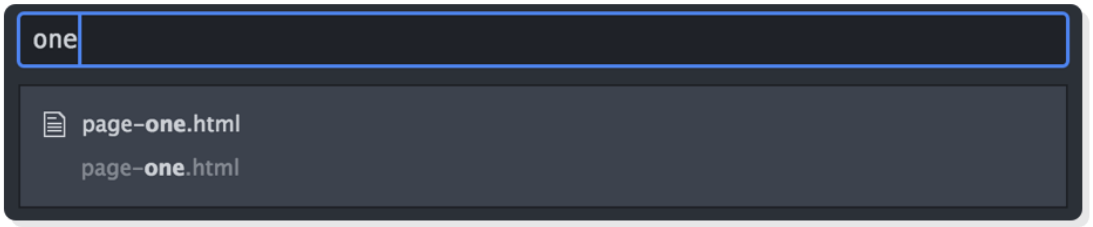
Faisons encore un fichier en appuyant sur Cmd + N (Mac) ou Ctrl + N (Windows, Linux). Cela va créer un autre onglet sans titre. Comme notre dernier fichier, ajoutez le texte que vous voulez, puis l'enregistrer comme **page-two.html**.

Naviguer sur le système de fichiers

Encore une fois, l'un des aspects les plus importants d'un éditeur de texte approprié est de vous permettre de naviguer efficacement dans tous les fichiers dans votre projet. Dans Atom, vous pouvez sélectionner l'onglet du fichier que vous souhaitez travailler ou le trouver dans le navigateur de fichiers sur le côté gauche de l'interface. Vous pouvez également utiliser *Ctrl* + *Tab* pour basculer entre les onglets ouverts.



C'est bien beau pour la navigation des fichiers, mais il y a beaucoup de moments où vous êtes la recherche d'un fichier spécifique. Par exemple, imaginez la découverte d'un lien cassé sur votre site Web. Vous voulez être en mesure d'atteindre ce fichier avec Atom pour fixer le lien le plus rapidement possible.

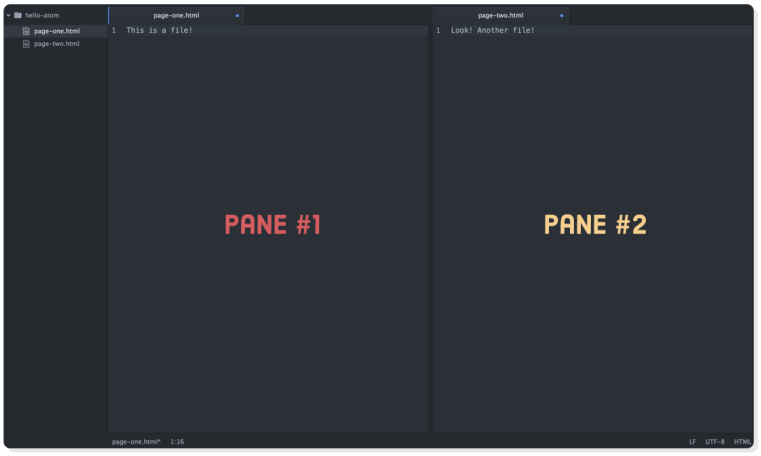


Pour cela, vous avez besoin du fuzzy finder d'Atom, qui est accessible via Cmd + T (Mac) ou Ctrl + T (Windows, Linux). Lorsque vous appuyez sur ce, Atom va ouvrir une barre de recherche et vous permettre de taper une partie du nom de fichier que vous recherchez. Essayez de fermer les deux

onglets, frapper Cmd + T ou Ctrl + T , et en entrant «one» . Le page-one.html fichier devrait apparaître, et vous pouvez faire Entrée pour le modifier. Cette fonctionnalité est indispensable une fois que votre projet se développe à quelques dizaines de fichiers répartis sur plusieurs dossiers.

Volets multiples

Atom non seulement vous permet d'avoir plusieurs onglets, mais aussi plusieurs volets. Pour voir ce dont nous parlons, essayez un clic droit sur un des fichiers dans le navigateur de fichier et en sélectionnant le panneau droit . Cela permettra d'ouvrir ce fichier dans une nouvelle fenêtre, vous permettant de voir plusieurs fichiers en même temps.



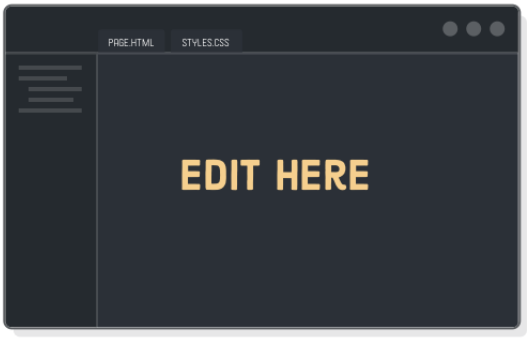
De multiples panneaux sont vraiment utiles pour l'examen d'un fichier CSS et son fichier HTML correspondant au même moment.

En dehors d’Atom

Enfin, nous allons de temps en temps besoin de travailler avec des fichiers en dehors de l'interface Atom (par exemple, lorsque l'on veut copier des fichiers d'image dans notre site Web). Nous pouvons utiliser intégré dans le navigateur de fichiers du système d'exploitation pour cela. Droit-cliquez sur un fichier dans le navigateur de fichiers Atom, puis sélectionnez Afficher

dans le Finder / Explorer / Certains navigateur de fichiers Autres pour l'ouvrir dans le navigateur de fichier par défaut de votre système.

De là, vous pouvez ajouter de nouveaux fichiers, créer des dossiers, ou ouvrir des fichiers HTML dans un navigateur web. Ce dernier va devenir une tâche commune pour le reste de ce tutoriel, nous allons donc donner le coup d'envoi avec notre fichier *page-one.html*. Faites un clic droit dans le navigateur de fichier par défaut de votre système et sélectionnez Ouvrir avec> Chrome / Firefox / Safari . Vous devriez voir le texte que vous avez ajouté au fichier de rendu en tant que page Web dans votre navigateur Web par défaut.



Maintenant, vous pouvez modifier le contenu de *page-one.html* dans Atom, enregistrez le, et le recharger dans votre navigateur en appuyant sur Cmd + R (Mac) ou Ctrl + R (Windows, Linux). Ceci est le flux de travail d'édition de base pour tous les développeurs web, et vous deviendrez très, très habitués à cela.

Les fonctions d'Atom que nous venons de découvrir vont devenir une partie de notre routine quotidienne, alors assurez-vous êtes à l'aise avec tout avant de passer à la suite. Nous vous encourageons à jouer avec l'exemple de projet un peu, en ajoutant plus de fichiers et de pratiquer la navigation de l'un à l'autre. Maîtriser l'art de l'écriture du HTML et du CSS est ce qui distingue les développeurs productifs des autres. Maintenant que nous avons un éditeur de texte proprement dit, nous sommes prêts à commencer à coder quelques pages Web réelles.

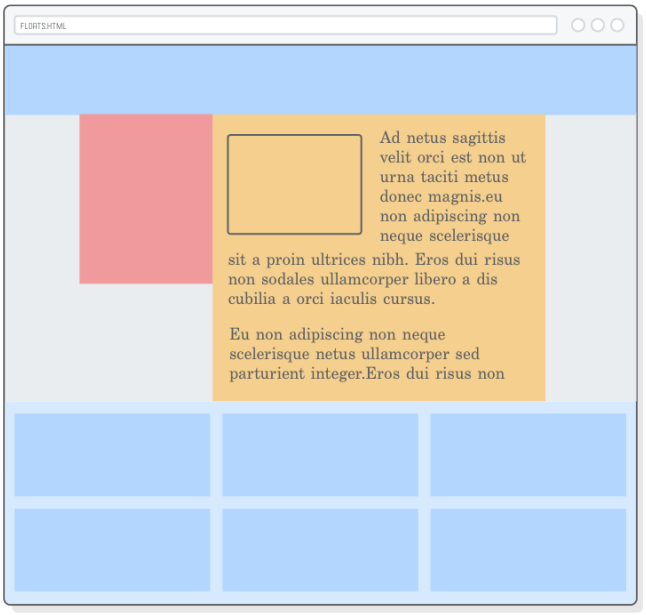
Float

"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 a 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 sur Internet, ce qui signifie que vous aurez certainement les rencontrez 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='en'>
  <head>
    <meta charset='UTF-8' />
    <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 pour la plupart des sites Web sur Internet. Nous avons un endroit pour mettre un menu de navigation, une barre latérale, le contenu principal de la page, et un pied de page. Pensez à tous ces divs conteneur que vous pouvez mettre votre contenu HTML réelle. Vous ne verrez pas beaucoup quand vous ouvrez *floats.html* dans un navigateur parce que les éléments vides ont la 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 de la boîte 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 l'avons introduit dans les éléments en bloc par rapport aux éléments en ligne, mais il est sur le point de devenir beaucoup plus important.
Nous pouvons obtenir un meilleur regard sur notre page d'exemple en ajoutant des couleurs de fond et des hauteurs explicites à chacun de nos éléments * *. Ajoutez ceci à *styles.css*:

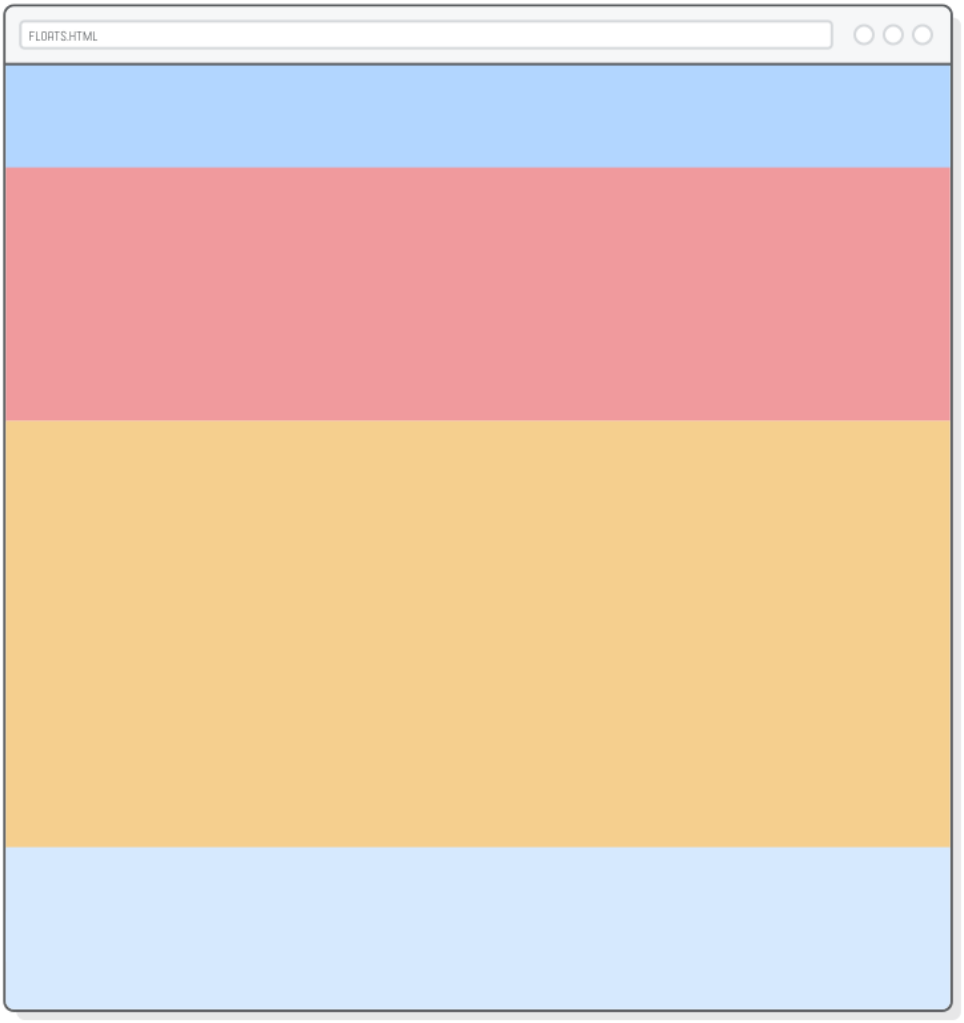
```
.menu {
  height: 100px;
  background-color: #B2D6FF;    /* Medium blue */
}

.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 qu'il 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 exacte. Tous les blocs sont toujours rendus 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.

Allez-y et flottez 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* serait enveloppé dans la boîte de la barre latérale. Cela nous donne une présentation de style magazine:

Vous pouvez également 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*.

```
    float: right;  /* Aligné à droite */
    float: none;   /* Revient au flux par défaut */
```

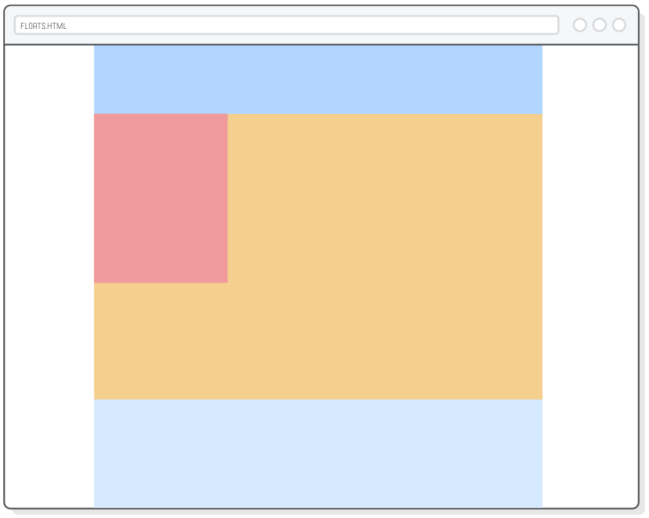
Nous disposons maintenant de tous les outils nécessaires pour aligner les éléments au niveau des blocs: *float* pour alignement gauche / droit et marges automatiques pour l'alignement central. N'oubliez pas que ceci s'applique uniquement aux éléments bloc. Les éléments en ligne sont alignés avec la propriété *text-align*.

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.



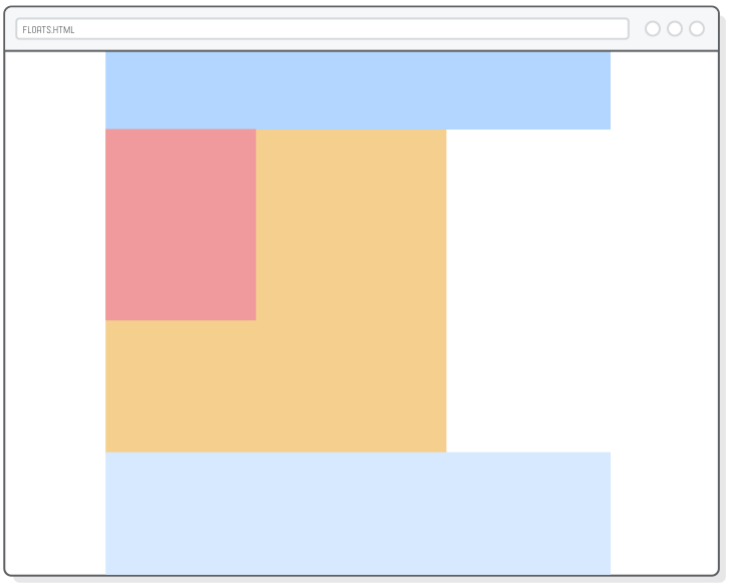
Le positionnement des éléments imbriqués de conteneur comme ceci c'est ainsi que vous construisez des dispositions sophistiquées de site Web. Ici, nous avons commencé avec *.page* pour centrer tout, puis nous avons aligné à gauche une barre latérale à l'intérieur de cette page centrée. Les choses peuvent 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;
}
```

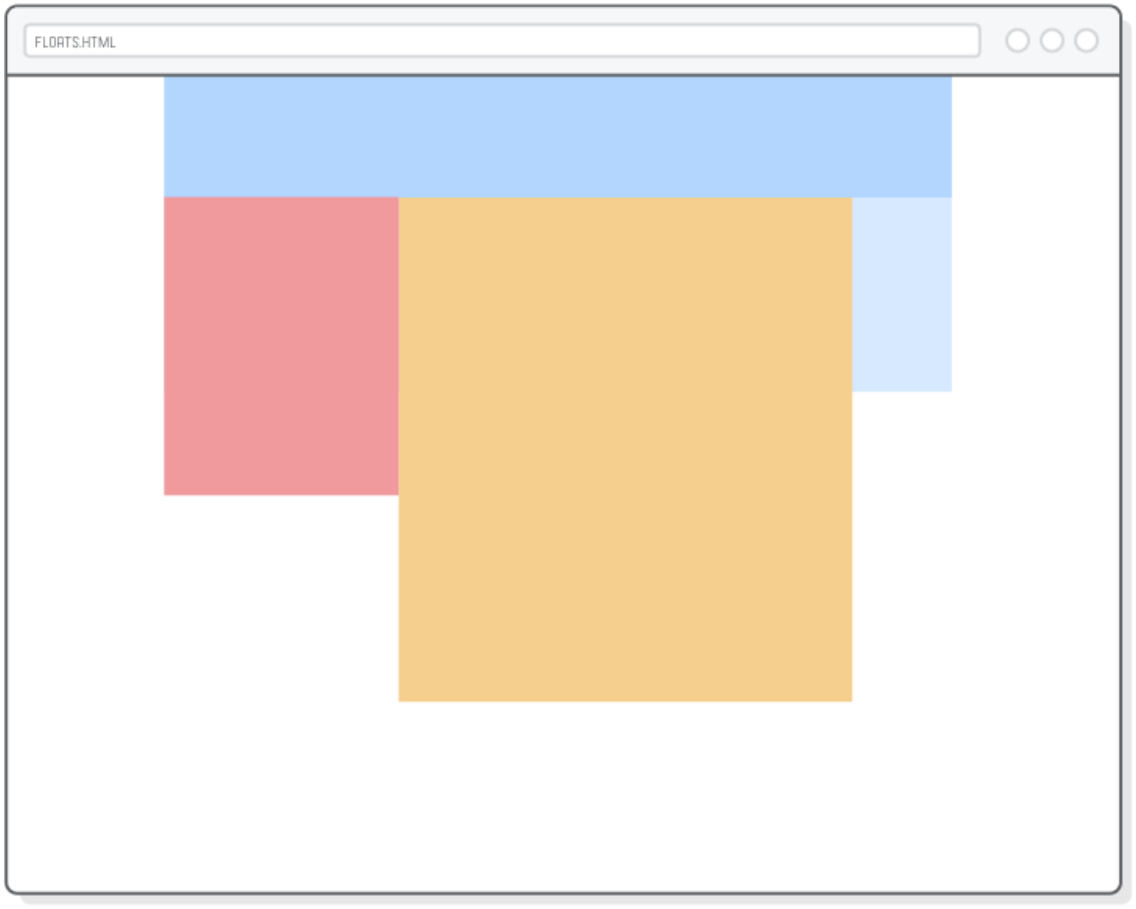
Cela démontre clairement que notre barre latérale est en fait dans le bloc *.content*: si vous prenez une capture d'écran, vous aurez une image de 650 pixels de large opposée à 850 pixels (notre barre latérale est de 200 pixels de large).



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*:

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

Lorsque vous flottez plusieurs éléments dans la même direction, ils s'empilent horizontalement, tout comme l'algorithme de mise en page verticale par défaut. Le code ci-dessus fait apparaître notre bloc de contenu entier à droite de la barre latérale au lieu de l'entourer.

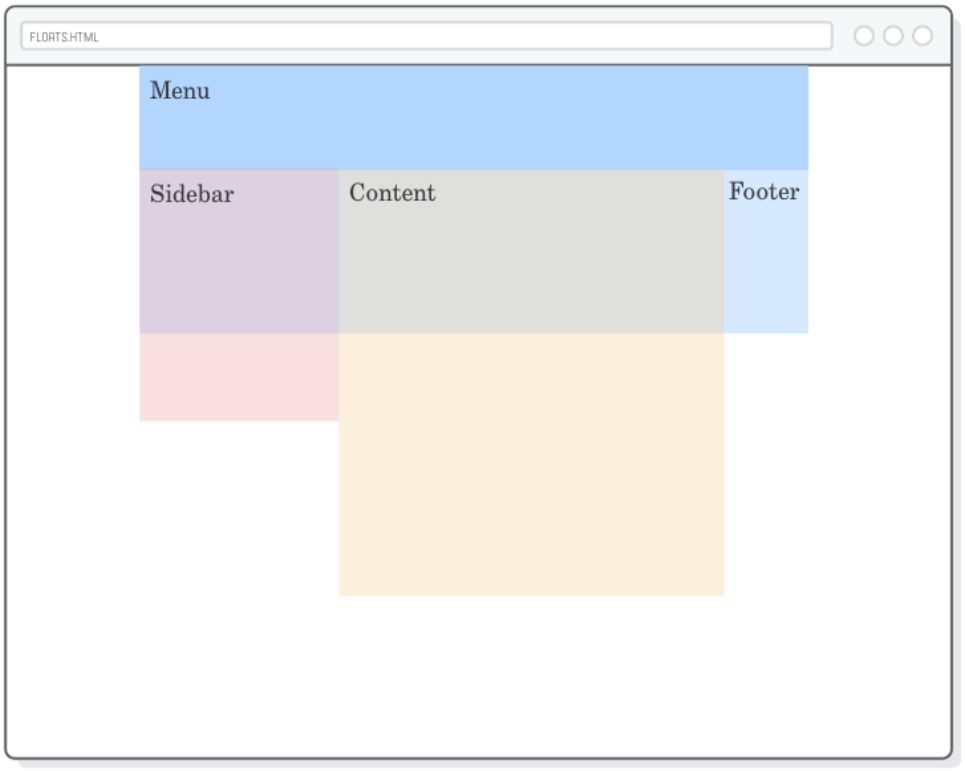


Cela nous donne un véritable contrôle sur l'alignement horizontal de nos boîtes de bloc. Essayez de jouer avec les valeurs `float` pour les deux `.sidebar` et `.content`, et vous verrez que nous avons déjà quelques dispositions distinctes à notre disposition:

Assurez-vous que les deux sont flottant à gauche. Cela fait ce que l'on attend de 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 .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 -clear- d'un float ou en cachant le débordement -overflow-.

Compensation -clear- de float

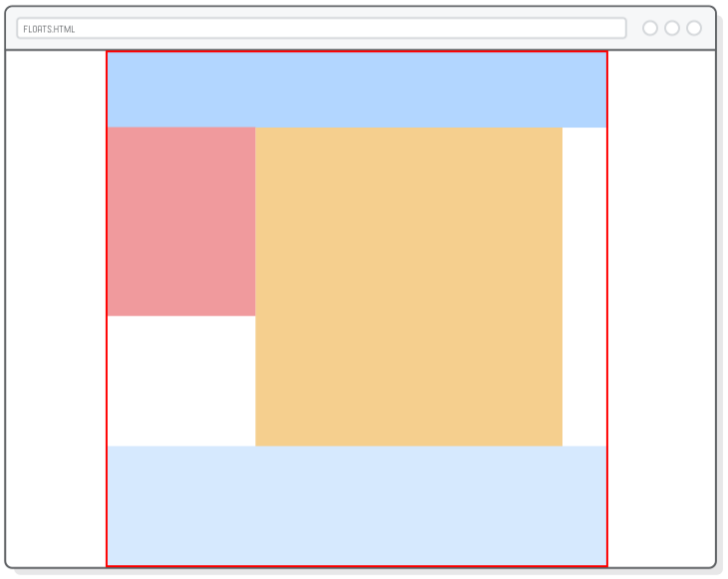
"Compenser" un float est quand nous disons à un bloc d'ignorer les float qui se présentent devant lui. Au lieu de couler autour de la boîte flottante, un élément clear apparaîtra toujours après des float. Il va forcer une boîte à faire un retour dans le flux vertical par défaut de la page.

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:

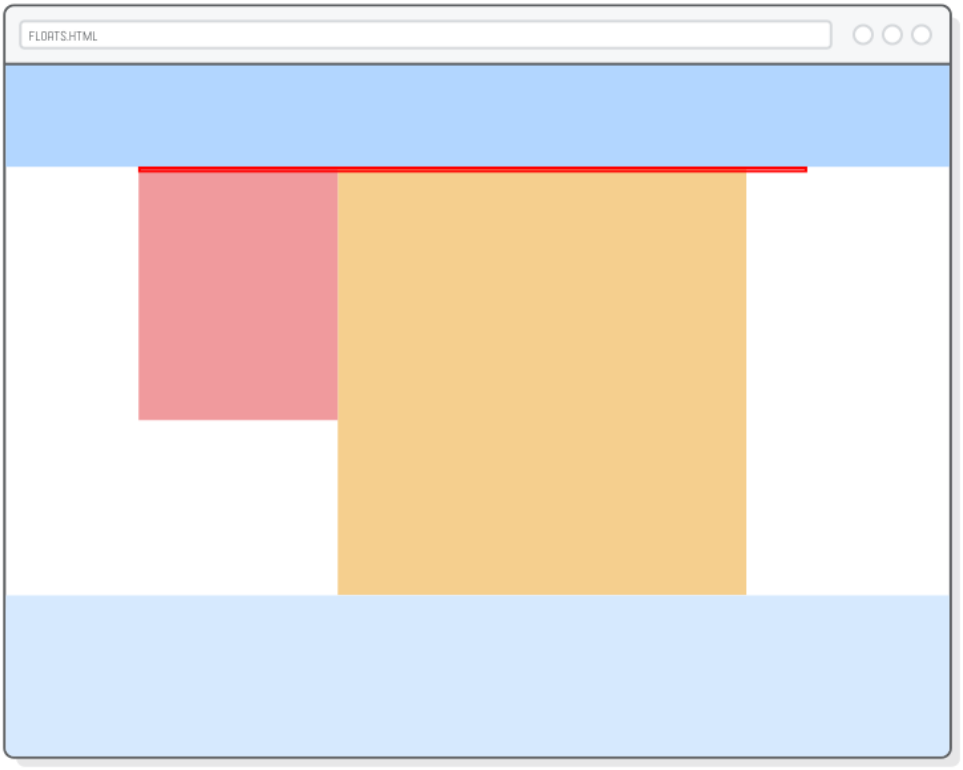
```
<body>
    <nav class='menu'>Menu</nav>

    <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

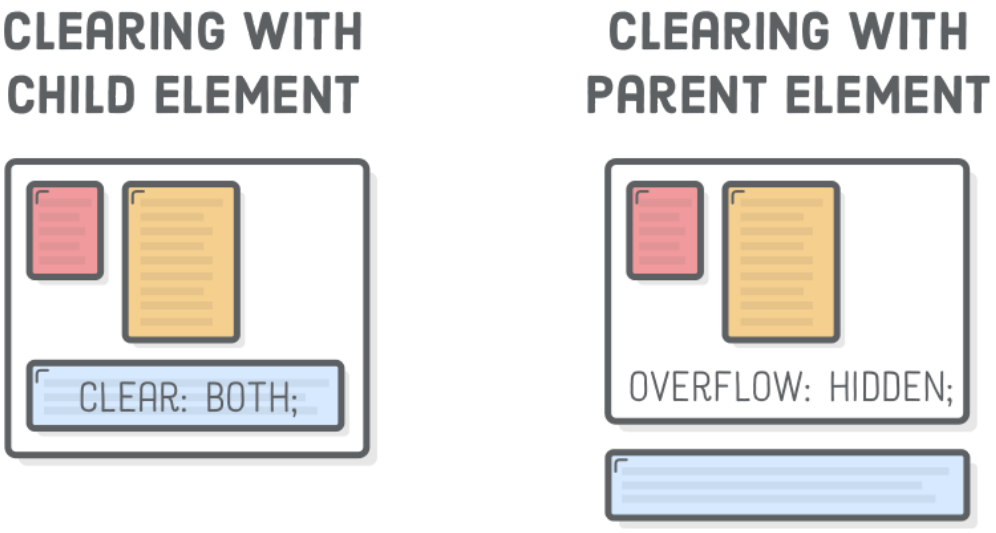
comment .page a encore une fois la hauteur zéro malgré le fait que le pied de page efface toujours la barre latérale et le contenu des blocs.



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 ya 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.

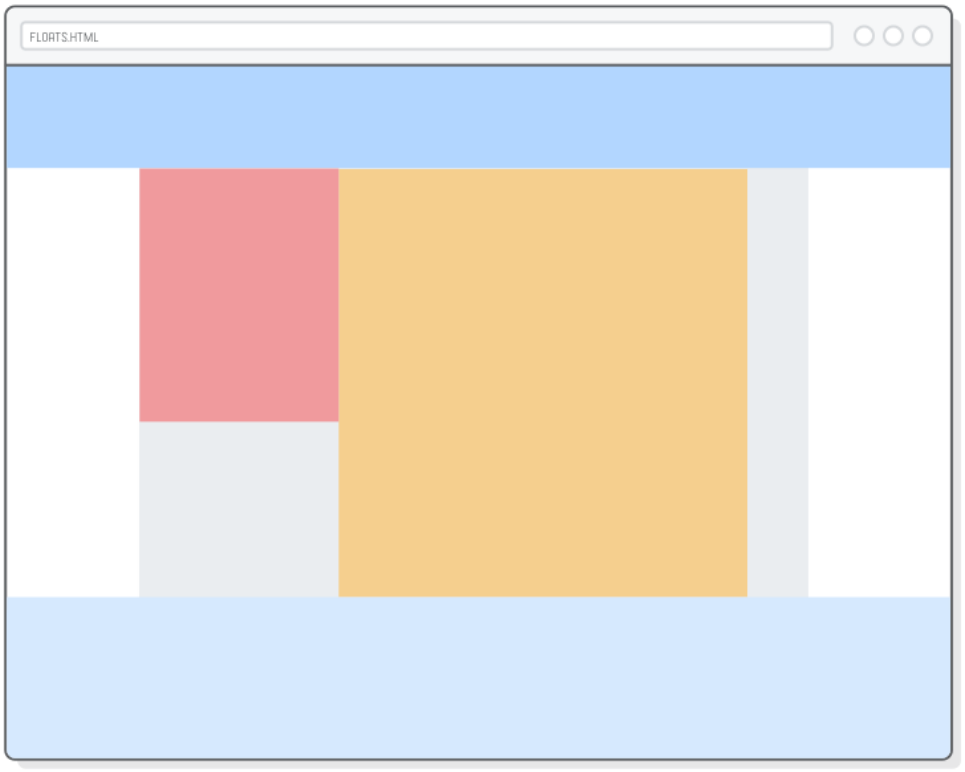


La solution est la propriété CSS overflow. En ajoutant une déclaration `overflow: hidden` à un div conteneur, nous lui disons de prendre en compte la hauteur de tous les éléments flottants qu'il contient.

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 */
}
```

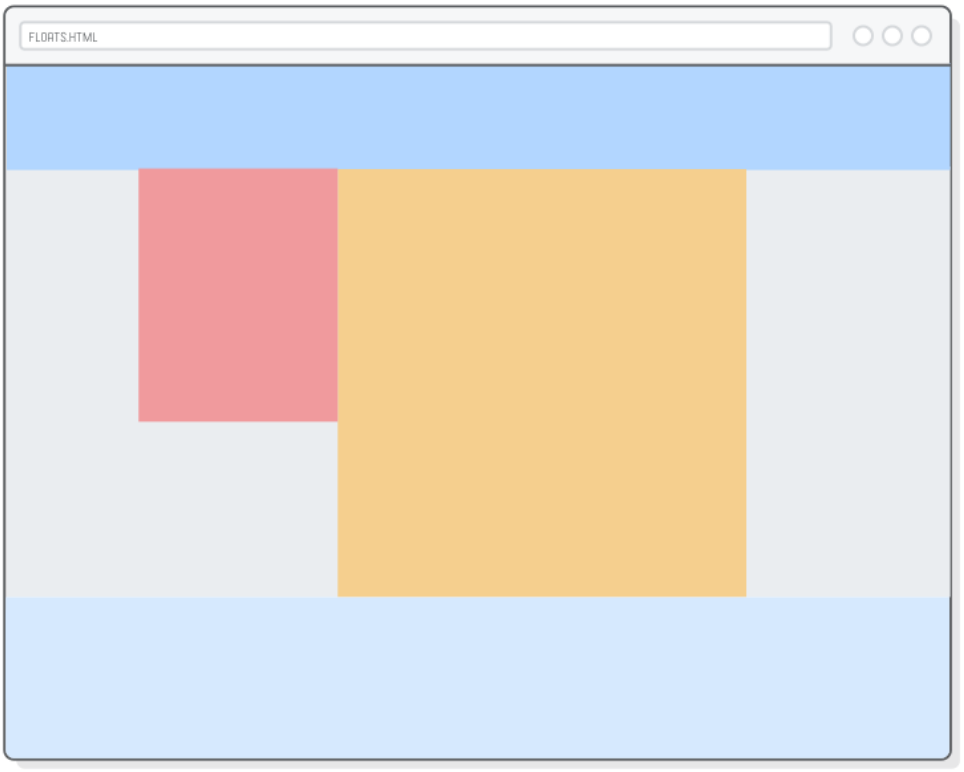
Vous devriez maintenant être en mesure de voir un fond gris clair pour `.page` au lieu du blanc par défaut. Ce n’est pas encore une mise en page complète (nous allons aborder que dans la section suivante). L'important ici est le comportement de `overflow: hidden`. Sans cette propriété, nous ne serions 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 filtré 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 float dans la hauteur de leur élément de 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 la .page permet de continuer à centrer les choses 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'>   <!-- Nouveau -->
   <section class='page'>
     <aside class='sidebar'>Sidebar</aside>
     <article class='content'>Content</article>
   </section>
  </div>                          <!-- Nouveau -->
  <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 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...

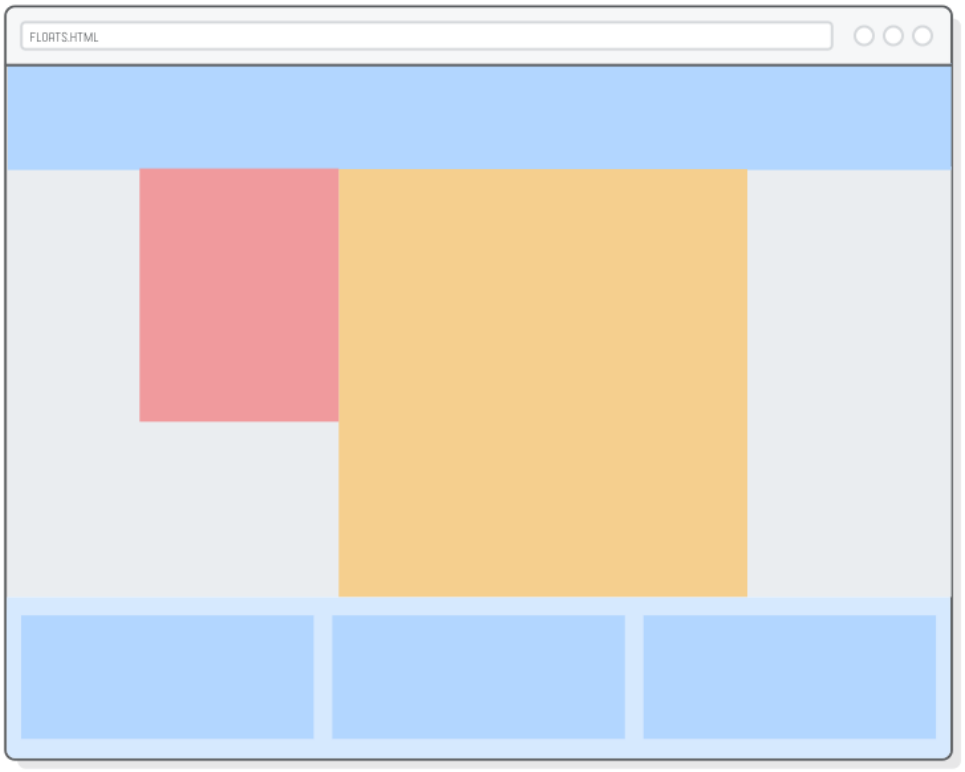
Nous allons ajouter trois colonnes de largeur égale à notre pied de page. Mettre à jour l'élément <footer>, comme suit:

```
<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 développent et se rétrécissent en conséquence. Ceci est le début de la conception responsive .



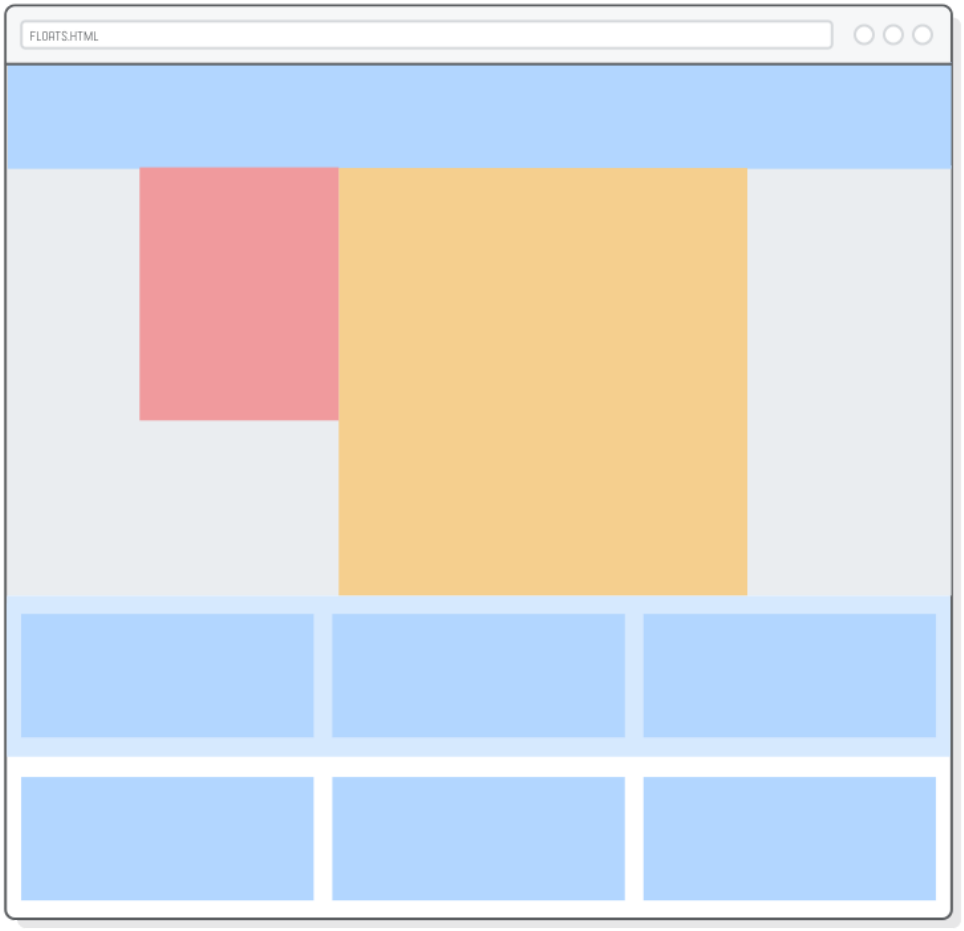
Quoi qu'il en soit, ne perdons pas de vue la thèse centrale de ce chapitre: les floats permettent d'empiler les objets horizontalement plutôt que verticalement. En modifiant la largeur des éléments que nous flottons, nous pouvons obtenir toutes sortes de mises en page différentes, des barres latérales 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 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 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 vous sidebar va se placer, 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'>
      <img src='?' class='article-image'>
      <p>Ad netus sagittis velit orci est non ut urna taciti metus donec magnis hendrerit adipiscing mauris sit a proin ultrices nibh.</p>
      <p>Enim suspendisse ac scelerisque nascetur vestibulum parturient sed mi a dolor eu non adipiscing non neque scelerisque netus ullamcorper sed parturient integer.Eros dui risus non sodales ullamcorper libero a discubilia a orci iaculis cursus.</p>
<p>Egestas at aliquam a egestas accumsan cum elementum consectetur conubia tristique eu et vitae condimentum in ante consectetur suscipit a a duis vestibulum gravida morbi sagittis.Parturient scelerisque facilisis ullamcorper a a pretium a nisl parturient semper senectus accumsan ipsum mus scelerisque eget ridiculus.Accumsan dolor a.</p>
      <p>Ligula taciti vel primis sit a tincidunt habitant parturient parturient in parturient ante nulla consectetur sem.Facilisis parturient litora.</p>
    </article>
  </section>
</div>
```

Nous avons une image et plusieurs paragraphes que nous pouvons styler tout comme nos divs structurelles.

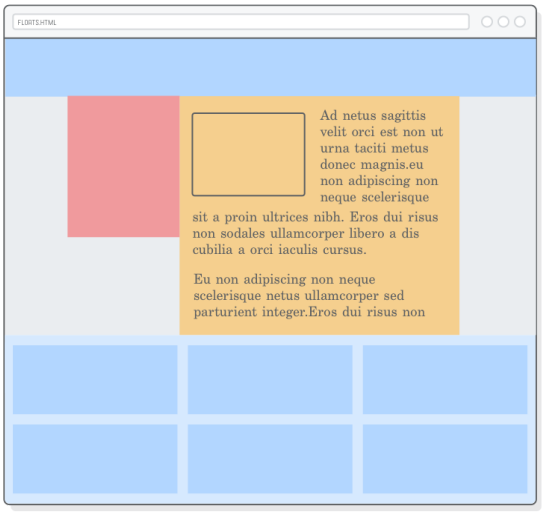
Par exemple, nous allons créer une mise en page de type magazine en faisant flotter l'image et de laisser le flux de texte autour d'elle. Ajouter un couple plus de règles à notre feuille de styles:

```
.content {
  padding: 20px;
}

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

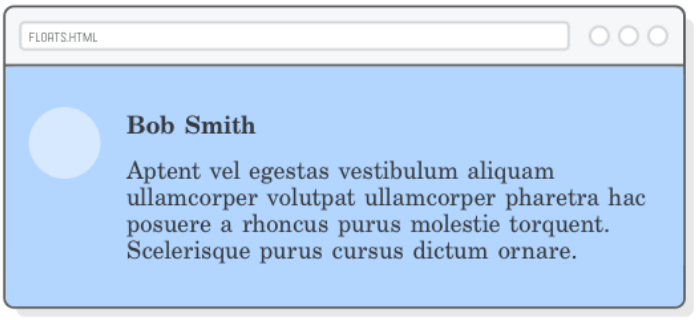
```
p {
  margin-bottom: 20px;
}
```

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 mises 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:



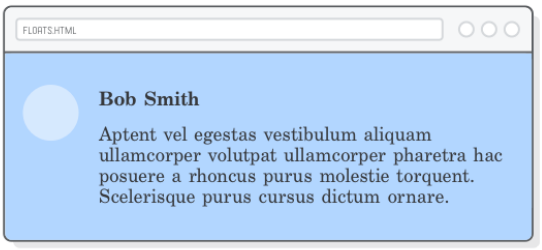
Essayons créer cela dans notre pied de page. Dans votre élément favori .column, ajouter ce qui suit:

```
<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>
```

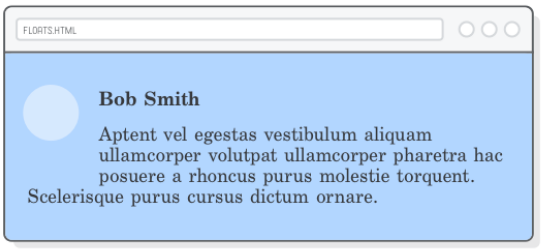
Et les règles CSS correspondantes:

```
.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.