HTML-CSS

Claude Petitpierre, André Maurer, complété par Brice Canvel

Automne 2010

Table des matières

Chapitre 1

Pages du Web

1.1 Introduction

1.2 Le langage HTML

Les pages qui suivent présentent le langage HTML. Il s'agit du langage qui permet d'afficher des informations sur une page web. En particulier, les points suivants sont discutés.
– les principes de base et la structure d'un document HTML
– comment mettre en forme du texte (gras, italique, etc...)
– comment afficher des images
– comment mettre du texte dans un tableau
– comment créer des liens vers d'autres documents.

1.2.1 Qu'est-ce que HTML ?

Définition

HTML est un langage pour décrire des pages web.
HTML est un acronyme pour **H**yper **T**ext **M**arkup **L**anguage (langage de balisage d'hypertexte)

Exemple de code HTML

Voici un exemple de code HTML :

```
<html>
  <head>
    <title>Le titre de la page</title>
  </head>
  <body>
    <h1>Mon premier titre</h1>
    <p>Mon premier <b>paragraphe</b></p>
  </body>
</html>
```

FIGURE 1.1 – Un exemple de code HTML

Lecture du code HTML

Les navigateurs Internet (comme Internet explorer, Firefox ou Chrome par exemple) sont capables de lire les documents HTML et de les afficher, sous forme de pages web. Certains éléments, comme <html>, <body>, etc... ne sont pas affiché sur la page web. Ces éléments donnent des indications au navigateur sur la façon d'afficher le texte.
Le code HTML ci-dessus, interprété par un navigateur, donne le résultat suivant :

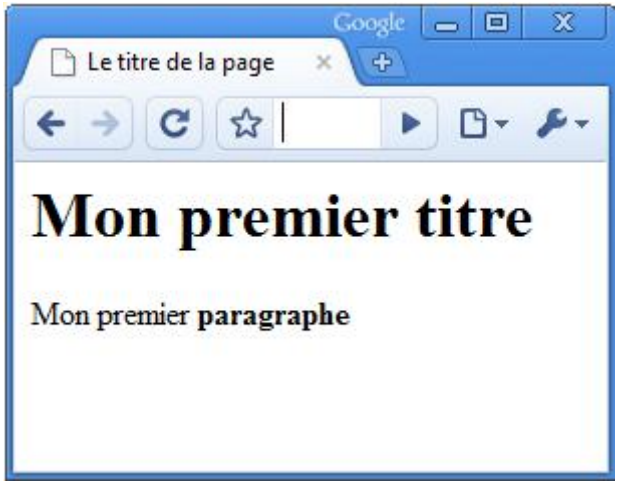


FIGURE 1.2 – Le code HTML de la figure 1.1 interprété par Google Chrome

1.2.2 Reconnaître une balise

Document HTML = textes + balises

Pour composer une page web, on écrit donc du texte, et on y insère des caractères spéciaux qui permettront au navigateur (par exemple Firefox ou Internet Explorer), de mettre en forme ce texte. Ces caractères spéciaux se nomment des balises Un document HTML est donc composé de
– textes
– balises
Sur la figure 1.1, les balises sont

```
<html>, <head>, <title>, </title>, </head>,
<body>, <h1>, </h1>, <p>, <b>, </b>, </p>,
</body>, </html>
```

1.2.3 Balises ouvrantes et balises fermantes

Les balises vont par pair : une balise *ouvrante* et une balise *fermante*. Les balises fermantes s'écrivent avec un slash (/). Voici les balises ouvrantes et les balises fermantes de l'exemple ci-dessus :

Balises ouvrantes : <html>, <head>, <title>, <body>, <h1>, ,

Balises fermantes : </title>, </head>, </h1>, , , </body>, </html>

Le texte écrit entre la balise ouvrante et la balise fermante est le *contenu* de la balise. Dans l'exemple suivant

Mon premier paragraphe

le terme *paragraphe* est le contenu de la balise <*b*>.
Sur la figure 1.1 voici le contenu de certaines balises :

– contenu de la balise <head> : <title>Le titre de la page</title>
– contenu de la balise <h1> : Mon premier titre</h1>
– contenu de la balise <body> : <h1>Mon premier titre</h1> Mon premier paragraphe
Les balises agissent sur leur contenu. Il existe de très nombreuses balises. Pour l'instant, nous allons uniquement décrire la signification des balises utilisées dans l'exemple de la figure 1.1.

Structure de base

La structure de base de tout document HTML est la suivante :

```
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

L'entier d'un document HTML est contenu entre les balises *<html>* et *</html>*. Entre les balises *<head>* et *</head>* se trouvent des informations supplémentaires, qui n'apparaissent pas directement sur la page, comme par exemple les styles utilisés sur la page, l'encodage des caractères, le nom donné à la page, ou encore le code des fonctions qui seront utilisées sur la page.
Les informations qui apparaîtront directement sur la page sont écrites entre les balises *<body>* et *</body>*

<title>

La balise <title>, inscrite dans la partie *head* correspond en fait au texte qui sera affiché comme titre de l'onglet (voir figure 1.2). Le contenu de cette balise ne correspond pas à un titre qui apparaîtrait directement sur la page.

<h1>

La balise <h1> signifie *heading 1*. Cette balise permet de faire des titres, à l'intérieur de la page. Les caractères entre <h1> et </h1> sont donc affichés avec une taille de police plus grande que les autres caractères de la page.
Il est également possible de définir des sous-titres, des titres de section, etc... en utilisant les balises suivantes : <h1>, <h2>, <h3>, <h4>, <h5>, <h6>. <h1> sera utilisée pour les titres principaux, <h2> pour les sous-titres, <h3> pour les titres de section, etc...

La balise permet de définir les paragraphes

La balise (abréviation pour *bold*) permet de mettre du texte en gras. Pour mettre du texte en italique, on utilise <i> (pour *italic*) ou la balise (pour *emphasis*).

1.2.4 Imbriquer des balises

Comment mettre à la fois un texte en gras et en italique ? On peut imbriquer les balises. Par exemple, pour écrire

Lausanne est une ***très*** belle ville

on écrira le code HTML suivant :

Lausanne est une très belle ville

Remarque : il serait faux d'écrire le code HTML suivant :

Lausanne est une très belle ville

En effet, dans cet exemple, l'imbrication des balises n'est pas correcte, car la balise *i*, qui est ouverte à l'intérieur de *b*, se ferme à l'extérieur de *b*.
Ce problème est similaire à celui des parenthèses en mathématiques. Assimilons un instant la balise *b* au crochet - correspond à [, correspond à] - et la balise *em* à la parenthèse - correspond à (et correspond à) - il est correct d'écrire

$$2 \cdot [3 - (2x + 3)]$$

tandis que l'expression n'est pas correcte :

$$2 \cdot [3 - (2x + 3])$$

1.2.5 Attributs des balises

Pour expliquer le rôle d'un attribut d'une balise, prenons l'exemple suivant :

```
Magnifique hôtel
<hr width='150px' align='left'/>
Vue sur le lac
```

Cet exemple donne le résultat présenté sur la figure 1.3

FIGURE 1.3 – Une ligne horizontale (<hr>), d'une largeur de 150 pixels, alignée à gauche

La balise <hr/> permet d'afficher des lignes horizontales. Par défaut, la ligne sera affichée sur toute la largeur de la page. L'attribut width='150px' permet de définir la largeur de la ligne à 150 pixels..
Voici quelques caractéristiques d'un attribut :
- Pour affecter une valeur à un attribut, on utilise le signe = (égal)
- Les attributs s'insèrent après l'élément de la balise et avant le chevron (>). Les attributs ne se placent que dans la balise d'ouverture, jamais dans celle de fermeture. Quand il y a plusieurs attributs, ils sont placés les uns à la suite des autres, dans un ordre quelconque et séparés par des espaces.
- La valeur de l'attribut est entourée par des guillemets (width="150px") ou des apostrophes (width='150px')

1.2.6 Les liens

La balise <a>, avec son attribut href permet de créer un hyperlien vers une autre page. Voici un exemple :

```
<a href="http://www.google.ch">GOOGLE</a>
```

Quelques explications sur cet exemple :
– Dans cet exemple, le terme GOOGLE apparaît sur la page. Lorsque l'utilisateur clique sur le lien, la page http://www.google.ch apparaîtra.
– Un lien, nommé GOOGLE, et pointant vers http://www.google.ch est créé.
– *href* est l'attribut de la balise a
– la valeur de l'attribut href donne l'adresse vers laquelle l'utilisateur sera dirigé
– Le texte qui est placé entre la balise *<a...>* et ** apparaît sur la page.

1.2.7 Les images

La balise *<img src='...'>* permet d'insérer une image dans un document HTML. Cette balise possède un attribut nommé *src* qui indique où prendre l'image à afficher.
Par exemple, l'instruction suivante

```
<img src='mon_img.jpg'/>
```

affiche l'image nommée *mon_ img.jpg* sur la page.

1.2.8 Combiner les balises <img> et <a>

Nous avons vu que la balise <a> (et son attribut href) permettent de mettre un texte en évidence (lien) et de provoquer le passage à une autre page lorsque l'on clique sur ce lien. En plaçant une balise d'image à la place du texte à mettre en évidence, nous pouvons faire en sorte que l'image devienne un lien.
Par exemple les instructions suivantes

```
<a href='http://www.google.ch'><img src='mon_img.jpg'/></a>
```

afficheront l'image dont la source est *mon_ img.jpg*. Lorsque l'utilisateur clique sur l'image, la page *http://www.google.ch* s'affichera

1.2.9 Les tableaux

Les tableaux sont créés à l'aide des balises suivantes :
– <table> : définit le tableau
– <tr> : définit une ligne dans le tableau (table row)
– <td> : définit une cellule dans le tableau (table data)
Par exemple, le code suivant produit une table de 2 lignes et 3 colonnes (voir 1.4)

```
<table border="1">
    <tr>
        <td>aaa</td>
        <td>b</td>
        <td>ccc ccc</td>
    </tr>
    <tr>
        <td>abcde</td>
```

```
        <td>b</td>
        <td>ccc</td>
    </tr>
</table>
```

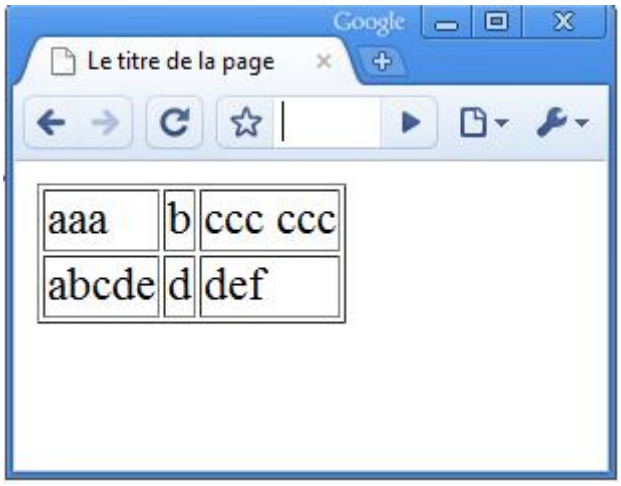


FIGURE 1.4 – Un tableau

1.2.10 Balises sans contenu

Certaines balises ont un contenu (par exemple une belle ville, le contenu de ** étant *belle*), d'autres n'ont pas de contenu (par exemple <hr>, ou encore <img src='...'>). Pour ces balises sans contenu, plutôt que d'écrire la balise ouvrante et la balise fermante (<hr></hr>), on *fusionne* ces deux balises, pour n'écrire qu'une balise : <hr/>, ou (où le slash se situe à la fin de la balise)

1.2.11 La balise

La balise *span* n'a aucun effet ! Par exemple, écrire

```
Un peu de texte
```

ou

```
Un <span> peu de </span> texte
```

produira exactement le même résultat. Cependant, une telle balise est utile pour délimiter une portion du texte, à savoir le texte présent entre la balise ouvrante ** et la balise fermante **. Dans l'exemple ci-dessus, il s'agit des termes *peu de*.
Nous verrons dans la section suivante (1.3), ainsi que dans le chapitre 3 qu'il est parfois utile de pouvoir délimiter une certaine portion du texte, soit pour la mettre en forme, soit pour la modifier.

1.2.12 Remarques

Avec les navigateurs actuels (Internet explorer, Firefox, Chrome, etc...), même lorsque le code HTML contient certaines erreurs, la page s'affichera malgré tout correctement. Cependant, certains supports n'ont pas les ressources nécessaires pour interpréter du code HTML contenant des erreurs. Il s'agit par exemple des pages destinées à être vues sur un téléphone portable.
Voici donc certaines règles qu'il convient de respecter, même si les navigateurs actuels afficheront correctement le résultat :

– l'imbrication des balises doit être correct : *Une belle ville*, et pas *Une belle ville*
– les balises doivent toujours être fermées : ** et pas *<img src='monImg.jpg'>*
– les noms des balises et leurs attributs s'écrivent en minuscule : ** et pas *IMG SRC='monImg.jpg'/>*
– les valeurs des attributs sont toujours encadrées par des guillemets ou des apostrophes : *img src='monImg.jpg'* et pas *img src=monImg.jpg*

1.2.13 Compléments sur HTML

Lors de la création d'un page Internet en HTML, quelques règles s'imposent quand à la présentation du code HTML et à la marche à suivre pour créer cette page [1].

Marche à suivre

– la structure de la page doit être définie AVANT de commencer à écrire du code. Cela ne veut pas dire que l'on ne peut pas changer cette structure par la suite pour y apporter des améliorations.
– avant de mettre du texte et des images sur la page, commencer par créer un squelette, c'est-à-dire par écrire les tags HTML sans texte ainsi que les informations de taille pour que cela ressemble le plus possible à la structure de la page finale.
Par exemple :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
  <head>

    <title></title>

    <meta http-equiv="Content-Language" content="fr" />
    <meta http-equiv="content-type" content="text/html;charset=UTF-8" />

  </head>

  <body>

    <table width = "800px">
      <tr>
        <td width = "50%">
        </td>
        <td>
        </td>
      </tr>
    </table>

    ...
  </body>
</html>
```

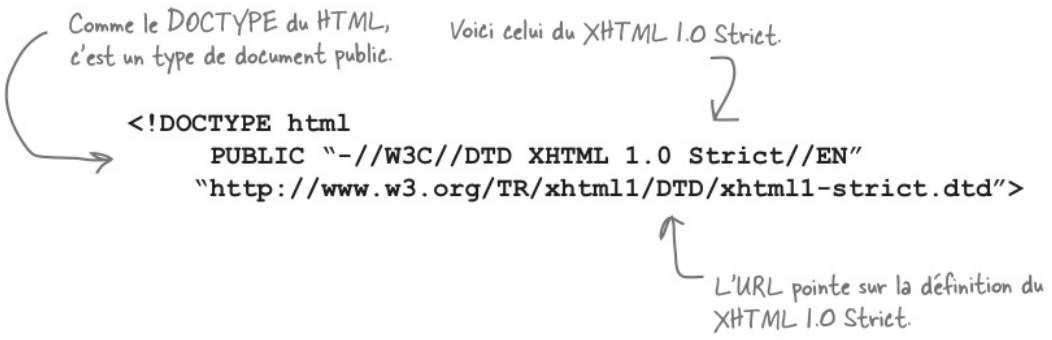
– le code doit être présenté de manière lisible. Une bonne habitude à prendre est d'indenter, c'est-à-dire décaler, le code comme ci-dessus. Ainsi les balises ouvrantes et fermantes se trouvent alignées dans une même colonne ce qui rend le code plus lisible et donc permet de trouver plus facilement les erreurs. Cela devient particulièrement important lorsque vous avez des milliers de lignes de code et que plusieurs personnes travaillent sur un même projet !
– une fois la structure de la page définie, introduire le texte et les images.
– le style vient en dernier. Les couleurs, les types de polices de caractère, leur taille, etc se met tout à la fin. Nous verrons d'ailleurs qu'avec CSS, le style est en général stocké dans un fichier séparé qu'on ne rempli qu'une fois le fichier html terminé.
– par convention, les noms de fichiers contenants du code html se terminent par l'extension .html. Par exemple : index.html.
– Stocker les fichiers dans un répertoire dont le nom décrit le site sur lequel vous travaillez. Par exemple pour l'exercice HTML sur le site du CERN :
`CERN_html/`
Il est courant de stocker les images dans un répertoire séparé, souvent nommé images :

1. Certains éléments de ce document sont tirés du livre :HTML avec CSS et XHTML, Tête la première, Eric Freeman et Elisabeth Freeman, ISBN : 978-2-8150-0005-5

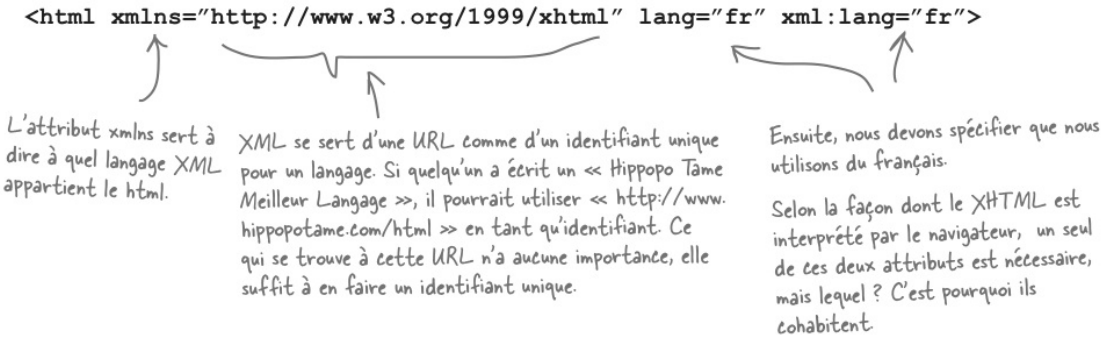
`CERN_html/images/`
Ainsi, quand vous utilisez une balise image, il vous faudra écrire, par exemple :
`<img src="images/nom_image.png">`
Nous verrons plus tard les feuilles de style (CSS). Il est courant de stocker ces fichiers dans un répertoire css :
`CERN_html/css/`

Le type de document

Nous avons vu mais peu expliqué la notion de DOCTYPE.
Le langage HTML a évolué au court du temps. Cela a amené à différentes versions du langage avec différentes règles d'écriture pour chacune de ces versions.
La dernière version est appelée XHTML 1.0 Strict. C'est la version qui fera référence dans ce cours. Vous pouvez trouver les différentes versions ainsi que la ligne DOCTYPE correspondant sur ce site. La ligne DOCTYPE pour XHTML 1.0 Strict est :



Il vous faudra également définir la balise HTML de la manière suivante :



Pourquoi utiliser cette version ? Il est vraisemblable que dans ce cas vos pages web fonctionnent de la même façon avec la plupart des navigateurs, ce qui donne à vos visiteurs une meilleure impression. Il y a aussi d'autres avantages : les pages conformes à cette version d'HTML se chargent plus vite et marchent mieux avec les autres types d'appareils dont on se sert pour surfer sur le Web (comme les TV et les téléphones). Elles offrent aussi une meilleure accessibilité aux mal-voyants qui se servent de lecteurs d'écrans vocaux.
Quelques règles pour qu'un document respecte la version XHTML 1.0 Strict :

1. Chaque page doit commencer par un DOCTYPE, ensuite, l'élément <html> doit toujours se trouver au début de la page. Après le DOCTYPE vient la balise <html> et enfin la balise </html> termine la page. Le reste du HTML est imbriqué à l'intérieur.
2. Seuls les éléments <head> et <body> peuvent se mettre directement dans l'élément <html>. Cela signifie que tous les autres éléments doivent sans exception être insérés, soit dans <head>, soit dans <body>.
3. Donnez toujours à l'élément <head> un élément <title>. C'est la loi. Ne pas la respecter donnera du HTML non conforme. Le seul endroit où insérer les éléments <title>, <meta> et

<style> est dans l'élément <head>.

4. L'élément <body> n'accepte que des éléments de bloc (<h1>, <h2>, ..., <h6>, <p>, <blockquote>, etc.). L'ensemble des éléments en-ligne et des textes doit être intégré à un élément de bloc avant de se trouver dans l'élément <body>.
5. les paragraphes servent aux textes, ils ne contiennent jamais d'éléments de bloc. Ils acceptent parfaitement n'importe quel élément en-ligne comme (, <a>, , <img>, <q>, etc).
6. La seule chose qui peut figurer dans un élément en-ligne est un texte ou un autre élément en-ligne. Les éléments de bloc n'ont en aucun cas droit de cité ici.
7. On peut imbriquer n'importe quel élément en-ligne dans un autre, cependant dans certains cas cela n'a aucun sens. N'imbriquez jamais un élément <a> dans un autre <a> car c'est très déroutant pour vos visiteurs. Les éléments vides comme <img> ne donnent aucun moyen d'y imbriquer un autre élément en-ligne.
8. Les balises vides se ferment toujours avec "/>". Par exemple :

   ```
   <img ... />
   <hr/>
   <br/>
   <meta ... />
   ```

   etc

Il existe un outil (en anglais) permettant de vérifier qu'un document HTML suit bien les règles de la version utilisée :
http://validator.w3.org
A partir ce maintenant, vos documents HTML devront suivre la version XHTML 1.0 Strict et vous devrez vérifier que c'est bien le cas avec cet outil.
La version Transitional peut être utile dans le cas de la transformation d'un site d'une ancienne version vers la version Strict :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

On pourra par exemple avec cette version garder les balises qui définissent le style alors que dans la version strict, le style doit être défini exclusivement à l'aide de CSS. C'est le type de document utilisé pour l'exercice HTML de la page du CERN.
Pour l'exercice CSS de la page du CERN, la version Strict sera utilisée.

1.3 CSS

1.3.1 Introduction : qu'est-ce que CSS ?

Cette section présente le langage CSS. CSS vous permet de définir l'apparence des textes (comme la police, la couleur, la taille, etc...), ainsi que l'agencement de la page (comme les marges, l'arrière-plan, etc...). CSS définit donc la présentation du document.
CSS est l'abréviation de Cascading Style Sheets. Un style définit la façon dont un élément HTML (par exemple <h1>) sera affiché.
Ces styles peuvent être définis dans une feuille de style externe (un fichier .css). Une feuille de style peut être utilisée pour définir la présentation de plusieurs documents HTML, ce qui permet de gagner beaucoup de temps.
HTML a été conçu pour définir la structure d'un document pas sa présentation. Par conséquent tout ce qui est lié à la présentation d'un document devrait être défini à l'aide de CSS. Typiquement, il faut préférer CSS à l'utilisation de balises HTML permettant de définir la présentation d'un document (comme par exemple <font color='...'>)

1.3.2 La syntaxe

La syntaxe de base de CSS est composée de 3 parties :
– un *sélecteur*
– une *propriété*
– une *valeur*
Ces trois parties sont écrites de la façon suivante :

```
sélecteur {propriété:valeur}
```

Un *sélecteur* correspond à une balise HTML (<p>, <h1>, etc...) et la propriété est un attribut dont on veut changer la valeur.
Exemple :

```
h1 {font-size:100px}
```

Dans cet exemple, tous les titres principaux (<h1>) du document auront une taille de 100 pixels. Si la valeur d'un attribut contient un espace, alors la valeur de l'attribut s'écrit en guillemets :

```
h2 {font-family:"sans serif"}
```

Il est possible de définir plusieurs attributs pour un même sélecteur. Dans ce cas, chaque propriété sera séparée par un point-virgule :

```
p {font-family:"sans serif"; font-size: 90%; color: blue}
```

En écrivant une propriété par ligne, la lisibilité est meilleure :

```
p
{
  font-family:"sans serif";
  font-size: 90%;
  color: blue;
}
```

1.3.3 Grouper les sélecteurs

Si certaines propriétés s'appliquent à plusieurs sélecteurs, il est possible de les grouper. Ainsi, plutôt que d'écrire

```
h1
{
font-family:"sans serif";
color: blue;
}
h2
{
font-family:"sans serif";
color: blue;
}
```

Il est possible, et plus rapide, de grouper les sélecteurs h1 et h2 et d'écrire :

```
h1,h2
{
font-family:"sans serif";
color: blue;
}
```

1.3.4 Le sélecteur de classe

Avec le sélecteur de classe il est possible d'appliquer des styles différents pour des éléments HTML d'un même type. Par exemple, si nous souhaitons que certains paragraphes (<p>) soient affichés en rouge (disons, pour mettre en évidence des paragraphes importants), et que d'autres paragraphes aient un alignement centré, nous pouvons définir deux classes pour chacun de types de paragraphe :

```
p.introduction{text-align:center}
p.important{color:red}
```

Dans le document HTML, l'attribut class permet de choisir la classe qui sera utilisée :

```
<p class="introduction">Un paragraphe d'introduction</p>
<p class="important">Un paragraphe important</p>
```

Pour définir une classe qui s'applique à tous les éléments HTML, et pas uniquement à un seul élément HTML (comme <p> dans l'exemple ci-dessus), on utilise la syntaxe suivante : l'élément HTML n'est pas écrit ; on commence directement par un point, suivi du nom de la classe.
Exemple :

```
.important{color:red}
```

Ainsi, la classe important pourra être utilisée pour tous les éléments HTML :

```
<h1 class="important">Un titre important</h1>
<p class="important">Un paragraphe important</p>
```

1.3.5 Lier CSS à un document HTML

Pour intégrer du CSS dans un document HTML, il existe trois possibilités :
– une feuille de style *externe*
– une feuille de style *interne*
– un style défini directement dans la balise (*inline style*)

Les feuilles de style externes

Les feuilles de style externes sont idéales lorsque les même styles sont appliqués sur plusieurs documents HTML. La balise <link>, à écrire dans la partie <head> du document HTML permet de lier une feuille de style à un document HTML :

```
<head>
<link rel="stylesheet" type="text/css" href="mon_style.css" />
</head>
```

Généralement, les fichiers contenant les styles ont l'extension .css

Les feuilles de style internes

Si le style est appliqué à un seul document HTML, il est possible de définir le style, directement dans la partie <head> du document HTML, à l'aide de la balise <*style*> :

```
<head>
  <style type="text/css">
    h1{font-size:110%}
    .avertissement{color:red}
  </style>
</head>
```

Inline Styles

Il est possible ne n'appliquer un style qu'à une seule balise. Cette façon de faire est un peu contraire à l'idée du CSS. En effet, en procédant ainsi, il n'y a plus de séparation entre la structure d'un document et sa présentation. Cette façon de faire est donc à utiliser avec précaution.

```
<img src='bille1.gif'
     style='position:absolute; top:50px; left:60px'>
```

1.3.6 Héritage des propriétés

Pour un même sélecteur, il est possible de définir plusieurs propriétés, soit de façon inline, soit dans plusieurs feuilles de style, externes ou internes. Cet élément héritera de l'ensemble des propriétés définies dans les différents styles (d'où l'appellation Cascading Style Sheet).
Par exemple, si on trouve, dans un fichier .css externe la propriété suivante

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

et, à l'intérieur d'un style interne (défini à l'aide d'une balise <style>) la propriété suivante

```
p{color:red;}
```

alors, l'ensemble des paragraphes du document auront les deux propriétés : les caractères auront une taille de 20 pixels et seront de couleur rouge

1.3.7 Texte et police de caractères

Alignement

La propriété *text-align* permet l'alignement horizontal des textes ; elle peut prendre les valeurs suivantes :
– center : le texte est centré
– left : le texte est aligné à gauche ; il s'agit de la valeur par défaut
– right : le texte est aligné à droite
– justify : le texte est justifié : on modifie donc l'espace entre les caractères pour que le texte soit aligné à la fois à gauche et à droite
Par exemple, pour *justifier* les paragraphes, on écrira l'instruction CSS suivante :

```
p{text-align:justify}
```

Police de caractères

La propriété *font-family* permet de spécifier la police de caractères :

```
h2{font-family:Arial,Verdana,Sans-serif}
```

Plusieurs polices (séparée par des virgules) peuvent être spécifiées. Ainsi, si le navigateur ne parvient pas à afficher la première police (Arial dans l'exemple ci-dessus), alors il tentera d'afficher la seconde police de la liste (Verdana dans notre exemple). S'il ne parvient pas non plus à afficher la police Verdana, alors il affichera une autre police, de la même famille de police (ici : sans-serif). Il existe trois familles de police :
– Serif : police avec de petites extensions (empattement) qui forment la terminaison des caractères
– Sans-serif : police sans empattement
– Monospace : police où tous les caractères ont la même largeur
Les exemples ci-dessous montrent la différence entre une police avec et une police sans serif. La police Helvetica est une police sans serif tandis que la police Times est une police serif :

FIGURE 1.5 – Un exemple d'écriture sans serif

FIGURE 1.6 – Un exemple d'écriture serif

Tailles de police

La propriété *font-size* permet de spécifier la taille d'un texte :

```
h1{font-size:40px}
```

Ainsi, les titres auront une taille de 40 pixels. Au lieu de spécifier la taille d'un texte à l'aide de px, correspondant à un nombre de pixels, on peut également utiliser l'unité em [2]. 1em vaut la taille de police par défaut dans un navigateur (typiquement 1em=16pixels). Par exemple, si nous décidons que les titres principaux valent une fois et demi la taille standard des caractères nous donnerons la valeur 1.5em à la propriété font-size :

```
h1{font-size:1.5em}
```

Couleur

La propriété *color* permet de spécifier les couleurs. On peut définir une couleur de plusieurs manières :
- par son nom (par exemple *h1{color :red}*)
- par sa valeur RGB décimale : R indique la quantité de rouge, G indique la quantité de vert (green) et B indique la quantité de bleu. Les valeurs sont comprises entre 0 et 255. Par exemple *h1{color :rgb(255,0,0)}* correspond à la couleur rouge
- par sa valeur RGB hexadécimale, où les quantités de rouge, vert et bleu prennent des valeurs entre 00 et FF [3] : *h1{color :#FF0000}*

1.3.8 Afficher et masquer des éléments

Il est parfois pratique de masquer certaines parties d'une page web. Typiquement, en fonction de choix effectué par l'utilisateur, nous allons afficher certaines informations et en cacher d'autres. Lors de notre étude de Javascript, nous verrons comment, dynamiquement (par exemple lorsque l'utilisateur clique sur un bouton), masquer certaines informations et en afficher d'autres. Pour ce faire, nous allons modifier les valeurs des propriétés display et visibility de certains éléments.
La propriété *visibility* prend généralement [4] les valeurs *visible* (l'élément est visible) ou *hidden* (l'élément est caché) :

```
img{visibility:hidden}
```

cache les images du document : le reste des éléments de la page ne changent toutefois pas de place : une zone vide (i.e. correspondant à la couleur d'arrière-plan de la page) apparaît.
Pour masquer un élément, et sans laisser de zones vides sur la page, on utilise la propriété *display*, qui prend des valeurs [5] comme *none*, *block* ou *inline* : *none* masque l'élément ainsi que la place

2. il est également possible de spécifier la taille relative des caractères en pour cent comme h1{font-size :150%}
3. la base du système hexadécimal est 16. Voici les premières valeurs hexadécimale : 0,1,2,3,4,5,6,7,8,9,10,A,B,C,D,E,F,11,12,13,14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21, etc...
4. les autres valeurs possible étant *collapse* (uniquement pour les tableaux, supprime une ligne ou une colonne), ou *inherit* (la visibilité est la même que pour l'élément parent)
5. de nombreuses autres valeurs peuvent également être prises par *display*, voir la documentation en ligne à ce sujet, comme par exemple www.w3schools.com/css

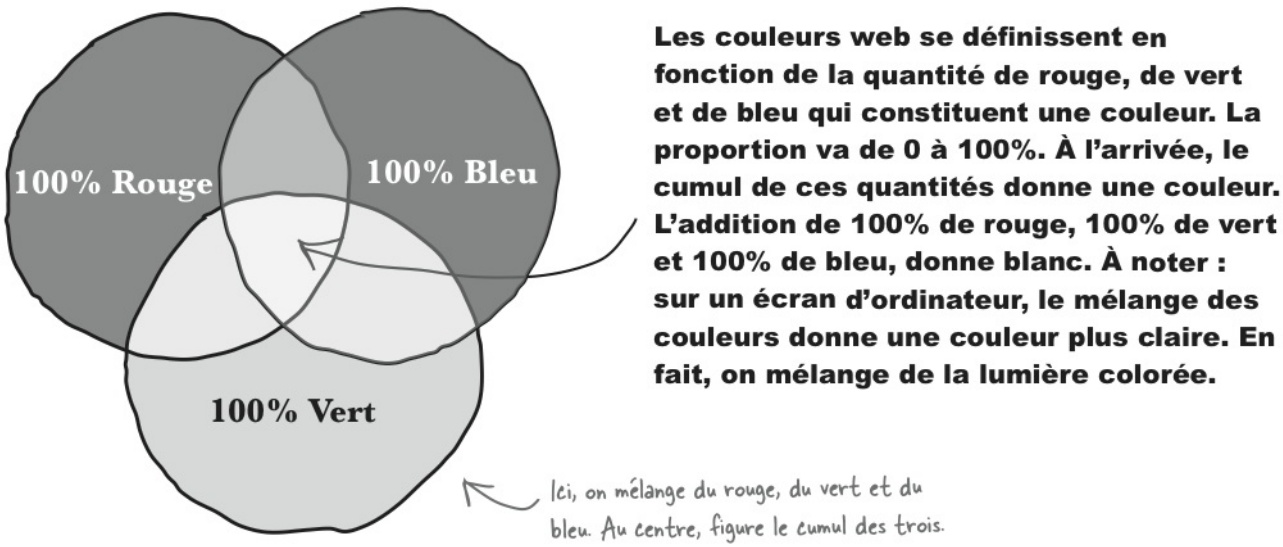
qu'il occupe (le résultat est donc le même que si l'élément n'existait pas), *block* affiche l'élément et ajoute un retour de ligne avant et après l'élément, *inline* affiche simplement l'élément, sans ajouter de retours de ligne.

1.3.9 Compléments sur CSS

Généralités

On définit les styles dans un ou plusieurs fichiers séparés, stockés dans un répertoire nomme css. Ces fichiers se terminent avec l'extension .css : style.css
Ce fichier doit être inclus dans le fichier html de la manière suivante entre les balises <head></head> :
`<link href="css/style.css" rel="stylesheet" type="text/css" media="screen" />`
Vous remarquerez l'attribut media. Il permet de définir différentes feuilles de style suivant le type de d'appareil utilisé :
- screen : pour un écran
- print : pour une imprimante
- hanheld : pour un téléphone portable
- etc

Les couleurs



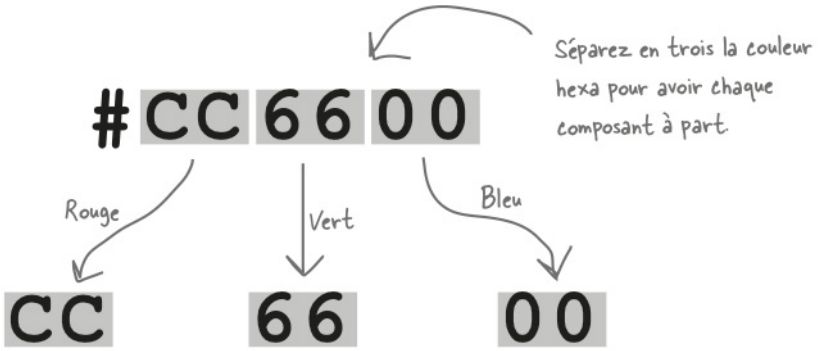
Pour décrire directement une couleur, utilisez son nom. L'ennui est que seules dix-sept d'entre elles peuvent être nommées. Certains navigateurs Internet permettent d'utiliser d'autres noms de couleurs tels que DarkBlue ou WhiteSmoke. Cependant, vous n'avez pas la garantie que ces couleurs s'afficheront correctement et surtout, elles ne respectent pas la norme CSS 2.1 !
Une couleur se définit aussi par sa composition de rouge, de vert et de bleu. Pour spécifier orange, nous sommes revenus en arrière de quelques pages pour trouver les valeurs suivantes : 80% de rouge, 40% de vert et 0% de bleu.

```
body {
background-color: rgb(80%, 40%, 0%);
}
```

Il est possible d'indiquer les valeurs de rouge, vert et bleu en valeur absolue avec des chiffres entre 0 et 255. Dans ce cas, 80% de rouge, 40% de vert et 0% de bleu deviennent 204 (rouge), 102 (vert) et 0 (bleu).

```
body {
background-color: rgb(204, 102, 0);
}
```

On peut également décrire une couleur en utilisant sont code hexadécimal. C'est la manière la plus courante de le faire.



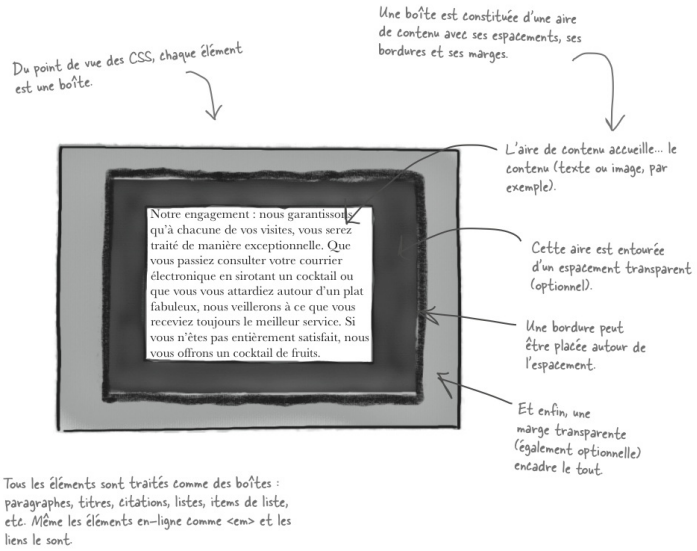
Nous reviendrons sur l'hexadécimal et son fonctionnement plus tard dans le cours.
Vous trouverez sur http://code-couleur.com/ un outil vous permettant de choisir votre couleur et d'obtenir son code hexadécimal.
Vous trouverez sur http://www.html-color-names.com/color-chart.php le code hexadécimal de certaines couleurs telles que DarkBlue, WhiteSmoke, etc.

Différence entre class et id

Servez-vous de l'attribut class pour styler un groupe d'éléments, c'est à dire pour appliquer un même style à plusieurs éléments d'une page.
L'attribut id permet de donner un nom unique à un élément. Il sert aussi à donner un style spécifique à un élément. Il ne doit y avoir qu'un seul élément par page portant le même id.

Balise div, marges et espacement

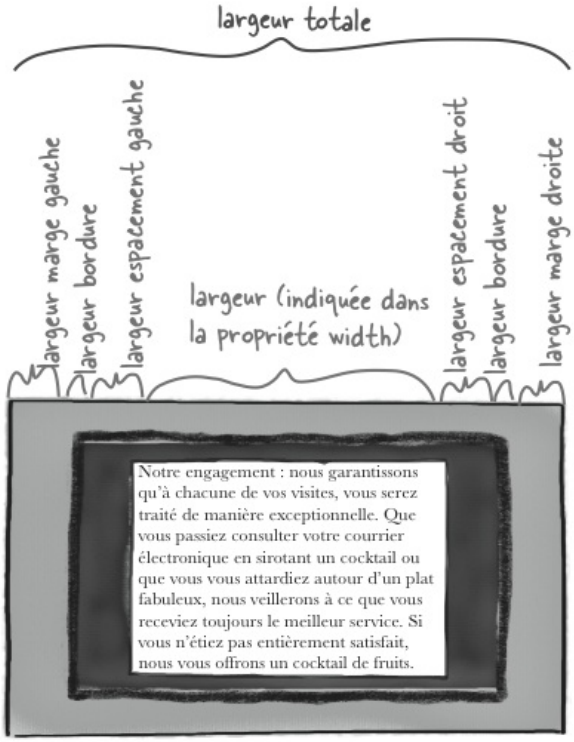
Un div est comme un conteneur dans lequel on met des éléments pour les garder ensemble. Cela permet, par exemple, d'appliquer un même style à un groupe d'éléments ou encore de placer ces différents éléments ensemble sur une page.



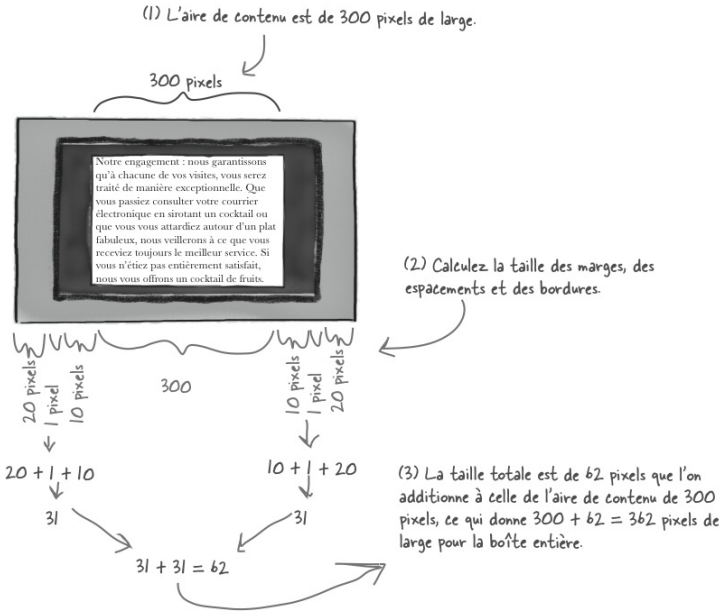
Pour ajouter de l'espacement, on utilise la propriété padding.

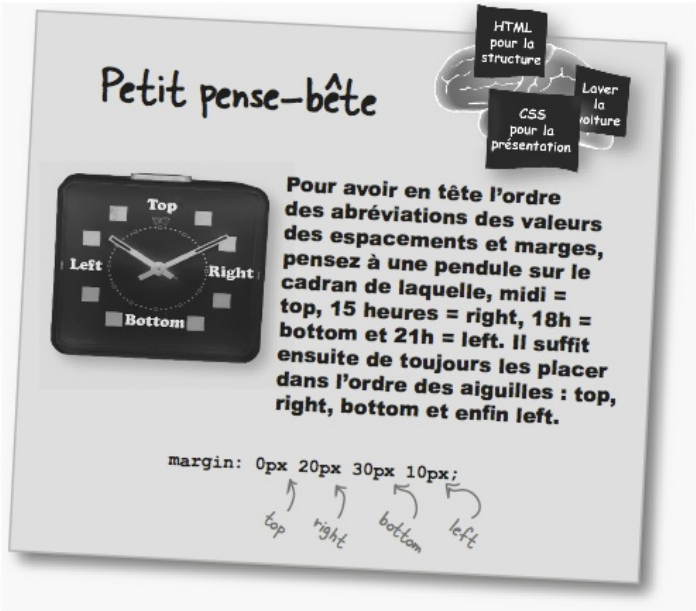
Pour ajouter une marge, on utilise la propriété margin.

Pour avoir une idée de la largeur de toute la boîte, il faut additionner la largeur de l'aire de contenu à celles des marges de gauche et de droite plus celle de la bordure qu'il faudra compter deux fois puisqu'il y en a une à gauche et une autre à droite.



Voici un exemple :



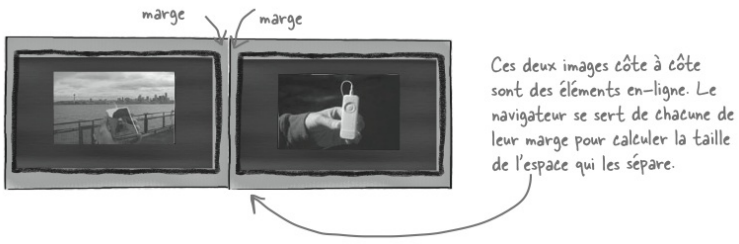


Interlignes

La propriété line-height permet de déterminer la taille de l'espace vertical situé entre deux lignes de texte. Comme les autres propriétés qui s'appliquent aux polices, on peut l'exprimer en pixels, en em ou en pourcentage, ces deux derniers étant relatifs à la taille de la police.

Le flux : placement des éléments sur une page

Le navigateur utilise le flux pour effectuer la mise en pages des éléments XHTML. Celui-ci commence par le début du fichier XHTML et suit le flux des éléments de haut en bas en affichant au fur et à mesure chacun des éléments qu'il rencontre.
Pour les éléments de bloc, il place un saut de ligne entre chaque. Le premier élément d'un fichier est le premier à s'afficher, un saut de ligne lui succède, puis le second élément s'affiche, suivi d'un saut de ligne, et ainsi de suite de bas en haut du fichier. C'est le flux.
Les éléments en lignes se placent les uns à côté des autres depuis le coin en haut à gauche jusqu'à celui en bas à droite.
Quand le navigateur doit placer côte à côte des éléments en-ligne ayant chacun une marge, il fait son travail normalement. Il met suffisamment d'espace entre les deux pour que les deux marges soient prises en compte. Si la marge de l'élément de gauche mesure 10 pixels et celle de l'élément de droite 20 pixels, l'espace entre les deux éléments mesurera bien 30 pixels.



Dans ce cas, c'est différent. Quand le navigateur place l'un au-dessus de l'autre, deux éléments de bloc, il fusionne leurs marges communes. La hauteur de la marge après fusion est celle de la marge la plus importante.

Pour faire flotter un élément :
1- Donner un identifiant unique à l'élément que vous voulez faire flotter, par exemple incroyable
2- Donner une largeur à cet élément, par exemple 200px
3- Ajouter la propriété float.

```
# id_flot {
width: 200px;
float: right;
}
```

Cela donne le résultat suivant :

Faites-le flotter

Ajoutons la propriété **float**. Celle-ci peut être mise à gauche ou bien à droite, nous retenons ce choix :

```
#incroyable {
   width: 200px;
   float: right;
}
```

Voyons maintenant comment ce paragraphe et ses voisins sont gérés par le navigateur.