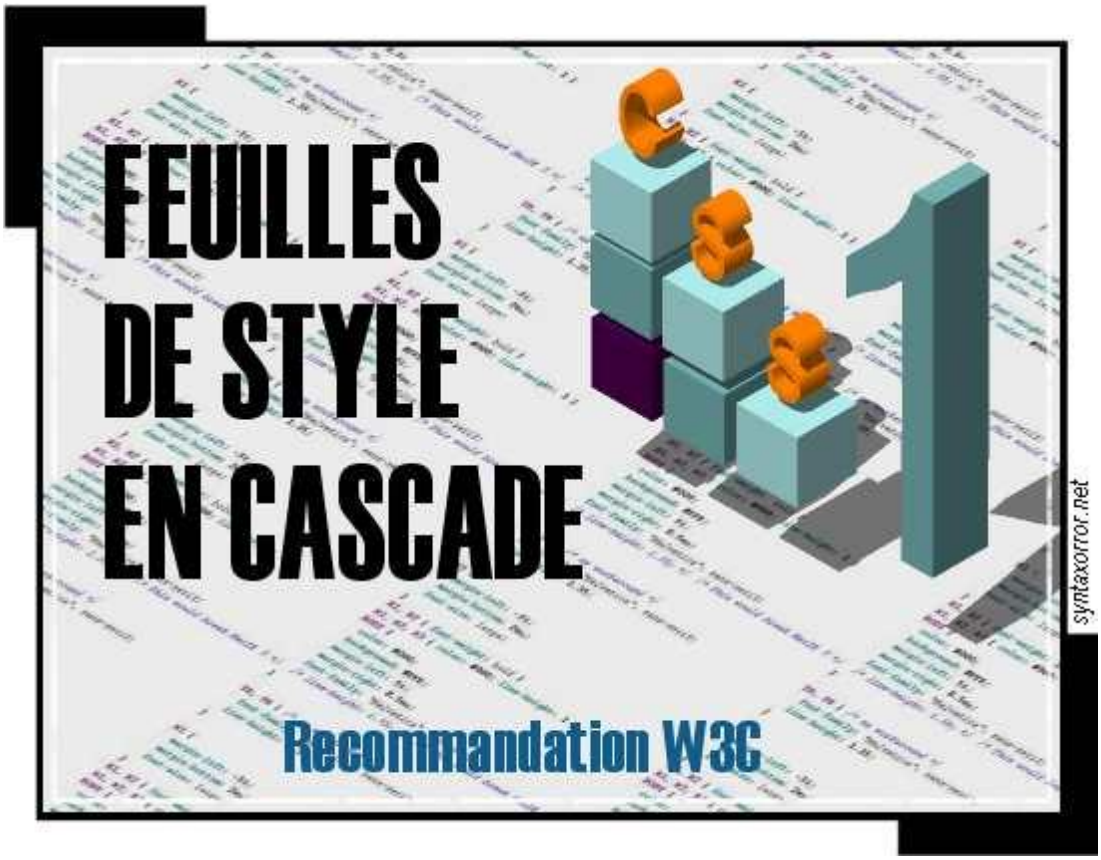


REC−CSS1−19990111

SYNTAXORROR.NET

Traduction:

J.J SOLARI

PDF:

G.ACCAD

*Logiciels utilisés**

HTMLDoc
Strata 3D
Code Edit

*gratuiciels

Table des matières

Table des matières

Recommandation CSS1 du W3C en version française

Statut du document traduit

Ceci est une traduction de la Recommandation du W3C portant sur le premier niveau de l'implémentation des feuilles de style pour le Web (Cascading Style Sheets, level 1).

Cependant ce n'est pas la version officielle en français de la Recommandation. Seul le document original en anglais a valeur de norme. On peut l'obtenir à : http://www.w3.org/TR/1999/REC−CSS1−19990111.

Avertissement

Des erreurs ont pu survenir malgré le soin apporté à ce travail.

Notes sur la traduction

Certains concepts importants sont difficiles à rendre en français. Aussi les expressions en anglais viennent parfois en renfort dans le texte sous cette forme :
ex. traduction [ndt. translation]

L'appendice F répertorie l'ensemble des corrections survenues depuis la publication originale de la Recommandation. Cette partie n'a pas été traduite. Cependant, cette version française intègre toutes les corrections et modifications survenues depuis la révision de 1999. Elle est donc à jour en date du 3 août 2001 (voir la liste des dernières modifications à l'adresse
http://www.w3.org/Style/css1−updates/REC−CSS1−19990111−errata.html)

- Adresse : <http://www.yoyodesign.org/doc/w3c/css1/>
- Date de traduction : 3 août 2001
- Traducteur : J.J.SOLARI <jjsolari@pobox.com>
- Remerciements à : Claude CHAUNIER, Éric SCHREINER et Gilles ACCAD pour leur travail de relecture et leurs suggestions.
- Dernières corrections apportées :
    - ♦ 21 novembre 2001 (erreurs typographiques, orthographiques et grammaticales).

Autres documents traduits

On peut consulter les traductions en français d'autres documents du W3C à
http://www.w3.org/Consortium/Translation/French

Notice légale

CSS1 : Les feuilles de style en cascade, niveau 1

Recommandation du W3C du 17 déc. 1996, révisée le 11 jan. 1999

Cette version :	http://www.w3.org/TR/1999/REC−CSS1−19990111
Dernière version :	http://www.w3.org/TR/REC−CSS1
Version précédente :	http://www.w3.org/TR/REC−CSS1−961217
Auteurs :	Håkon Wium Lie (howcome@w3.org) Bert Bos (bert@w3.org)

Statut de ce document

Ce document est une Recommandation du W3C. Il a été vérifié par les membres du W3C (http://www.w3.org/) et l'utilisation de cette spécification a été approuvée par consensus général. Ce document est stable et peut être considéré comme matériel de référence ou cité par un autre document comme norme de référence. Le W3C encourage un large déploiement de cette Recommandation.

On peut trouver une liste des actuelles Recommandations et des autres documents techniques du W3C à l'adresse http://www.w3.org/TR/.

Ceci est une version corrigée du premier document publié le 17 décembre 1996. Les modifications apportées à la version originale sont répertoriées à l'Appendice F. Une liste des erreurs connues dans cette spécification est disponible à l'adresse http://www.w3.org/Style/CSS/Errata/REC−CSS1−19990111−errata

Résumé

Ce document spécifie le niveau 1 du mécanisme des *feuilles de style en cascade*. Le CSS1 est un mécanisme de feuille de style simple qui permet aux auteurs et aux lecteurs de lier un style (par exemple les polices de caractères, les couleurs et l'espacement) à un document HTML. Le langage CSS1 peut être lu et écrit par un humain, les styles sont exprimés dans la terminologie habituelle de la publication par ordinateur.

La cascade des feuilles de style est une des caractéristiques fondamentales des CSS ; les auteurs peuvent attacher une feuille de style préférée tandis que les lecteurs ont la possibilité de choisir une feuille de style personnelle pour pallier un un handicap physique ou technologique. Cette spécification définit les règles de résolution des conflits entre différentes feuilles de style.

Cette Recommandation est le résultat des travaux du W3C dans le domaine des feuilles de style. Pour des informations complémentaires sur les feuilles de style, voir [1].

Terminologie

agent utilisateur [ndt. User Agent (UA)]
: un agent utilisateur, souvent *un navigateur* ou *un client web*.

attribut
: attribut HTML

auteur
: l'auteur d'un document HTML

balisage fictif [ndt. fictional tag sequence]
: un moyen pour décrire le comportement des pseudo−classes et des pseudo−éléments.

canevas
: la surface dessinée par l'agent utilisateur sur laquelle un document est rendu

CSS1
: CSS niveau 1. Ce document définit le CSS1 qui est un mécanisme de feuille de style simple pour le Web.

déclaration
: une propriété (ex. 'font−size') et sa valeur correspondante (ex. '12pt').

designer
: l'auteur d'une feuille de style.

dimensions intrinsèques

la largeur et la hauteur définies par le seul élément, non imposées par son environnement. On considère dans cette spécification que tous les éléments remplacés, et seulement ceux−ci, ont des dimensions intrinsèques.

document
: un document HTML.

élément
: un élément HTML.

élément de type bloc [ndt. block−level element]
: un élément qui a une fin de ligne avant et après (ex. 'H1' en HTML)

élément de type en−ligne [ndt. inline element]
: un élément qui n'a pas de fin de ligne avant ni après (ex. 'STRONG' en HTML)

élément remplacé [ndt. replaced element]
: un élément dont l'interpréteur de CSS ne connaît que les dimensions intrinsèques. En HTML,'IMG', 'INPUT', 'TEXTAREA', 'SELECT' et 'OBJECT' sont des exemples d'éléments remplacés. Par exemple, le contenu de l'élément 'IMG' est souvent remplacé par l'image que l'attribut 'SRC' indique. CSS1 ne définit pas comment sont trouvées les dimensions intrinsèques.

élément enfant ou enfant [ndt. child element]
: un *sous−élément* dans la terminologie SGML [5]

élément parent ou parent [ndt. parent element]
: l'*élément contenant* dans la terminologie SGML [5].

extension HTML [ndt. HTML extension]
: Balisage introduit par les compagnies éditrices d'agents utilisateurs, le plus souvent pour obtenir des effets visuels. Les balises 'FONT', 'CENTER' et 'BLINK' en sont des exemples, ainsi que l'attribut 'BGCOLOR'. Un des buts des feuilles de style est de fournir une alternative aux extensions HTML.

feuille de style [ndt. style sheet]
: un ensemble de règles

feuille de style ou Cascading Style Sheets
: les feuilles de style en cascade et leur format

fonctions avancées de CSS1 [ndt. CSS1 advanced features]
: caractéristiques décrites dans cette spécification mais qui n'appartiennent pas aux fonctions de base du CSS1.

fonctions de base de CSS1 [ndt. CSS1 core features]
: la part du CSS1 requise par tous les agents utilisateurs conformes au CSS1.

HTML
: Hypertext Markup Language [2] une application de SGML.

interpréteur CSS1 [ndt. CSS1 parser]
: un agent utilisateur qui lit les feuilles de style CSS1.

lecteur
: la personne qui consulte le document.

poids
: l'importance relative d'une règle, son ordre de priorité.

propriété
: un paramètre de style sur lequel peut agir une feuille de style. Cette spécification définit une liste de propriétés et leurs valeurs associées.

pseudo−élément
: les pseudo−éléments qui sont utilisés par les sélecteurs de CSS pour agir sur des éléments typographiques (ex. la première ligne d'un élément) plutôt que sur les éléments structuraux proprement dits.

pseudo−classe
: les pseudo−classes qui sont utilisées par les sélecteurs de CSS permettent des indications en supplément de la source HTML (ex. un lien a été visité ou non) pour la classification des éléments.

règle
: une déclaration (ex. 'font−family: helvetica') et son sélecteur (ex. 'H1').

sélecteur
: une chaîne de caractères qui identifie les éléments sur lesquels s'applique la règle correspondante. Un sélecteur est ou bien simple (ex. 'H1') ou bien contextuel (fait de plusieurs sélecteurs simples, p.ex. 'H1 B').

sélecteur contextuel
: un sélecteur qui agit sur les éléments selon leur position dans la structure du document. Un sélecteur contextuel consiste en plusieurs sélecteurs simples. Par exemple, le sélecteur contextuel 'H1.initial B' est constitué des deux sélecteurs simples 'H1.initial' et 'B'.

sélecteur simple
: un sélecteur qui identifie un élément par son type et/ou son attribut et non par la position de celui−ci dans la structure du document (ex. 'H1.initial' est un sélecteur simple).

SGML
: Standard Generalized Markup Language [5], HTML en est une application.

taille ou corps de police

La taille du dessin d'une police. Typiquement, le corps d'une police est à peu près égal à la distance entre le bas de la lettre avec jambage la plus basse et le haut de la lettre avec hampe la plus haute avec éventuellement un accent diacritique.

type d'élément
: un *identifiant générique* dans la terminologie SGML [5].

utilisateur
: synonyme de *lecteur*

Tout au long de ce texte, l'utilisation de guillemets simples ('...') correspond à des extraits de code HTML ou CSS.

1 Notions de base

Créer une feuille de style simple est facile. On a besoin de connaître un peu de HTML et quelques termes de base de la publication par ordinateur. Par exemple, pour rendre bleu les éléments 'H1', on écrit :

```
H1 { color: blue }
```

L'exemple ci−dessus est une simple règle CSS. Une règle consiste en deux parties : un sélecteur ('H1') et une déclaration ('color: blue'). La déclaration comprend deux parties : la propriété ('color') et la valeur associée ('blue'). Bien que cet exemple n'agisse que sur une seule des propriétés nécessaires au rendu d'un document HTML, cela suffit pour qu'elle soit qualifiée de feuille de style. Combinée avec d'autres feuilles de style (la combinaison de feuilles de style est une fonction fondamentale des CSS), elle agira sur la présentation finale du document.

Le sélecteur est le lien entre le document HTML et la feuille de style, tous les types d'éléments HTML pouvant être des sélecteurs. Les types d'éléments HTML sont définis dans la spécification du HTML [2].

La propriété 'color' fait partie de la cinquantaine de propriétés qui déterminent la présentation d'un document HTML. La liste des propriétés et leurs valeurs possibles est définie dans cette spécification.

Les auteurs HTML ne devraient écrire des feuilles de style que pour suggérer un style particulier à leurs documents. Chaque agent utilisateur (navigateur ou client web) est pourvu d'une feuille de style par défaut qui présente les documents d'une manière (sinon banale) tout au moins raisonnable. L'appendice A contient un exemple de feuille de style pour la présentation de documents HTML tel que suggéré dans la spécification du HTML 2.0 [3].

La grammaire formelle du langage CSS1 est définie dans l'appendice B.

1.1 Incorporation dans HTML

Pour que les feuilles de style puissent avoir une influence sur la présentation, les agents utilisateurs doivent en connaître l'existence. La spécification HTML [2] définit la façon de relier HTML aux feuilles de style. Ce chapitre est donc informatif et non normatif :

```
<HTML>
  <HEAD>
    <TITLE>Titre</TITLE>
    <LINK REL=STYLESHEET TYPE="text/css"
      HREF="http://style.com/cool" TITLE="Cool">
    <STYLE TYPE="text/css">
      @import url(http://style.com/basic);
      H1 { color: blue }
    </STYLE>
  </HEAD>
  <BODY>
    <H1>Ce titre est en bleu</H1>
    <P STYLE="color: green">et le paragraphe est en vert.
  </BODY>
</HTML>
```

L'exemple montre quatre façons de combiner le style au HTML : avec l'élément 'LINK' vers une feuille de style externe, avec l'élément 'STYLE' à l'intérieur de l'élément 'HEAD', avec une feuille de style importée à l'aide de la notation CSS '@import' et enfin avec l'attribut 'STYLE' d'un élément (ici 'P') dans 'BODY'. Cette dernière option mélange style et contenu, et on perd de ce fait les avantages d'une feuille de style traditionnelle.

L'élément 'LINK' fait référence à des feuilles de style alternatives que le lecteur peut sélectionner, alors que les feuilles de style importées sont automatiquement combinées avec les autres feuilles de style.

En pratique, les agents utilisateurs ignorent les balises qui leur sont inconnues. Une conséquence, les vieux agents utilisateurs ignoreront l'élément 'STYLE', mais non son contenu qui sera traité comme faisant partie du corps du document, et qui sera rendu comme tel. Pendant une période transitoire, le contenu de l'élément 'STYLE' peut être caché à l'aide de commentaires SGML :

```
<STYLE TYPE="text/css"><!--
  H1 { color: green }
--></STYLE>
```

Comme l'élément 'STYLE' est déclaré dans le DTD en tant que "CDATA" (défini dans [2]), les interpréteurs conformes à SGML ne confondront pas la feuille de style avec un commentaire et en tiendront donc compte.

1.2 Regroupement

Pour réduire la taille des feuilles de style, on peut grouper des sélecteurs dans une liste en les séparant par des virgules :

```
H1, H2, H3 { font−family: helvetica }
```

De même, on peut grouper les déclarations :

```
H1 {
  font−weight: bold;
  font−size: 12pt;
  line−height: 14pt;
  font−family: helvetica;
  font−variant: normal;
  font−style: normal;
}
```

De plus, quelques propriétés ont leur propre syntaxe de regroupement :

```
H1 { font: bold 12pt/14pt helvetica }
```

qui équivaut à l'exemple précédent.

1.3 Héritage

Dans le premier exemple, on donnait une couleur bleu aux éléments de 'H1'. Supposons qu'il y ait un élément accentué parmi ceux compris dans l'élément 'H1' :

```
<H1>Le titre <EM>est</EM> important !</H1>
```

Si aucune couleur n'a été assignée à l'élément 'EM', le "est" accentué hérite de la couleur de l'élément parent, il apparaît donc en bleu. D'autres propriétés de style sont pareillement héritées, par exemple 'font−family' et 'font−size'.

Pour donner une propriété de style par "défaut" à un document, on peut assigner la propriété à un élément dont tous les éléments visibles descendent. Dans les documents HTML, l'élément 'BODY' peut servir à cet usage :

```
BODY {
  color: black;
  background: url(texture.gif) white;
}
```

Ceci fonctionne, même si l'auteur omet la balise 'BODY' (ce qui est autorisé), l'interpréteur HTML inférant la balise manquante. L'exemple ci−dessus donne un rendu du texte en noir et une image de fond. Le fond sera blanc si l'image n'est pas disponible (Voir chapitre 5.3 pour plus d'infos).

Quelques propriétés de style ne sont pas transmises de l'élément parent à l'élément enfant. Le plus souvent, on constate intuitivement quand ce n'est pas le cas. Par exemple, la propriété 'background' n'est pas héritée, mais le fond de l'élément parent transparaîtra par défaut.

Souvent, la valeur d'une propriété est un pourcentage de celle d'une autre propriété :

```
P { font−size: 10pt }
P { line−height: 120% }  /* relatif à 'font−size', c−à−d 12pt */
```

Pour toute propriété qui admet des valeurs en pourcentage est déterminée la propriété à laquelle elle se réfère. Les éléments enfant de 'P' hériteront de la valeur calculée de 'line−height' (c.à.d 12pt), et non du pourcentage.

1.4 Classe pour sélecteur

Pour accroître la granularité de l'action sur les éléments, un nouvel attribut a été ajouté à HTML [2]: 'CLASS'. Tous les éléments à l'intérieur d'un élément 'BODY' peuvent appartenir à une classe, celle−ci étant déclarée dans la feuille de style :

```
<HTML>
 <HEAD>
  <TITLE>Titre</TITLE>
  <STYLE TYPE="text/css">
    H1.pastoral { color: #00FF00 }
  </STYLE>
 </HEAD>
 <BODY>
  <H1 CLASS=pastoral>Beaucoup trop vert</H1>
 </BODY>
</HTML>
```

Les règles d'héritage habituelles s'appliquent aux éléments définis avec une classe ; ils héritent des valeurs de leur parent dans la structure du document.

On peut adresser tous les éléments d'une même classe sans pour autant mettre le nom de la balise dans le sélecteur :

```
.pastoral { color: green }  /* tous les éléments de la classe pastoral */
```

On ne peut spécifier qu'une seule classe par sélecteur. Ainsi 'P.pastoral.marine' est un sélecteur invalide en CSS1 (les sélecteurs contextuels décrits plus bas peuvent avoir une classe par sélecteur simple)

CSS confère tant de pouvoir à l'attribut CLASS que, dans de nombreux cas, l'élément sur laquelle porte la classe importe peu ; quel que soit l'élément, vous pouvez presque le faire se comporter comme un autre. Une telle utilisation n'est pas recommandée car on supprime un niveau de structure qui a une signification universelle (éléments HTML). Une structure basée sur CLASS n'a d'utilité que dans un domaine restreint où la signification d'une classe peut reposer sur un accord mutuel.

1.5 ID pour sélecteur

HTML [2] introduit aussi l'attribut 'ID' qui garantit le caractère d'unicité d'un objet dans un document. Il peut revêtir un intérêt particulier en tant que sélecteur dans la feuille de style, où on l'adresse en le faisant précéder d'un dièse '#' :

```
#z98y { letter−spacing: 0.3em }
H1#z98y { letter−spacing: 0.5em }

<P ID=z98y>Texte espacé</P>
```

Dans l'exemple ci−dessus, le premier sélecteur vise l'élément 'P' en raison de la valeur de l'attribut 'ID'. Le second sélecteur spécifie à la fois un type d'élément ('H1') et une valeur d'ID, il ne concernera donc pas l'élément 'P'.

En utilisant l'attribut ID comme sélecteur, on peut fixer des propriétés de style par élément. Alors que les feuilles de style sont avant tout destinées à améliorer la structure du document, cette fonction donne aux auteurs la possibilité de créer des documents de bonne présentation dans la page mais sans tirer avantage des éléments structuraux de HTML. Cette utilisation des feuilles de style n'est donc pas encouragée.

1.6 Sélecteurs contextuels

L'héritage de CSS épargne les doigts des auteurs. Au lieu de la mise en place de toutes les propriétés de style, on peut créer une feuille de style par défaut et l'amender d'une liste des exceptions. Pour donner une couleur différente aux éléments 'EM' inclus dans 'H1', on pourrait écrire :

```
H1 { color: blue }
EM { color: red }
```

La feuille de style prenant effet, toutes les sections accentuées dedans et hors de 'H1' auront la couleur rouge. Peut−être voulait−on que seuls les éléments 'EM' dans 'H1' soient rouge, on pouvait le faire avec :

```
H1 EM { color: red }
```

Le sélecteur est devenu un motif de recherche sur la pile des éléments ouverts, on le nomme un *sélecteur contextuel*. Les sélecteurs contextuels sont composés d'une suite de plusieurs sélecteurs simples séparés par un espace (tous les sélecteurs décrits jusqu'ici étaient des sélecteurs simples). Seuls les éléments existants qui correspondent au dernier sélecteur simple de la suite (ici l'élément 'EM') sont concernés, et seulement si le motif de recherche correspond. Les sélecteurs contextuels en CSS1 sont liés à leurs ancêtres, mais l'évolution de la norme pourraient introduire d'autres relations (ex. parent−enfant). Dans l'exemple au−dessus, la reconnaissance

était probante si 'EM' est un descendant de 'H1', c.−à−d. si 'EM' est à l'intérieur d'un élément 'H1'.

```
UL LI    { font−size: small }
UL UL LI { font−size: x−small }
```

Ici, le premier sélecteur correspond aux éléments 'LI' qui ont au moins un ancêtre 'UL'. Le second sélecteur correspond à un sous−ensemble du premier, où les éléments 'LI' ont au moins deux ancêtres 'UL'. Le conflit apparent est résolu par le fait que le motif de recherche du second sélecteur est plus long, et donc plus spécifique, que le premier. Voir l'ordre de cascade (chapitre 3.2) pour plus d'infos.

Les sélecteurs contextuels peuvent être spécifiés au travers des types d'élément, des attributs CLASS, des attributs ID ou d'une combinaison de ceux−ci :

```
DIV P           { font: small sans−serif }
.reddish H1     { color: red }
#x78y CODE      { background: blue }
DIV.sidenote H1 { font−size: large }
```

Le premier sélecteur correspond aux éléments 'P' qui ont un élément 'DIV' parmi leurs ancêtres. Le deuxième correspond aux éléments 'H1' qui ont un ancêtre de la classe 'reddish'. Le troisième aux éléments 'CODE' qui ont un parent dont l'attribut ID est égal à 'x78y'. Et le quatrième aux éléments 'H1' qui ont un ancêtre 'DIV', lui−même de la classe 'sidenote'.

On peut regrouper plusieurs sélecteurs contextuels :

```
H1 B, H2 B, H1 EM, H2 EM { color: red }
```

ce qui est équivalent à :

```
H1 B { color: red }
H2 B { color: red }
H1 EM { color: red }
H2 EM { color: red }
```

1.7 Commentaires

Les commentaires textuels dans les feuilles de style CSS sont les mêmes que ceux employés pour le langage de programmation C [7] :

```
EM { color: red }  /* rouge, vraiment rouge ! */
```

Les commentaires ne peuvent pas être imbriqués. Un interpréteur CSS1 considère un commentaire comme un blanc.

2 Pseudo−classes et pseudo−éléments

En CSS1, un élément reçoit un style en fonction de sa position dans la structure du document. Ce modèle simple est suffisant pour une grande variété de styles, sauf pour quelques effets courants. Les concepts de pseudo−classes et de pseudo−éléments élargissent l'adressage et permettent à des directives externes d'influer sur la mise en forme.

Les sélecteurs CSS peuvent utiliser les pseudo−classes et pseudo−éléments, sans que ceux−ci n'apparaissent dans la source HTML. L'agent utilisateur les "insère" plutôt dans les feuilles de style, sous certaines conditions, lors de l'adressage. On les qualifie par commodité de "classes" et "éléments" pour décrire leur comportement. Plus particulièrement, on décrit leur mode opératoire à l'aide d'une *séquence de balise fictive*.

On utilise les pseudo−éléments pour adresser une partie d'un élément, les pseudo−classes permettent une distinction entre les styles des différents types d'élément de la feuille de style.

2.1 Pseudo−classes des ancres

Habituellement, les agents utilisateurs différencient à l'affichage les liens visités des liens non−visités. En CSS1, cela est fait par des pseudo−classes sur l'élément 'A' :

```
A:link { color: red }       /* lien non-visité */
A:visited { color: blue }   /* lien visité */
A:active { color: lime }    /* lien activé */
```

Tous les éléments 'A' avec un attribut 'HREF' vont recevoir le style d'un et un seul de ces groupes (les ancres avec un nom ne sont pas concernées). Les agents utilisateurs peuvent faire passer un élément de l'état 'visited' à 'link' après un temps non−précisé. L'état 'active' désigne un lien en train d'être sélectionné par l'utilisateur (clic maintenu sur la souris).

On peut assimiler l'action d'une pseudo−classe d'ancre à l'insertion manuelle d'une classe. L'agent utilisateur n'est pas obligé de redessiner la page déjà affichée pour tenir compte des transitions induites par une pseudo−classe d'ancre. Par exemple, une feuille de style peut très bien spécifier que la taille de la police 'font−size' d'un lien actif 'active' soit plus grande que celle d'un lien visité 'visited', ainsi l'agent utilisateur n'est pas tenu de reformater dynamiquement le document quand le lecteur clique sur un lien visité.

Les sélecteurs de pseudo−classe n'ont rien à voir avec les classes normales et vice versa. La règle de style qui suit sera inopérante :

```
A:link { color: red }

<A CLASS=link NAME=target5> ... </A>
```

En CSS1, les pseudo−classes d'ancre n'ont d'effet que sur l'élément 'A'. On pourra omettre le type d'élément du sélecteur :

```
A:link { color: red }
:link { color: red }
```

Les deux sélecteurs ci−dessus adressent les mêmes éléments en CSS1.

Le nom des pseudo−classes est insensible à la casse.

Les pseudo−classes sont utilisables dans les sélecteurs contextuels :

```
A:link IMG { border: solid blue }
```

On peut également utiliser les pseudo−classes en combinaison avec les classes normales :

```
A.external:visited { color: blue }

<A CLASS=external HREF="http://de.hors/">lien externe</A>
```

Si le lien avait été visité dans cet exemple, il apparaîtrait en bleu. Noter que le nom d'une classe normale précède le nom d'une pseudo−classe dans le sélecteur.

2.2 Pseudo−éléments typographiques

Quelques effets typographiques courants ne sont pas associés aux éléments structuraux mais aux objets typographiques de la mise en page. Avec CSS1, deux objets typographiques peuvent être concernés par le biais de pseudo−éléments : la première lettre d'un élément et la première ligne d'un élément.

CSS1 de base : Les agents utilisateurs peuvent ignorer ces pseudo−éléments ':first−line' ou ':first−letter' dans un sélecteur, ou respecter toute ou partie de leurs propriétés. (Voir le chapitre 7).

2.3 Le pseudo−élément 'first−line'

Le pseudo−élément 'first−line' est utilisé pour appliquer un style spécial à la première ligne issue de la mise en page :

```
<STYLE TYPE="text/css">
  P:first-line { font-variant: small-caps }
</STYLE>

<P>La première ligne d'un article dans un journal.
```

Un agent utilisateur en mode texte donnerait :

```
LA PREMIÈRE LIGNE D'UN ARTICLE
dans un journal.
```

La séquence de balise fictive dans cet exemple est :

```
<P>
<P:first-line>
La première ligne d'un article
</P:first-line>
dans un journal.
</P>
```

La balise fermante de 'first−line' est insérée à la fin de la première ligne telle qu'elle apparaît dans la mise en page.

Le pseudo−élément 'first−line' ne s'applique qu'aux éléments de type bloc.

Le pseudo−élément 'first−line' se comporte comme un élément de type en−ligne avec quelques restrictions. Seules sont applicables à 'first−line' : les propriétés de police (5.2), les propriété de couleur et de fond (5.3), et les propriétés 'word−spacing' (5.4.1), 'letter−spacing' (5.4.2), 'text−decoration' (5.4.3), 'vertical−align' (5.4.4), 'text−transform' (5.4.5), 'line−height' (5.4.8), 'clear' (5.5.26).

2.4 Le pseudo−élément 'first−letter'

Le pseudo−élément 'first−letter' s'utilise pour obtenir des effets typographiques courants comme la "lettrine" ou le changement "en bas de casse". Si sa propriété 'float' a pour valeur 'none', il se comporte comme un élément de type en−ligne ou, sinon, comme un élément flottant. Voici les propriétés qui s'appliquent à 'first−line' : les propriétés de police (5.2), de couleur et de fond (5.3), 'text−decoration' (5.4.3), 'vertical−align' (seulement si 'float' a comme valeur 'none', 5.4.4), 'text−transform' (5.4.5), 'line−height' (5.4.8), les propriétés de marge (5.5.1−5.5.5), d'espacement (5.5.6−5.5.10), de bordure (5.5.11−5.5.22), et les propriétés 'float' (5.5.25) et 'clear' (5.5.26).

Voici comment on pourrait faire une lettrine qui s'étend sur deux lignes :

```
<HTML>
 <HEAD>
  <TITLE>Titre</TITLE>
  <STYLE TYPE="text/css">
   P              { font-size: 12pt; line-height: 12pt }
   P:first-letter { font-size: 200%; float: left }
   SPAN           { text-transform: uppercase }
  </STYLE>
 </HEAD>
 <BODY>
  <P><SPAN>Les premiers</SPAN> mots d'un article de journal.</P>
 </BODY>
</HTML>
```

Une représentation (improbable) par un agent utilisateur en mode texte supportant le pseudo−élément 'first−letter' donnerait :

```
|  ES PREMIERS
|_ mots d'un
article de journal.
```

La séquence de balise fictive en est :

```
<P>
<SPAN>
<P:first−letter>
L
</P:first−letter>es premiers
</SPAN>
mots d'un article de journal.
</P>
```

Noter que les balises du pseudo−élément 'first−letter' butent contre le contenu (c.−à−d. le caractère initial), alors que la balise ouvrante du pseudo−élément 'first−line' s'insère juste après la balise ouvrante de l'élément qui le contient.

L'agent utilisateur détermine les caractères qui font partie de l'élément 'first−letter'. Normalement, il devrait inclure le guillemet éventuel qui précède la première lettre :

```
|||    | n tiens
  |    | vaut mieux
  |    | que deux
  |____| tu l'auras,"
dit un vieux proverbe.
```

Quand la première lettre d'un paragraphe est un autre signe de ponctuation (ex. parenthèse, ellipses) ou un autre caractère non assimilé à une lettre (ex. chiffres et symboles mathématiques), 'first−letter' est en général ignoré.

L'usage diffère selon les langues. Par exemple en hollandais, un mot qui commence par "ij" alors cette combinaison devrait être considérée dans son entier par 'first−letter'.

On ne peut lier le pseudo−élément 'first−letter' qu'à un élément de type bloc.

2.5 Pseudo−éléments dans les sélecteurs

Dans un sélecteur contextuel, on ne peut mettre les pseudo−éléments qu'à la fin du sélecteur :

```
BODY P:first−letter { color: purple }
```

On peut combiner les pseudo−éléments avec des classes dans les sélecteurs :

```
P.initial:first−letter { color: red }

<P CLASS=initial>Premier paragraphe</P>
```

Dans cet exemple, la première lettre des éléments 'P' qui ont pour classe 'initial' est de couleur rouge. En combinaison avec des classes ou des pseudo−classes, on doit spécifier les pseudo−éléments à la fin du sélecteur. Il ne peut y avoir qu'un pseudo−élément par sélecteur.

2.6 Pseudo−éléments multiples

On peut combiner plusieurs pseudo−éléments :

```
P { color: red; font−size: 12pt }
P:first−letter { color: green; font−size: 200% }
P:first−line { color: blue }

<P>Texte qui se poursuit sur deux lignes</P>
```

Dans cet exemple, la première lettre de chaque élément 'P' a une couleur verte et un corps de police 24pt. Le reste de la première ligne (selon le formatage de la page) a une couleur bleu, puis le reste du paragraphe une couleur rouge. En supposant qu'il y ait une fin de ligne avant le mot "poursuit", la séquence de balise fictive serait :

```
<P>
<P:first−line>
<P:first−letter>
T
</P:first−letter>exte qui se
</P:first−line>
poursuit sur deux lignes
</P>
```

Noter que l'élément 'first−letter' est placé à l'intérieur de l'élément 'first−line'. Les propriétés de 'first−line' seront héritées par 'first−letter', mais avec leurs valeurs remplacées par celles déclarées pour 'first−letter' si tel est le cas.

Si un pseudo−élément coupe un élément réel, des balises supplémentaires doivent nécessairement être recréées dans la séquence de balise fictive. Par exemple, si un élément 'SPAN' s'étend au−delà d'une balise `</P:first−line>`, alors un jeu de balise fermante et ouvrante sera créé, la séquence de balise fictive devient :

```
<P>
<P:first−line>
<SPAN>
Ce texte est à l'intérieur d'un
</SPAN>
</P:first−line>
<SPAN>
interminable élément 'SPAN'.
</SPAN>
```

3 La cascade

Avec CSS, la présentation peut être influencée simultanément par plusieurs feuilles de style. Deux raisons à cela : la modularité, et l'équilibre entre les prérogatives de l'auteur et du lecteur.

modularité
: Un auteur peut combiner plusieurs feuilles de style (partielles) pour éviter la redondance :

```
@import url(http://www.style.org/pastoral);
@import url(http://www.style.org/marine);

H1 { color: red }     /* annule le style importé */
```

équilibre
: Le lecteur et l'auteur peuvent tous deux agir sur la présentation au travers des feuilles de style. Pour cela, ils usent du même langage de feuille de style, ce qui illustre un aspect fondamental du Web : tout le monde peut publier. Le mécanisme de référencement des feuilles de style personnelles est laissé au choix de l'agent utilisateur.

Des conflits peuvent naître entre les feuilles de styles agissant sur la présentation. La résolution de ceux−ci est fondée sur un poids attribué à chaque règle. Les règles des feuilles de style du lecteur ont par défaut un poids moindre que celles des feuilles de style de l'auteur. C.−à−d., si un conflit apparaît entre les feuilles de style du lecteur et celles de l'auteur, les dernières auront la priorité. Les directives du lecteur et de l'auteur remplacent toutes deux les valeurs par défaut de l'agent utilisateur.

Les feuilles de style importées agissent aussi en cascade les unes avec les autres, dans l'ordre d'apparition, et selon les règles de cascade définies ci−dessous. Toutes règles spécifiées dans la feuille de style elle−même prennent le pas sur celles importées. C.−à−d., les feuilles de style importées sont placées plus bas dans l'ordre de cascade que les règles de la feuille de style. Ces feuilles de style importées peuvent à leur tour importer et surcharger d'autres feuilles de style récursivement.

En CSS1, toutes les déclarations '@import' doivent être placées au début de la feuille de style, avant toutes autres déclarations. Cela constitue un rappel visuel de la préséance des règles qui sont dans la feuille de style sur celles des feuilles de style importées.

3.1 'important'

L'auteur peut accroître le poids des règles de sa feuille de style :

```
H1 { color: black ! important; background: white ! important }
P  { font−size: 12pt ! important; font−style: italic }
```

Dans l'exemple ci−dessus, un poids supplémentaire est donné aux trois premières déclarations, la dernière gardant un poids normal.

La règle déclarée importante du lecteur est prioritaire sur celle normale de l'auteur. La règle déclarée importante de l'auteur annule celle importante du lecteur.

3.2 Ordre de cascade

Les règles conflictuelles sont intrinsèques au mécanisme des CSS. L'algorithme suivant doit être respecté pour trouver la valeur d'une combinaison entre élément et/ou propriété :

1. Trouver toutes les déclarations qui s'appliquent à l'élément ou la propriété en question. Les déclarations s'appliquent si le sélecteur concerne l'élément donné. Si aucune déclaration ne correspond, les valeurs héritées sont utilisées. S'il n'y en a pas (l'élément 'HTML' et autres propriétés qui ne sont pas héritées), alors utiliser la valeur initiale.
2. Trier les déclarations selon leur poids explicite : les déclarations marquées '!important' ont plus de poids que celles non−marquées (normal).
3. Trier selon l'origine : les feuilles de style de l'auteur surclassent celles du lecteur qui surclassent à leur tour les valeurs internes par défaut de l'agent utilisateur. Une feuille de style importée est considérée avoir la même origine que celle dont elle est issue.
4. Trier selon la spécificité du sélecteur : un sélecteur général est surclassé par un sélecteur plus spécifique. Déterminer la spécificité en comptant le nombre d'attributs ID (a), le nombre d'attributs CLASS (b) et le nombre de noms de balise (c) dans le sélecteur. Rassembler ces trois nombres (dans un système de

nombres sur une base étendue) pour obtenir la spécificité. Un exemple :

```
LI            {...}  /* a=0 b=0 c=1 -> spécificité =   1 */
UL LI         {...}  /* a=0 b=0 c=2 -> spécificité =   2 */
UL OL LI      {...}  /* a=0 b=0 c=3 -> spécificité =   3 */
LI.red        {...}  /* a=0 b=1 c=1 -> spécificité =  11 */
UL OL LI.red  {...}  /* a=0 b=1 c=3 -> spécificité =  13 */
#x34y         {...}  /* a=1 b=0 c=0 -> spécificité = 100 */
```

On considère respectivement les pseudo−classes et pseudo−éléments comme classes et éléments normaux.

5. Trier dans l'ordre de spécification : si deux règles ont le même poids, alors la dernière l'emporte. Les règles importées sont considérées comme venant en premier dans la feuille de style elle−même.

La valeur d'une propriété est fixée dès que la règle de plus fort poids parmi celles qui s'appliquent à la même combinaison d'éléments/propriétés est trouvée.

Cette stratégie donne un poids considérable aux feuilles de styles d'un auteur. C'est pourquoi il est important que le lecteur puisse annuler l'influence de telle ou telle feuille de style, par exemple, choix au moyen d'un menu déroulant.

Une déclaration faite au moyen de l'attribut 'STYLE' d'un élément a le même poids que celle dans un sélecteur basé sur un ID (voir un exemple au chapitre 1.1) spécifié en fin de feuille de style :

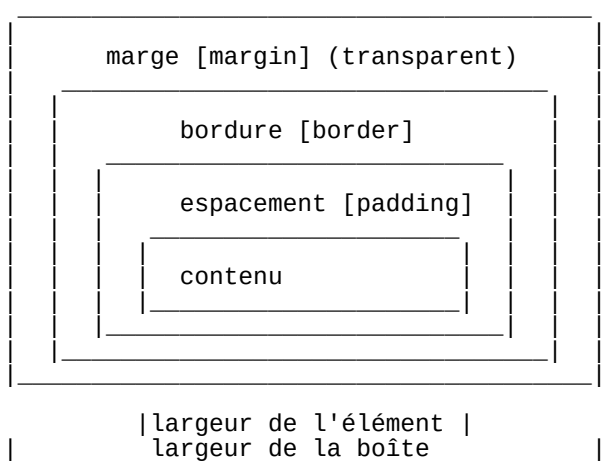
```
<STYLE TYPE="text/css">
  #x97z { color: blue }
</STYLE>

<P ID=x97z STYLE="color: red">
```

Dans cet exemple, l'élément 'P' a la couleur rouge. Bien que les deux déclarations aient le même poids, celle de l'attribut 'STYLE' annule celle de l'élément 'STYLE' du fait de la règle de cascade numéro 5.

L'agent utilisateur peut honorer les autres attributs stylistiques propres au HTML, p.ex. 'ALIGN'. Dans ce cas, ils sont traduits dans leurs équivalents CSS avec une spécificité égale à 1. Ces règles induites sont placées en tête de la feuille de style de l'auteur et peuvent donc être remplacées par les règles écrites qui suivent. Cette façon de faire facilitera la coexistence des attributs stylistiques et des feuilles de style pendant une phase de transition.

4 Modèle de mise en forme

CSS1 repose sur un modèle de mise en forme simple bâti sur un concept de boîtes dans lequel chaque élément formaté s'inscrit dans une ou plusieurs boîtes rectangulaires. Les éléments ayant une valeur 'none' pour la propriété 'display' n'ont pas de format et ne sont donc pas inscrits dans une telle boîte. Toutes les boîtes sont formées d'une aire avec un contenu de base entourée d'un espacement optionnel, d'une bordure et d'aires de marge.



Les tailles de la marge, de l'espacement et de la bordure sont fixées respectivement avec les propriétés 'margin' (5.5.1−5.5.5), 'padding' (5.5.6−5.5.10) et 'border' (5.5.11−5.5.22). La zone d'espacement est sur le fond de l'élément lui−même (spécifié par les propriétés de fond (5.3.2−5.3.7)). La couleur et le style de la bordure est donnée par les propriétés de bordure. Les marges sont toujours transparentes, on voit donc le parent de l'élément à travers.

La taille de la boîte est égale à la largeur de l'élément (c.−à−d. celle d'un texte formaté ou d'une image) et de l'espacement, auquel on ajoute la bordure et les marges.

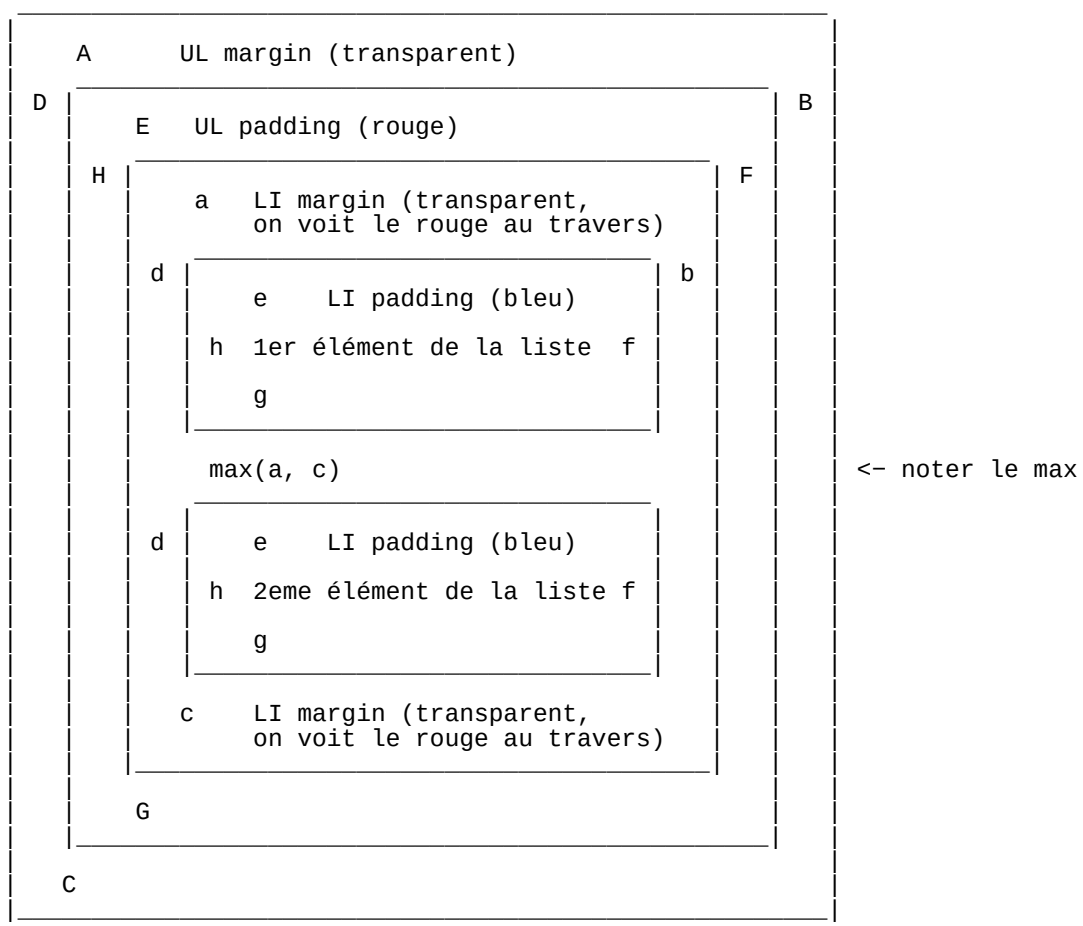
Le formateur considère principalement deux genres d'élément : ceux de type bloc et ceux de type en−ligne.

4.1 Éléments de type bloc

Les éléments qui ont une valeur 'block' ou 'list−item' pour la propriété 'display' sont des éléments *de type bloc*. On considère également de ce type les éléments flottants (dont la propriété 'float' a une valeur autre que 'none').

L'exemple suivant montre comment les marges et l'espacement régissent un élément 'UL' avec deux enfants. On a omis la bordure dans le schéma par souci de simplification. Les lettres faisant office de "constantes" ne font pas partie de la syntaxe CSS1, elles montrent juste le lien entre les valeurs de la feuille de style et le dessin.

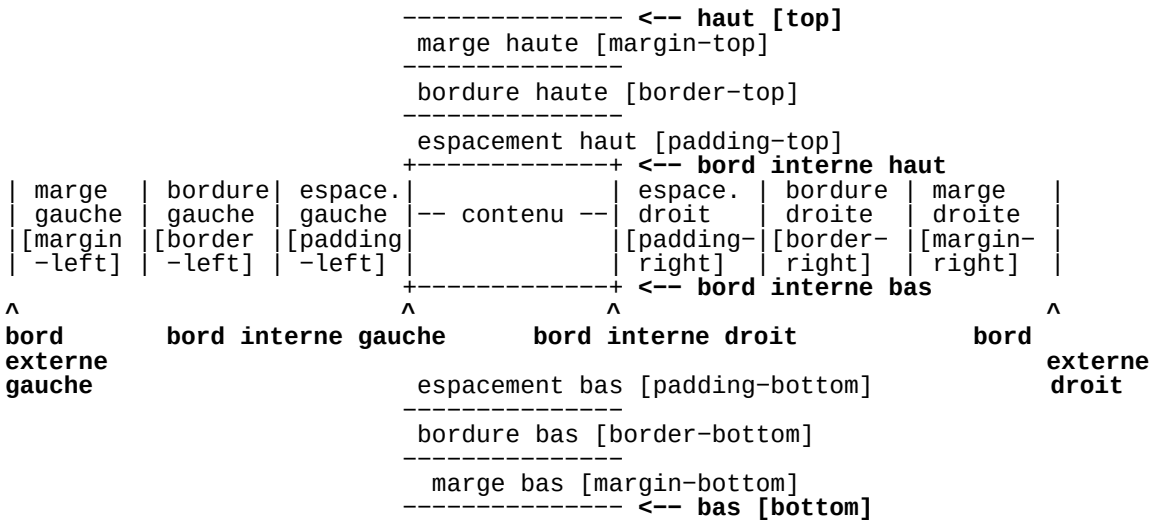
```
<STYLE TYPE="text/css">
  UL {
    background: red;
    margin: A B C D;      /* marge */
    padding: E F G H;     /* espacement */
  }
  LI {
    color: white;
    background: blue;     /* le texte est blanc sur fond bleu */
    margin: a b c d;
    padding: e f g h;
  }
</STYLE>
..
<UL>
  <LI>1er élément de la liste
  <LI>2eme élément de la liste
</UL>
```



Techniquement, les propriétés d'espacement [ndt. padding] et de marge [ndt. margin] ne sont pas héritées. Mais, dans l'exemple, l'emplacement d'un élément est relatif à ses parents et à ses enfants, ainsi les propriétés d'espacement et de marge de ces éléments agissent sur leurs enfants.

S'il y avait eu des bordures, elles apparaîtraient entre l'espacement et les marges.

Le schéma suivant amène une terminologie utile :



Le *bord externe gauche* représente le bord d'un élément, y compris l'espacement, la bordure et la marge. Le *bord interne gauche* représente le bord du seul élément, à l'exclusion de l'espacement, la bordure et la marge. Idem pour le côté droit. Le *haut* correspond au sommet de l'élément, y compris l'espacement, la bordure et la marge : il est seulement défini pour les éléments de type bloc et les éléments flottants, pas pour les éléments de type bloc non−flottants. Le *bas* correspond au bas de l'élément, y compris l'espacement, la bordure et la marge : il est seulement défini pour les élément de type bloc et les éléments flottants, pas pour les élément de type bloc

non−flottants. Le *bord interne bas* représente le bas de l'élément, à l'exclusion de l'espacement, la bordure et la marge.

La *largeur* d'un élément est égale à la largeur du contenu, c.−à−d. la distance entre le bord interne gauche et le bord interne droit. Et sa *hauteur* est égale à la hauteur du contenu, c.−à−d. la distance entre le bord interne haut et le bord interne bas.

4.1.1 Mise en forme verticale

Pour les éléments de type bloc non−flottants, la largeur de la marge correspond à une distance minimum entre le contenu et le bord des boîtes qui l'entourent. Si aucune bordure, ni espacement, ni contenu ne séparent deux (ou plus) marges verticales adjacentes, alors elles fusionnent et la résultante est la plus grande de ces valeurs de marge. Dans la plupart des cas, cette fusion des marges verticales donne un meilleur aspect visuel, plus proche des attentes de l'auteur. Dans l'exemple précédent, les marges entre les deux éléments 'LI' ont fusionné et la marge résultante prend la valeur la plus grande entre celle de 'margin−bottom' du premier élément 'LI' et celle de 'margin−top' du second élément 'LI'. Pareillement, si l'espacement entre le 'UL' et le premier élément 'LI' (constante "E") avait été nul, les marges des éléments 'UL' et du premier 'LI' auraient fusionné.

Cas des marges négatives : la valeur maximale absolue entre marges négatives adjacentes est déduite de la plus grande valeur des marges positives. S'il n'y a pas de marges positives, alors la valeur maximale absolue d'entre les marges négatives est déduite de zéro.

4.1.2 Mise en forme horizontale

L'emplacement horizontal et la taille d'un élément de type bloc non−flottant sont déterminés par sept propriétés : 'margin−left', 'border−left', 'padding−left', 'width', 'padding−right', 'border−right' et 'margin−right'. La somme des sept valeurs associées est toujours égale à la valeur de 'width' du parent de l'élément.

Par défaut, la propriété 'width' d'un élément a pour valeur 'auto'. Si ce n'est pas un élément remplacé (ex. IMG), cela implique de la part de l'agent utilisateur l'addition des valeurs des sept propriétés citées ci−dessus pour arriver à la valeur de 'width' du parent. Si c'est un élément remplacé, la valeur 'auto' de 'width' devient automatiquement celle de la largeur intrinsèque de cet élément.

Trois de ces sept propriétés peuvent prendre la valeur 'auto' : 'margin−left', 'width' et 'margin−right'. Pour les éléments remplacés, la valeur 'auto' de 'width' étant remplacée par leur largeur intrinsèque, il ne leur reste donc que deux propriétés pouvant avoir la valeur 'auto'.

La propriété 'width' a une valeur minimum positive propre à l'agent utilisateur (variant d'un élément à un autre et pouvant dépendre d'autres propriétés). Si la valeur calculée devient inférieure à cette limite, qu'elle soit déclarée explicitement ou à 'auto', et que les règles suivantes la rendent trop petite, elle prendrait cette valeur limite.

Si *une et une seule* parmi 'margin−left', 'width' ou 'margin−right' a la valeur 'auto', l'agent utilisateur assigne à cette propriété−ci une valeur telle que la somme des sept propriétés soit égale à la largeur du parent.

Si *aucune* de ces trois propriétés n'a la valeur 'auto', alors la valeur de 'margin−right' passe à 'auto'.

Si *plus d'une* de ces trois propriétés ont la valeur 'auto', l'une d'elles étant 'width', alors les deux autres ('margin−left' et/ou 'margin−right') prennent la valeur zéro et 'width' une valeur telle que la somme des sept soit égale à la largeur du parent.

Autrement, si 'margin−left' et 'marging−right' ont la valeur 'auto', alors ces deux propriétés reçoivent la même valeur. De cette façon, l'élément est centré dans son parent.

Si une des sept propriétés d'un élément de type en−ligne ou flottant a la valeur 'auto', la valeur de cette propriété−ci devient nulle.

À la différence des marges verticales, les marges horizontales ne fusionnent pas.

4.1.3 Éléments de liste

Les éléments dont la propriété 'display' a la valeur 'list−item' ont un format de type bloc, mais avec la marque de liste qui les précède. La propriété 'list−style' détermine le type de la marque. Le placement de cette marque dépend de la valeur de la propriété 'list−style' :

```
<STYLE TYPE="text/css">
  UL         { list-style: outside }
```

```
  UL.compact { list-style: inside }
</STYLE>

<UL>
  <LI>le premier article vient avant
  <LI>le second article vient après
</UL>

<UL CLASS=COMPACT>
  <LI>le premier article vient avant
  <LI>le second article vient après
</UL>
```

Ce qui apparaît ainsi :

```
* le premier article
  vient devant

* le second article
  vient après

  * le premier
  article vient avant

  * le second
  article vient après
```

Avec un sens de lecture de droite à gauche, les marques seraient du côté droit de la boîte.

4.1.4 Éléments flottants

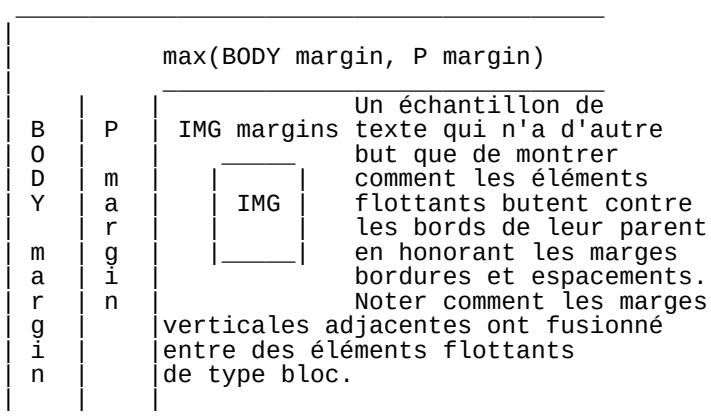
Avec une propriété 'float', un élément sort du flux normal des autres éléments et acquiert un format de type bloc. Par exemple, en donnant la valeur 'left' à la propriété 'float' d'une image, celle-ci est repoussée vers la gauche jusqu'à buter sur les marges, espacements ou bordures d'un autre élément de type bloc. Le flux normal se déroule sur le côté droit de l'image. Ses marges, bordures et espacements sont respectés, cependant les marges ne fusionnent jamais avec celles des éléments adjacents.

La position d'un élément flottant obéit aux contraintes suivantes (voir le chapitre 4.1 pour le sens des termes) :

1. Le bord externe gauche d'un élément flottant à gauche ne peut se trouver plus à gauche que le bord interne gauche de son parent. Et inversement pour un élément flottant à droite.
2. Le bord externe gauche d'un élément flottant à gauche doit se trouver sur la droite du bord externe droit de tout élément flottant à gauche qui précède (dans l'ordre de la source HTML) ou encore le sommet du suivant doit se trouver sous le bas du précédent. Et inversement pour les éléments flottants à droite.
3. Le bord externe droit d'un élément flottant à gauche ne peut pas se trouver à droite du bord externe gauche d'éléments flottants à droite situés à sa droite. Et inversement pour un élément flottant à droite.
4. Le sommet d'un élément flottant ne peut être plus haut que le bord haut interne de son parent.
5. Le sommet d'un élément flottant ne peut être plus haut que celui d'un élément flottant ou de type bloc qui le précèdent.
6. Le sommet d'un élément flottant ne peut être plus haut que le sommet d'une quelconque *boîte de ligne* avec contenu (voir chapitre 4.4) qui le précède dans la source HTML.
7. Un élément flottant est placé aussi haut que possible.
8. Un élément flottant à gauche doit être placé aussi à gauche que possible, et inversement pour un élément flottant à droite. Un emplacement au plus haut est préférable à celui décalé sur la gauche ou la droite.

```
<STYLE TYPE="text/css">
  IMG { float: left }
  BODY, P, IMG { margin: 2em }
</STYLE>

<BODY>
  <P>
    <IMG SRC=img.gif>
    Un échantillon de texte qui...
</BODY>
```

Voici ce que donnerait l'exemple :



Noter comment la marge de l'élément 'P' enserre l'élément flottant 'IMG'.

Deux situations peuvent se présenter où des éléments flottants peuvent recouvrir de leurs marges, bordures et espacements les emplacements d'autres éléments :

- quand l'élément flottant a une marge négative ; les marges négatives de ces éléments sont respectées comme pour les autres éléments de type bloc.
- quand l'élément flottant est plus large ou plus haut que son parent.

4.2 Éléments de type en−ligne

Les éléments qui n'ont pas un format de type bloc sont des éléments de type *en−ligne*. Un élément en−ligne peut partager l'étendue de la ligne avec d'autres éléments. Un exemple :

```
<P>Plusieurs mots <EM>accentués</EM> sont <STRONG>visibles</STRONG>.</P>
```

L'élément 'P' est de type bloc, alors que 'EM' et 'STRONG' sont de type en−ligne. Si l'élément 'P' est suffisamment large pour contenir le tout sur une ligne, ces deux éléments en−ligne sont donc sur celle−ci :

```
Plusieurs mots accentués sont visibles.
```

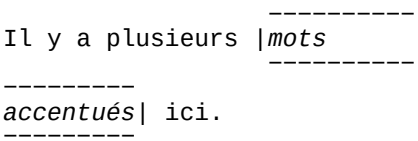
S'il manque de place sur une ligne, un élément en−ligne est partagé en plusieurs boîtes :

```
<P>Il y a plusieurs <EM>mots accentués</EM> ici.</P>
```

ce qui est illustré par :

```
Il y a plusieurs mots
accentués ici.
```

Si l'élément en−ligne a des marges, bordures, espacements ou des décorations de texte qui lui sont associés, ces propriétés n'ont aucun effet à l'endroit où intervient une coupure de l'élément :



(Le dessin à partir de lettres est légèrement distordu. Voir le chapitre 4.4 sur la manière de calculer la hauteur des lignes).

4.3 Éléments remplacés

On appelle élément remplacé un élément auquel est substitue un contenu désigné depuis l'intérieur même de celui−ci. Par exemple en HTML, l'élément 'IMG' est remplacé par une image référencée par l'attribut 'SRC'. L'élément substitué est sensé apporter ses dimensions intrinsèques. Si la propriété 'width' a la valeur 'auto', la largeur de l'élément sera celle de la valeur de la largeur intrinsèque. Si on spécifie une valeur autre que 'auto' dans la feuille de style, c'est celle−ci qui est retenue pour dimensionner l'élément substitué (la méthode de dimensionnement dépendra du type du contenu référencé). La propriété 'height' est utilisée de la même façon.

Les éléments remplacés sont aussi bien de type bloc que en−ligne.

4.4 Hauteur des lignes

Tous les éléments ont une propriété 'line−height' qui est en principe la hauteur totale d'une ligne de texte. Dans la ligne, le texte reçoit un espace complémentaire au−dessus et en−dessous de celui−ci pour atteindre la valeur de la hauteur de ligne. Par exemple, si le texte a un corps de 12pt et la valeur de 'line−height' est 14pt, il est réparti un complément de 2pt entre le dessus (1pt) et le dessous (1pt) de la ligne. Ce calcul dépend des éléments, qu'ils aient un contenu ou non.

On appelle *interligne [ndt. leading]* la différence entre le corps de la police et la hauteur de la ligne, sa moitié se nomme *demi−interligne*. Après mise en forme, chaque ligne prend la forme d'une *boîte de ligne [ndt. line−box]* rectangulaire.

Si une ligne contient des parties dont les valeurs pour 'line−height' sont différentes (c'est qu'elle contient des éléments en−ligne), alors chacune de ces parties a sa propre demi−interligne au−dessus et au−dessous. La hauteur de la boîte de ligne s'étend du sommet de l'élément le plus haut au bas de l'élément le plus bas. Noter que le sommet et le bas de la ligne ne correspondent pas forcément à l'élément de plus grande hauteur, car la position dans le sens vertical d'un élément dépend aussi de sa propriété 'vertical−align'. Chaque boîte de ligne est empilée immédiatement sous la ligne précédente pour former un paragraphe.

Noter qu'espacement, bordure et marge du dessus et du dessous d'un élément en−ligne mais qui n'est pas remplacé n'influencent pas la hauteur de la ligne. C.−à−d., si la valeur de 'line−height' est trop petite pour l'espacement et la bordure choisie, l'élément risque de recouvrir le texte des autres lignes.

Dans une ligne, les éléments remplacés (ex. images) peuvent agrandir la boîte de ligne si son sommet (espacements, bordures et marges inclus) est plus haut que le sommet de la partie texte ou si son bas est en dessous du bas de ce même texte.

En temps normal, quand il n'y a qu'une seule valeur pour 'line−height' dans tout un paragraphe, sans images trop hautes, cette définition garantie que la base du texte des lignes successives est dissociée de 'line−height'. C'est important quand des colonnes de texte avec différentes polices doivent être alignées, dans une table par exemple.

Noter que ceci n'empêche pas le texte sur deux lignes adjacentes de se recouvrir. La valeur de 'line−height' peut être plus petite que la hauteur du texte ce qui traduit un interlignage négatif. Cela peut être utile si on sait que le texte ne contient pas de caractères avec jambage (ex. texte tout en capitales), on peut ainsi rapprocher les lignes entre elles.

4.5 Le canevas

On appelle canevas la partie de la surface affichée par l'agent utilisateur sur laquelle le document est représenté. Aucun élément de structure ne correspond à cette définition, ce qui amène deux questions au moment du mise en forme d'un document :

- quel est le repère à partir duquel établir les dimensions du canevas ?
- si un document ne recouvre pas le canevas en son entier, comment rendre cet espace non occupé ?

À la première question, la raison veut que la largeur initiale du canevas soit basée sur la largeur de la fenêtre, cependant CSS1 ne répond pas à cette interrogation laissant l'agent utilisateur décider. On peut attendre de l'agent utilisateur qu'il modifie la largeur du canevas quand on redimensionne la fenêtre, mais ce n'est pas le but de CSS1 non plus.

À la deuxième question, on a le précédent des extensions HTML : les attributs de l'élément 'BODY' qui fixent les valeurs de fond de tout le canevas. Pour répondre aux souhaits des auteurs, CSS1 introduit une règle spéciale pour trouver le fond du canevas :

> Utiliser la valeur de la propriété 'background' de l'élément 'HTML' si elle est différente de 'transparent', sinon prendre la valeur de 'background' de l'élément 'BODY'. Si cette valeur est 'transparent', alors le rendu n'est pas défini.

Cette règle permet cette construction :

```
<HTML STYLE="background: url(http://style.com/marble.png)">
<BODY STYLE="background: red">
```

Ici, le canevas est couvert par l'image "marble.png". Le fond de l'élément 'BODY' (qui recouvre complètement ou non le canevas) est rouge.

Comme on n'a pas encore de moyens disponibles pour adresser directement le canevas, il est recommandé de reporter les propriétés souhaitées pour celui−ci sur l'élément 'BODY'.

4.6 Élément 'BR'

Le comportement de l'élément 'BR' n'est pas décrit par les propriétés et valeurs de CSS1. En HTML, 'BR' équivaut à une coupure de ligne entre des mots. En fait, l'élément est remplacé par une fin de ligne. Les versions ultérieures de CSS prendront peut−être en compte un contenu ajouté ou remplacé, dans l'immédiat les interpréteurs CSS1 doivent traiter 'BR' spécifiquement.

5 Propriétés de CSS1

Les feuilles de style agissent sur la présentation des documents par assignation de valeurs aux propriétés de style. Ce chapitre dresse la liste des définitions de ces propriétés ainsi que les valeurs qui peuvent leur être associées selon CSS1.

5.1 Notation des valeurs des propriétés

La liste des valeurs autorisées pour chaque propriété suit une syntaxe de ce genre :

Valeur : N | NW | NE
Valeur : [<longueur> | thick | thin]{1,4}
Valeur : [<nom de la famille> ,]* <nom de la famille>
Valeur : <url>? <couleur> [/ <couleur>]?
Valeur : <url> || <couleur>

Les mots entre "<" et ">" indiquent le type de la valeur. Les plus communs sont <longueur>, <pourcentage>, <url>, <nombre> et <couleur> qui sont définis au chapitre 6. Ceux d'un type plus spécialisé (ex. <font−family> et <border−style>) sont décrits dans la propriété correspondante.

Les autres mots sont des mots−clés devant apparaître littéralement, sans guillemets, tout comme la barre oblique (/) et la virgule (,).

Si plusieurs items sont juxtaposés, alors tous doivent survenir dans l'ordre donné. La barre (|) sépare les choix possibles, cela signifie qu'un parmi ceux−ci doit être choisi. La double barre (A || B) signifie le choix de A ou B ou des deux, dans n'importe quel ordre. Les crochets ([ ]) servent au regroupement. La juxtaposition a un poids plus grand que la double barre, et la double barre que la barre simple. Ainsi "a b | c || d e" équivaut à "[a b] | [c || [d e]]".

Chaque type, mot−clé ou groupe entre crochets peut être suivi par l'un des modificateurs suivants :

- L'astérisque (*) signifie que ce qui précède se répète zéro ou plusieurs fois.
- Le plus (+), ce qui précède se répète une ou plusieurs fois.
- Le point d'interrogation (?), ce qui précède est facultatif.
- Une paire de nombre dans des accolades ({A,B}), ce qui précède se répète au moins A fois et au plus B fois.

5.2 Propriétés des polices

Agir sur les propriétés des polices est un des usages les plus communs des feuilles de style. Malheureusement, il n'existe pas de taxonomie bien définie et universellement reconnue pour classer les polices, les termes appropriés à une famille de police ne s'appliquent pas forcément à d'autres familles. Par exemple, 'italic' est souvent utilisé pour désigner un texte en italique, mais un texte en italique est aussi qualifié d'écriture *Oblique, Penchée, Inclinée, Cursive* voire *Kursiv*. Ce qui rend difficile la relation entre les propriétés typiques de la police sélectionnée et une police spécifique.

CSS1 définit les propriétés 'font−family', 'font−style', 'font−variant', 'font−weight', 'font−size' et 'font'.

5.2.1 Ajustement des polices

Du fait d'une taxonomie des polices non universelle, l'ajustement des propriétés à l'aspect des polices requiert un soin particulier. Elles sont ajustées dans un ordre bien déterminé pour assurer un rendu aussi consistant que possible entre agents utilisateurs (en supposant que la même librairie de polices leur soit présentée).

1. L'agent utilisateur construit (ou accède à) une base de données des propriétés concernées de CSS1 pour toutes les polices dont l'existence lui est connue. Une police peut être connue parce qu'elle est déjà installée localement ou parce qu'elle a été téléchargée du Web un peu plus tôt. Si deux polices ont exactement les mêmes caractéristiques, l'une d'elles est ignorée.
2. L'agent utilisateur réunit les propriétés de texte qui s'applique à un élément donné et pour chaque caractère de cet élément. Celui−ci, à l'aide du jeu complet des propriétés, applique la propriété 'font−family' pour y choisir une police et l'essayer. Pour chacune des autres propriétés, une vérification de concordance est faite avec celle−ci. Si toutes les propriétés sont concordantes, cette police est retenue pour cet élément.

3. Si cette précédente étape échoue à faire correspondre la police désignée par 'font−family' et une police disponible, alors cette étape est reconduite avec la police suivante spécifiée dans 'font−family' tant qu'il y en a une.
4. Si une police est trouvée mais qu'un caractère ne dispose pas d'un glyphe dans celle−ci, on répète le processus de correspondance avec la police suivante désignée par 'font−family', s'il y en a une. Voir l'appendice C pour des renseignements sur les polices et l'encodage des caractères.
5. Si aucune police ne correspond à la famille spécifiée par 'font−family', alors l'agent utilisateur recommence les tests à partir de ses propres polices par défaut jusqu'à trouver le meilleur compromis parmi celles−ci.

(Cet algorithme peut être amélioré en évitant de tester chaque caractère avec l'ensemble des propriétés CSS1).

Les tests de correspondance de l'étape 2 ci−dessus se fondent sur les règles suivantes :

1. Le test sur 'font−style' est essayé en premier. La valeur 'italic' est retenue si la base de données des polices de l'agent utilisateur contient une police marquée avec les mots−clés 'italic' (préférable) ou bien 'oblique'. Autrement les valeurs doivent correspondre exactement sinon le test sur 'font−style' échoue.
2. Puis vient le test sur 'font−variant'. La valeur 'normal' retient une police qui n'est pas marquée 'small−caps'. La valeur 'small−caps' retient (1) une police marquée 'small−caps' ou (2) une police dans laquelle les petites capitales sont synthétisées ou (3) une police dont toutes les minuscules sont remplacées par des capitales. On peut synthétiser électroniquement les petites capitales en jouant sur l'échelle des capitales d'une police normale.
3. Le test sur 'font−weight' n'échoue jamais. (Voir 'font−weight' plus loin).
4. Le résultat du test sur 'font−size' doit s'inscrire dans une marge de valeurs qui dépend de l'agent utilisateur (typiquement, la taille des polices variables est arrondie au pixel entier le plus proche alors que la tolérance sur la taille des polices fixes peut aller jusqu'à 20%). Les calculs supplémentaires (p.ex. sur les valeurs exprimées en 'em' pour les autres propriétés) se basent alors sur la valeur établie pour 'font−size' et non sur la valeur qui y est spécifiée.

5.2.2 'font−family'

Valeur : [[<nom de la famille de police> | <famille générique>],]* [<nom de la famille de police> | <famille générique>]
Valeur initiale : propre à l'agent
S'applique à : tous les éléments
Héritée : oui
Pourcentage : sans objet (s.o.)

Valeur représente une liste par ordre de priorité de noms de famille de polices et/ou de noms de famille génériques. Différence avec la plupart des autres propriétés CSS1, on sépare les valeurs avec une virgule pour montrer les choix possibles :

```
BODY { font−family: gill, helvetica, sans−serif }
```

On a deux types de liste de valeurs :

<nom de la famille de police>
: C'est le nom de la famille de police choisie. Dans l'exemple, "gill" et "helvetica" sont des familles de polices.

<famille générique>
: Toujours dans l'exemple, la dernière valeur est le nom d'une famille générique. Voici celles qui sont définies :
  - ◊ 'serif' (ex. Times)
  - ◊ 'sans−serif' (ex. Helvetica)
  - ◊ 'cursive' (ex. Zapf−Chancery)
  - ◊ 'fantasy' (ex. Western)
  - ◊ 'monospace' (ex. Courier)

  On encourage les auteurs à proposer en dernier ressort une famille générique.

Si le nom d'une police contient un espace, on l'écrit entre guillemets :

```
BODY { font−family: "new century schoolbook", serif }
```

```
<BODY STYLE="font−family: 'Ma jolie police', fantasy">
```

Si on ne met pas les guillemets, les blancs situés avant et après le nom de la police sont ignorés et toute séquence de blancs à l'intérieur du nom se réduit à un seul.

5.2.3 'font−style'

Valeur : normal | italic | oblique
Valeur initiale : normal
S'applique à : tous les éléments
Héritée : oui
Pourcentage : s.o.

La propriété 'font−style' effectue une sélection entre les déclinaisons d'une police, soit normale (qu'on qualifie aussi de "romaine"), italique et oblique.

La valeur 'normal' sélectionne une police dans la base de données des polices de l'agent utilisateur classée 'normal', et 'oblique' une police marquée 'oblique'. La valeur 'italic' recherche une police marquée 'italic' ou, si non disponible, 'oblique'.

Une police marquée 'oblique' dans la base de données de l'agent utilisateur peut en fait être une police normale qui a été penchée électroniquement.

Les polices dont le nom contient "Oblique", "Slanted" ou "Incline" sont typiquement considérées comme 'oblique' dans la base de données de l'agent utilisateur et celles dont le nom contient *"Italic"*, *"Cursive"* ou *"Kursiv"* comme 'italic'.

```
H1, H2, H3 { font−style: italic }
H1 EM { font−style: normal }
```

Dans cet exemple, le texte accentué à l'intérieur de 'H1' prend un style normal.

5.2.4 'font−variant'

Valeur : normal | small−caps
Valeur initiale : normal
S'applique à : tous les éléments
Héritée : oui
Pourcentage : s.o.

Les petites capitales [ndt. small−caps] sont un autre genre de variation dans une famille de police. Les minuscules d'une police en petites capitales ressemblent aux capitales avec une taille réduite et des proportions légèrement différentes. La propriété 'font−variant' sélectionne une telle police.

La valeur 'normal' retient une police qui n'est pas en petites capitales et 'small−caps' une police en petites capitales. CSS1 tolère (mais n'oblige pas) que les petites capitales soient créées à partir des majuscules d'une police normale. Les majuscules se substituent aux petites capitales en dernier ressort.

Cet exemple montre un élément 'H3' en petites capitales avec des mots accentués en petites capitales italiques :

```
H3 { font−variant: small−caps }
EM { font−style: oblique }
```

D'autres variations sont possibles dans une famille de police, telles les polices avec des chiffres de style ancien, des chiffres en petites capitales, des lettres au style condensé ou étendu, etc. CSS1 n'a pas de propriétés pour cela.

CSS1 de base : dans la mesure où cette propriété change le texte en capitales, ces considérations sont aussi valables pour 'text−transform'.

5.2.5 'font−weight'

Valeur : normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
Valeur initiale : normal
S'applique à : tous les éléments
Héritée : oui
Pourcentage : s.o.

La propriété 'font−weight' donne la graisse [ndt. weight] de la police. Les valeurs de '100' à '900' forment une séquence ordonnée où chacune indique une graisse au moins aussi noire que la précédente. Le mot−clé 'normal' équivaut à '400' et 'bold' à '700'. Les mots−clés autres que ces derniers étaient souvent confondus avec le nom des polices, c'est pourquoi une échelle numérique a été préférée pour cette liste de 9 valeurs.

```
P { font−weight: normal }   /* 400 */
H1 { font−weight: 700 }     /* bold */
```

Les valeurs 'bolder' et 'lighter' donnent un résultat relatif à la graisse héritée du parent :

```
STRONG { font−weight: bolder }
```

Les enfants héritent de la graisse résultante et non de la valeur du mot−clé.

Les polices (les données) ont typiquement une ou plusieurs propriétés dont le nom est révélateur de leur "graisse". Il n'existe pas de nomenclature universellement acceptée pour les dénominations des graisses. Elles servent principalement à distinguer les déclinaisons des graisses d'une même police. L'usage est très variable selon les familles de police. Vous pourriez par exemple considérer un texte écrit en gras alors qu'on le dit *Regular, Roman, Book, Medium, Semi−* ou *DemiBold, Bold* ou *Black,* dans la mise en page en fonction de la graisse de la police "normale". En l'absence d'une dénomination standard, CSS1 indique des valeurs pour la propriété de graisse sur une échelle graduée dans laquelle la valeur '400' ('normal') correspond au dessin de la police normale dans la famille. On associe couramment à celle−ci le nom de graisse *Book, Regular, Roman, Normal* ou parfois *Medium.*

Les autres valeurs de graisse disponibles dans l'échelle numérique ne sont présentes que pour conserver une certaine graduation de la graisse dans une famille de police. Voici quelques pistes qui montrent comment traiter certains cas typiques :

- Si la famille de police utilise déjà une échelle de neuf valeurs (comme *OpenType*), les graisses devraient avoir une correspondance directe avec ces valeurs.
- S'il existe à la fois une police marquée *Medium* et une marquée *Book, Regular, Roman* ou *Normal,* alors on fait correspondre la *Medium* à '500'.
- La police marquée "Bold" correspond souvent à la valeur de graisse '700'.
- S'il y a moins de 9 graisses dans la famille, un algorithme par défaut peut combler les "manques" comme ceci : si '500' n'a pas de police correspondante, on fait alors correspondre cette graisse à la même police qui correspond déjà à '400'. Si une parmi les valeurs '600', '700', '800' ou '900' n'a pas de police correspondante, alors cette valeur partage la même police correspondante, si elle existe, que celle du mot−clé 'bolder', ou sinon, que celle qui correspond à 'lighter'. Si une parmi les valeurs '300', '200' ou '100' n'a pas de correspondance, alors cette valeur partage la même correspondance, si elle existe, que celle du mot−clé 'lighter', ou sinon, que celle qui correspond à 'bolder'.

Les deux exemples suivants illustrent le processus. Supposons, allant du plus maigre au plus gras, que la famille "Exemple1" dispose des quatre graisses : *Regular, Medium, Bold, Heavy*. Et la famille "Exemple2" des six graisses : *Book, Medium, Bold, Heavy, Black, ExtraBlack.* Pour la famille "Exemple2", noter que "Exemple2 ExtraBlack" dans le deuxième exemple *n'a pas* de correspondance.

```
 Polices disponibles  | Corresp.     |  En comblant
----------------------+--------------+------------------
"Exemple1 Regular"    | 400          |  100, 200, 300
"Exemple1 Medium"     | 500          |
"Exemple1 Bold"       | 700          |  600
"Exemple1 Heavy"      | 800          |  900

 Polices disponibles  | Corresp.     |  En comblant
----------------------+--------------+------------------
"Example2 Book"       | 400          |  100, 200, 300
"Example2 Medium"     | 500          |
"Example2 Bold"       | 700          |  600
"Example2 Heavy"      | 800          |
"Example2 Black"      | 900          |
"Example2 ExtraBlack" | (...)        |
```

L'action des mots−clés relatifs 'bolder' et 'lighter' a pour effet de d'épaissir ou d'amaigrir une police *dans une même famille*. Quand une famille n'a pas de correspondant pour chacune des valeurs de graisse numériques, 'bolder' va être assigné à la police de graisse supérieure disponible pour l'agent utilisateur, et 'lighter' à la police de graisse inférieure. Suit la signification précise des mots−clés relatifs 'bolder' et 'lighter' :

- 'bolder' prend la valeur de la graisse supérieure à celle qui est spécifiée. S'il manque la police correspondante, sa valeur prend simplement celle au−dessus dans l'échelle numérique (et la police ne

change pas d'aspect), à moins que cette valeur ne soit déjà à '900', valeur qui, auquel cas, est conservée.
- 'lighter' fonctionne de la même façon dans le sens inverse. La sélection se porte sur la police de graisse inférieure à celle qui est spécifiée, et, s'il manque la police correspondante, elle se fait sur la valeur numérique inférieure dans l'échelle (et la police ne change pas d'aspect).

L'existence d'une graisse plus épaisse pour toutes les valeurs de 'font−weight' n'est pas garantie. Par exemple, certaines familles de police n'ont que les graisses normale et grasse, d'autres peuvent avoir neuf graisses. On ne sait jamais comment un agent utilisateur va faire correspondre les polices d'une famille avec les valeurs de graisse. La seule garantie est qu'une police pour une valeur de graisse donnée ne sera pas moins grasse qu'une police de valeur de graisse moindre.

5.2.6 'font−size'

Valeur : <taille absolue> | <taille relative> | <longueur> | <pourcentage>
Valeur initiale : medium
S'applique à : tous les éléments
Héritée : oui
Pourcentage : relatif à la taille de police du parent

<taille absolue>
Le mot−clé <taille absolue> est un index pointant sur la table des tailles de police qui est calculée et maintenue par l'agent utilisateur. Les valeurs possibles : [xx−small | x−small | small | medium | large | x−large | xx−large]. Pour un écran d'ordinateur, il est suggéré un facteur d'échelle de 1.5 entre deux index adjacents ; si pour 'medium' la taille de la police fait 10pt, pour 'large', elle fait 15pt. Ce facteur nécessite sûrement une adaptation aux différents médias. L'agent utilisateur peut aussi considérer la qualité et la disponibilité des polices au moment de dresser cette table. Cette table peut être différente d'une famille de police à l'autre.

<taille relative>
Le mot−clé <taille relative> s'entend relatif à la table des tailles de police et à la taille de police du parent. Les valeurs possibles sont : [larger | smaller]. Par exemple, admettons que le parent ait une taille de police 'medium', la valeur 'larger' fera passer la taille de police de l'enfant à 'large'. Si la taille de la police du parent est trop éloignée d'une des entrées de la table, l'agent utilisateur est libre d'interpoler entre différentes entrées ou d'arrondir au plus proche. Il peut y être contraint si les mots−clés provoquent une sortie du champs de valeur de la table.

Les valeurs de longueur et de pourcentage ne devraient pas dépendre de la table des tailles de police pour le calcul de la taille de police d'un élément.

Les valeurs négatives sont interdites.

Pour toutes les autres propriétés, les valeurs en 'em' et 'ex' dépendent de la taille de police de l'élément même. Pour la propriété 'font−size', ces unités de longueur se réfèrent à la taille de police de l'élément parent.

Noter qu'une application peut redéfinir une taille explicite selon le contexte. Exemple, une scène en Réalité Virtuelle où la taille de la police peut varier avec la perspective.

Exemples :

```
P { font−size: 12pt; }
BLOCKQUOTE { font−size: larger }
EM { font−size: 150% }
EM { font−size: 1.5em }
```

Avec le facteur d'échelle 1.5 suggéré, les trois dernières déclarations sont identiques.

5.2.7 'font'

Valeur : [<font−style> || <font−variant> || <font−weight>]? <font−size> [/ <line−height>]? <font−family>
Valeur initiale : non défini pour les propriétés génériques
S'applique à : tous les éléments
Héritée : oui
Pourcentage : admis sur <font−size> et <line−height>

La propriété générique 'font' est un raccourci pour regrouper 'font−style' 'font−variant' 'font−weight' 'font−size', 'line−height' et 'font−family' dans la feuille de style. Sa syntaxe est basée sur une notation traditionnelle écourtée dans le but d'adresser plusieurs propriétés liées aux polices.

Pour leurs valeurs admises et leur valeur initiale, voir les propriétés déjà définies. Les propriétés dont aucune valeur n'apparaît prennent leur valeur initiale.

```
P { font: 12pt/14pt sans−serif }
P { font: 80% sans−serif }
P { font: x−large/110% "new century schoolbook", serif }
P { font: bold italic large Palatino, serif }
P { font: normal small−caps 120%/120% fantasy }
```

Dans la deuxième règle, la valeur exprimée en pourcentage se réfère à la taille de la police de l'élément parent. Dans la troisième règle, la hauteur de ligne exprimée en pourcentage se réfère à la taille de police de l'élément lui−même.

Dans les trois premières règles, les propriétés 'font−style', 'font−variant' et 'font−weight' ne sont pas mentionnées explicitement, cela signifie qu'elles gardent leur valeur initiale ('normal'). La quatrième règle spécifie implicitement les valeurs 'bold' pour 'font−weight', 'italic' pour 'font−style' et 'normal' pour 'font−variant'.

La cinquième règle fixe les valeurs des propriétés 'font−variant' ('small−caps'), 'font−size' (120% du parent), 'line−height' (120% de la taille de la police) et 'font−family' ('fantasy'). Leur valeur initiale qui est 'normal' est donc appliquée aux deux propriétés restantes, 'font−style' et 'font−weight'.

5.3 Propriétés de couleur et de fond

Ces propriétés décrivent la couleur (souvent appelé *couleur de premier plan*) et le fond d'un élément (c.−à−d., la surface sur laquelle le contenu est rendu). On peut donner une couleur et/ou une image au fond. On peut aussi préciser la position de cette image, si elle doit se répéter et comment, et dire si elle doit rester fixe ou défiler avec le canevas.

La propriété 'color' s'hérite normalement. Les propriétés de fond n'héritent pas, cependant le fond du parent sera visible par défaut car la propriété 'background−color' a pour valeur initiale 'transparent'.

5.3.1 'color'

Valeur : <couleur>
Valeur initiale : selon l'agent
S'applique à : tous les éléments
Héritée : oui
Pourcentage : s.o.

Cette propriété décrit la couleur du texte d'un élément. On peut spécifier la couleur rouge de plusieurs façons :

```
EM { color: red }              /* langage naturel */
EM { color: rgb(255,0,0) }     /* valeurs RGB de 0 à 255 */
```

Voir au chapitre 6.3 pour les valeurs de couleur possibles.

5.3.2 'background−color'

Valeur : <couleur> | transparent
Valeur initiale : transparent
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété fixe la couleur du fond d'un élément.

```
H1 { background−color: #F00 }
```

5.3.3 'background−image'

Valeur : <url> | none
Valeur initiale : none
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété précise l'image de fond d'un élément. Quand on a une image de fond, il est aussi souhaitable de donner une couleur de fond qui reste visible quand l'image n'est pas disponible. Celle ci s'affiche par dessus la couleur de fond dès qu'elle est disponible.

```
BODY { background−image: url(marble.gif) }
P { background−image: none }
```

5.3.4 'background−repeat'

Valeur : repeat | repeat−x | repeat−y | no−repeat
Valeur initiale : repeat
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Quand on a une image de fond, la propriété 'background−repeat' détermine si l'image doit se répéter et le cas échéant la manière dont elle se répète.

L'image est répétée à la fois dans les sens horizontal et vertical pour la valeur 'repeat', dans le sens horizontal pour la valeur 'repeat−x', ce qui produit une bande d'images d'un bord à l'autre. Idem pour 'repeat−y' dans le sens vertical. La valeur 'no−repeat' supprime toute répétition.

```
BODY {
  background: red url(pendant.gif);
  background−repeat: repeat−y;
}
```

Ici, l'image est répétée verticalement.

5.3.5 'background−attachment'

Valeur : scroll | fixed
Valeur initiale : scroll
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Si on a une image de fond, la valeur de 'background−attachment' détermine si celle−ci est fixe par rapport au canevas ('fixed') ou si elle défile en même temps que le contenu ('scroll').

```
BODY {
  background: red url(pendant.gif);
  background−repeat: repeat−y;
  background−attachment: fixed;
}
```

CSS1 de base : L'agent utilisateur peut substituer la valeur 'scroll' à 'fixed'. Néanmoins, 'fixed' doit être interprété correctement, au moins pour les éléments 'HTML' et 'BODY', l'auteur n'ayant aucun moyen de fournir une image uniquement aux navigateurs comprenant cette valeur. (Voir chapitre 7).

5.3.6 'background−position'

Valeur : [<pourcentage> | <longueur>]{1,2} | [top | center | bottom] || [left | center | right]
Valeur initiale : 0% 0%
S'applique à : ceux des éléments de type bloc et remplacés
Héritée : non
Pourcentage : relatif à la taille de l'élément lui−même

Si on a une image de fond, la valeur de 'background−position' en précise la position initiale.

Avec la paire de valeur '0% 0%', le coin en haut à gauche de l'image est aligné sur celui de la boîte qui entoure le contenu de l'élément (pas la boîte qui englobe l'espacement, la bordure et la marge). La paire de valeur '100% 100%' positionne le coin en bas à droite de l'image sur celui de l'élément. Avec la paire de valeur '14% 84%', le point situé 14% vers la droite et 84% vers le bas dans l'image est placé sur le point situé 14% vers la droite et 84% vers le bas dans l'élément.

Avec la paire de valeur '2cm 2cm', le coin en haut à gauche de l'image est placé 2cm vers la droite et 2cm sous le coin en haut à gauche de l'élément.

Si une seule valeur de pourcentage ou de longueur est fournie, celle−ci ne donne que la position horizontale, la position verticale sera 50%. Si on fournit deux valeurs, la première est la position horizontale. Les combinaisons de valeurs exprimées en longueur et en pourcentage sont admises, ex. '50% 2cm'. Les positions négatives ne le sont pas.

On peut aussi utiliser des mots−clés pour indiquer la position de l'image de fond. Les combinaisons avec des mots−clés et des valeurs en pourcentage ou longueur ne sont pas permises. Voici les combinaisons possibles de mots−clés et leurs interprétations :

- 'top left' et 'left top' <==> '0% 0%'.
- 'top', 'top center' et 'center top' <==> '50% 0%'.
- 'right top' et 'top right' <==> '100% 0%'.
- 'left', 'left center' et 'center left' <==> '0% 50%'.
- 'center' et 'center center' <==> '50% 50%'.
- 'right', 'right center' et 'center right' <==> '100% 50%'.
- 'bottom left' et 'left bottom' <==> '0% 100%'.
- 'bottom', 'bottom center' et 'center bottom' <==> '50% 100%'.
- 'bottom right' et 'right bottom' <==> '100% 100%'.

Exemples :

```
BODY { background: url(banner.jpeg) right top }    /* 100%   0% */
BODY { background: url(banner.jpeg) top center }   /*  50%   0% */
BODY { background: url(banner.jpeg) center }       /*  50%  50% */
BODY { background: url(banner.jpeg) bottom }       /*  50% 100% */
```

Si l'image de fond est fixe par rapport au canevas (cf. 'background−attachment' plus haut), l'image se place relativement à celui−ci et non à l'élément. Exemple :

```
BODY {
  background−image: url(logo.png);
  background−attachment: fixed;
  background−position: 100% 100%;
}
```

Ici, l'image bute dans le coin en bas à droite du canevas.

5.3.7 'background'

Valeur : <background−color> || <background−image> || <background−repeat> || <background−attachment> || <background−position>
Valeur initiale : non définie pour les propriétés génériques
S'applique à : tous les éléments
Héritée : non
Pourcentage : admis pour <background−position>

La propriété générique 'background' est un raccourci pour regrouper les propriétés de fond individuelles ('background−color', 'background−image', 'background−repeat', 'background−attachment' et 'background−position') dans la feuille de style.

Les valeurs de la propriété 'background' sont formées d'un jeu des valeurs possibles pour ces propriétés individuelles.

```
BODY { background: red }
P { background: url(chess.png) gray 50% repeat fixed }
```

La propriété 'background' fixe toutes les propriétés de fond individuelles. Dans l'exemple ci−dessus, la première règle spécifie juste une valeur pour 'background−color', les valeurs initiales des autres propriétés individuelles sont implicites. Dans la seconde règle, les valeurs sont spécifiées pour chacune des propriétés individuelles.

5.4 Propriétés du texte

5.4.1 'word−spacing'

Valeur : normal | <longueur>
Valeur initiale : normal
S'applique à : tous les éléments
Héritée : oui
Pourcentage : s.o.

La valeur indiquée par <longueur> s'ajoute à celle de l'espace normal entre les mots. Cette valeur peut être négative, mais son interprétation s'inscrira dans des limites spécifiques à l'agent utilisateur. Celui−ci est libre de calculer l'espace exact. L'espace−mot [ndt. word−spacing] peut aussi subir l'influence de la justification (qui est une valeur de la propriété 'text−align').

```
H1 { word-spacing: 1em }
```

Ici, l'espace−mot des éléments 'H1' est augmenté de 1em.

CSS1 de base : L'agent utilisateur peut substituer la valeur de longueur de 'word−spacing' par 'normal'. Voir chapitre 7.

5.4.2 'letter−spacing'

Valeur : normal | <longueur>
Valeur initiale : normal
S'applique à : tous les éléments
Héritée : oui
Pourcentage : s.o.

La valeur indiquée par <longueur> s'ajoute à l'interlettrage par défaut. Cette valeur peut être négative, mais son interprétation s'inscrira dans des limites spécifiques à l'agent utilisateur. L'agent utilisateur est libre de calculer l'espace exact. L'interlettrage peut subir l'influence de la justification (une des valeurs de la propriété 'text−align').

```
BLOCKQUOTE { letter-spacing: 0.1em }
```

Ici, l'interlettrage des éléments 'BLOCKQUOTE' est augmenté de 0.1em.

La valeur 'normal' autorise l'agent utilisateur à modifier l'interlettrage pour la justification du texte. Si 'letter−spacing' a une valeur de longueur, ce n'est plus le cas :

```
BLOCKQUOTE { letter-spacing: 0 }
BLOCKQUOTE { letter-spacing: 0cm }
```

L'agent utilisateur ne peut utiliser de ligatures si l'interlettrage a été modifié.

CSS1 de base : L'agent utilisateur peut interpréter une valeur de longueur de 'letter−spacing' par 'normal'. Voir chapitre 7.

5.4.3 'text−decoration'

Valeur : none | [underline || overline || line−through || blink]
Valeur initiale : none
S'applique à : tous les éléments
Héritée : non, voir explication plus bas
Pourcentage : s.o.

Cette propriété décrit les décorations ajoutées au texte d'un élément. Elle n'a aucun effet sur les éléments dépourvus de texte (ex. 'IMG' en HTML) ou sur les éléments vides (ex. '<EM></EM>'). La valeur 'blink' fait clignoter le texte.

La(les) couleur(s) requise(s) pour la décoration devrai(en)t être issue(s) de la valeur de la propriété 'color'.

La propriété ne s'hérite pas, cependant les éléments doivent tenir de leur parent. Un exemple, les enfants d'un élément souligné devraient aussi l'être. La couleur de soulignage doit se perpétuer même si les descendants ont d'autres valeurs pour 'color'.

```
A:link, A:visited, A:active { text−decoration: underline }
```

Dans l'exemple, le texte de tous les liens (les éléments 'A' avec un attribut 'HREF') est souligné.

Les agents utilisateurs doivent reconnaître le mot−clé 'blink' mais ne sont pas tenus d'en réaliser l'effet.

5.4.4 'vertical−align'

Valeur : baseline | sub | super | top | text−top | middle | bottom | text−bottom | <percentage>
Valeur initiale : baseline
S'applique à : les éléments en−ligne
Héritée : non
Pourcentage : selon la valeur de 'line−height' de l'élément lui−même

La propriété agit sur le positionnement vertical de l'élément. Un jeu de mots−clés est relatif à l'élément parent :

'baseline'
: aligne la ligne de base d'un élément (ou le bas, si l'élément n'a pas de ligne de base) sur celle de son parent.

'middle'
: aligne le milieu vertical d'un élément (en général une image) sur le milieu vertical de l'élément parent, c.à.d., sa ligne de base relevée de la moitié de sa hauteur.
: met l'élément en indice.

'super'
: met l'élément en exposant.

'text−top'
: aligne le sommet d'un élément avec celui du texte du parent.

'text−bottom'
: aligne le bas d'un élément avec le bas du texte du parent.

Un autre jeu est relatif à la ligne dont fait partie l'élément :

'top'
: aligne le sommet de l'élément avec celui de l'élément le plus haut dans la ligne.

'bottom'
: aligne le bas de l'élément avec celui de l'élément le plus bas dans la ligne.

L'utilisation conjointe de 'top' et 'bottom' peuvent conduire à des situations sans issues en cas de références circulaires.

Les valeurs en pourcentage se réfèrent à la valeur de la propriété 'line−height' de l'élément en question. Ces valeurs montent la ligne de base de élément (ou le bas, s'il n'en a pas) au−dessus de celle du parent de la quantité déterminée. Les valeurs négatives sont admises. Par exemple, une valeur de '−100%' abaisse l'élément de telle manière que sa ligne de base prend la place de celle de la ligne suivante. Ceci permet un contrôle précis du positionnement vertical des éléments qui n'ont pas de ligne de base (comme avec l'utilisation d'images en lieu de texte).

Une version ultérieure de CSS autorisera sans doute les valeurs de <longueur> pour cette propriété.

5.4.5 'text−transform'

Valeur : capitalize | uppercase | lowercase | none
Valeur initiale : none
S'applique à : tous les éléments
Héritée : oui
Pourcentage : s.o.

'capitalize'
: met la première lettre de chaque mot en majuscule

'uppercase'
: met en majuscule toutes les lettres de l'élément

'lowercase'
: met en minuscule toutes les lettres de l'élément

'none'
: neutralise la valeur héritée.

Dans chaque cas, la transformation effectuée dépend de la langue. Voir en [4] la façon de déterminer la langue pour un élément.

```
H1 { text−transform: uppercase }
```

Le contenu de 'H1' est en majuscule.

CSS1 de base : Les agents utilisateurs peuvent ignorer 'text−transform' et le traiter comme 'none' pour les caractères qui ne font pas partie du répertoire Latin−1 ou pour les éléments avec des langues dont la transformation serait différente par rapport à celle prévue dans les tables de conversion Unicode [8].

5.4.6 'text−align'

Valeur : left | right | center | justify
Valeur initiale : selon l'agent utilisateur
S'applique à : ceux des éléments de type bloc
Héritée : oui
Pourcentage : s.o.

Cette propriété décrit l'alignement du texte d'un élément. L'algorithme de justification appliqué dépend de l'agent utilisateur et de la langue.

Exemple :

```
DIV.center { text−align: center }
```

Comme la propriété 'text−align' est héritée, toutes les boîtes de ligne de tous les éléments de 'DIV' de type bloc dont la classe est 'center' sont centrés. Noter que l'alignement est appliqué relativement à la largeur de l'élément et non au canevas. L'agent utilisateur doit remplacer la valeur 'justify' s'il ne reconnaît pas celle−ci. C'est en général 'left' pour les langues occidentales.

CSS1 de base : L'agent utilisateur peut interpréter 'justify' respectivement comme 'left' ou 'right' selon que le sens d'écriture est de la gauche vers la droite ou de la droite vers la gauche.

5.4.7 'text−indent'

Valeur : <longueur> | <pourcentage>
Valeur initiale : 0
S'applique à : ceux des éléments de type bloc
Héritée : oui
Pourcentage : relatif à la largeur du parent

La propriété spécifie l'alinéa [ndt. indent] appliqué sur la première ligne formatée. La valeur de 'text−indent peut être négative, mais son interprétation est soumise aux limites de tel ou tel agent utilisateur. L'alinéa ne peut apparaître au milieu d'un élément coupé par un autre (ainsi 'BR' en HTML).

Exemple:

```
P { text−indent: 3em }
```

5.4.8 'line−height'

Valeur : normal | <nombre> | <longueur> | <pourcentage>
Valeur initiale : normal
S'applique à : tous les éléments
Héritée : oui
Pourcentage : relatif à la taille de la police de l'élément lui−même

Cette propriété fixe l'écart entre les lignes de base de deux lignes adjacentes.

Si on spécifie une valeur numérique, la hauteur de ligne est obtenue en multipliant la taille de police de l'élément en question avec cette valeur−là. Ce qui est différent d'une valeur en pourcentage qui est héritée car les enfants héritent de la valeur numérique elle−même et non pas de la valeur résultante (comme c'est le cas pour le pourcentage et d'autres unités).

On n'admet pas de valeurs négatives.

Les trois règles suivantes donnent une même hauteur de ligne :

```
DIV { line−height: 1.2; font−size: 10pt }     /* nombre */
DIV { line−height: 1.2em; font−size: 10pt }   /* longueur */
DIV { line−height: 120%; font−size: 10pt }    /* pourcentage */
```

La valeur 'normal' met la hauteur de ligne dans un rapport raisonnable avec la police de l'élément. De préférence, l'agent utilisateur établit une valeur dans un rapport de 1.0 à 1.2.

Voir le chapitre 4.4 pour une description de l'influence de 'line−height' sur le format d'un élément de type bloc.

5.5 Propriétés de boîte

Ces propriétés fixent la taille, les contours et la position des boîtes qui représentent les éléments. Voir le modèle de mise en forme (chapitre 4) pour des exemples d'application.

Les propriétés de marge spécifient les marges d'un élément. La propriété 'margin' fixe les marges sur les quatre bords, les propriétés de marge spécifiques fixent leur côté respectif.

Les propriétés d'espacement spécifient l'espace entre la bordure et le contenu (ex. texte ou image). La propriété 'padding' fixe l'espacement sur les quatre bords, les propriétés d'espacement spécifiques de leur côté respectif.

Les propriétés de bordure spécifient les bordures d'un propriété. Chaque élément possède quatre bordures définies par leur largeur, couleur et style.

Les propriétés 'width' et 'height' spécifient la taille de la boîte. Les propriétés 'float' et 'clear' peuvent modifier la position des éléments.

5.5.1 'margin−top'

Valeur : <longueur> | <pourcentage> | auto
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne la marge du haut d'un élément :

```
H1 { margin−top: 2em }
```

On admet une valeur négative, mais son interprétation est soumise aux limites de l'agent utilisateur.

5.5.2 'margin−right'

Valeur : <longueur> | <pourcentage> | auto
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne la marge de droite d'un élément :

```
H1 { margin−right: 12.3% }
```

On admet une valeur négative, mais son interprétation est soumise aux limites de l'agent utilisateur.

5.5.3 'margin−bottom'

Valeur : <longueur> | <pourcentage> | auto
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne la marge du bas d'un élément :

```
H1 { margin−bottom: 3px }
```

On admet une valeur négative, mais son interprétation est soumise aux limites de l'agent utilisateur.

5.5.4 'margin−left'

Valeur : <longueur> | <pourcentage> | auto
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne la marge de gauche d'un élément :

```
H1 { margin−left: 2em }
```

On admet une valeur négative, mais son interprétation est soumise aux limites de l'agent utilisateur.

5.5.5 'margin'

Valeur : [<longueur> | <pourcentage> | auto]{1,4}
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

La propriété 'margin' est un raccourci pour fixer et regrouper les propriétés 'margin−top', 'margin−right', 'margin−bottom' et 'margin−left' dans la feuille de style.

Si on spécifie quatre valeurs de longueur, celles−ci correspondent respectivement aux marges haute, droite, basse et gauche. Si on ne spécifie qu'une valeur, elle s'applique aux quatre côtés et s'il y en a deux ou trois, on déduit les valeurs manquantes de celles des côtés opposés.

```
BODY { margin: 2em }         /* toutes les marges ont 2em */
BODY { margin: 1em 2em }     /* haut et bas = 1em, droite et gauche = 2em */
BODY { margin: 1em 2em 3em } /* haut = 1em, droite = 2em, bas = 3em, gauche = 2em */
```

La dernière règle de cet exemple est équivalente à :

```
BODY {
  margin−top: 1em;
  margin−right: 2em;
  margin−bottom: 3em;
  margin−left: 2em;    /* pris du côté opposé (droit) */
}
```

On admet les valeurs négatives, mais leur interprétation est soumise aux limites de l'agent utilisateur.

5.5.6 'padding−top'

Valeur : <longueur> | <pourcentage>
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne l'espacement du haut d'un élément.

```
BLOCKQUOTE { padding−top: 0.3em }
```

Les valeurs d'espacement ne sont jamais négatives.

5.5.7 'padding−right'

Valeur : <longueur> | <pourcentage>
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non

Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne l'espacement de droite d'un élément.

```
BLOCKQUOTE { padding−right: 10px }
```

Les valeurs d'espacement ne sont jamais négatives.

5.5.8 'padding−bottom'

Valeur : <longueur> | <pourcentage>
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne l'espacement du bas d'un élément.

```
BLOCKQUOTE { padding−bottom: 2em }
```

Les valeurs d'espacement ne sont jamais négatives.

5.5.9 'padding−left'

Valeur : <longueur> | <pourcentage>
Valeur initiale : 0
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

Cette propriété concerne l'espacement de gauche d'un élément.

```
BLOCKQUOTE { padding−left: 20% }
```

Les valeurs d'espacement ne sont jamais négatives.

5.5.10 'padding'

Valeur : [<longueur> | <pourcentage>]{1,4}
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Héritée : non
Pourcentage : relatif à la dimension du parent de type bloc le plus proche

La propriété 'padding' est un raccourci pour fixer et regrouper les propriétés 'padding−top', 'padding−right', 'padding−bottom' et 'padding−left' dans la feuille de style.

Si on spécifie quatre valeurs de longueur, celles−ci correspondent respectivement aux espacements haut, droit, bas et gauche. Si on ne spécifie qu'une valeur, elle s'applique aux quatre côtés et s'il y en a deux ou trois, on déduit les valeurs manquantes de celles des côtés opposés.

Le fond de la surface d'espacement est spécifié par la propriété 'background' :

```
H1 {
  background: white;
  padding: 1em 2em;
}
```

Cet exemple montre un espacement vertical de 1em ('padding−top' et 'padding−bottom') et un espacement horizontal de 2em ('padding−right' et 'padding−left'). L'unité 'em' est relative à la taille de la police de l'élément, ainsi '1em' correspond à la taille de cette police.

Les valeurs d'espacement ne sont jamais négatives.

5.5.11 'border−top−width'

Valeur : thin | medium | thick | <longueur>
Valeur initiale : 'medium'
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété spécifie l'épaisseur de la bordure du haut d'un élément. La dimension donnée par les mots−clés dépend de l'agent utilisateur, mais ceci est toujours vérifié : 'thin' <= 'medium' <= 'thick'.

Les valeurs issues des mots−clés sont constantes pour un document :

```
H1 { border: solid thick red }
P  { border: solid thick blue }
```

Ici, les éléments 'H1' et 'P' ont une même épaisseur indépendamment de la taille de la police. On peut obtenir des dimensions relatives en utilisant l'unité 'em' :

```
H1 { border: solid 0.5em }
```

Les bordures n'ont pas de valeurs négatives.

5.5.12 'border−right−width'

Valeur : thin | medium | thick | <longueur>
Valeur initiale : 'medium'
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété spécifie l'épaisseur de la bordure de droite d'un élément. Sinon, elle équivaut à 'border−top−width'.

5.5.13 'border−bottom−width'

Valeur : thin | medium | thick | <length>
Valeur initiale : 'medium'
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété spécifie l'épaisseur de la bordure du bas d'un élément. Sinon, elle équivaut à 'border−top−width'.

5.5.14 'border−left−width'

Valeur : thin | medium | thick | <length>
Valeur initiale : 'medium'
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété spécifie l'épaisseur de la bordure de gauche d'un élément. Sinon, elle équivaut à 'border−top−width'.

5.5.15 'border−width'

Valeur : [thin | medium | thick | <longueur>]{1,4}
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Pourcentage : s.o.

Cette propriété est un raccourci pour fixer et regrouper 'border−top−width', 'border−right−width', 'border−bottom−width' et 'border−left−width' dans la feuille de style.

Elle se compose d'une à quatre valeurs qu'on interprète ainsi :

- une valeur : les quatre bordures ont la même épaisseur
- deux valeurs : la première valeur est attribuée aux bordures du haut et du bas, la deuxième valeur à celles de droite et de gauche
- trois valeurs : la première pour le haut, la deuxième pour la droite et la gauche la troisième pour le bas
- quatre valeurs : le haut, la droite, le bas et la gauche respectivement.

Dans cet exemple, les commentaires indiquent les valeurs attribuées dans l'ordre haut, droite, bas et gauche :

```
H1 { border−width: thin }                  /* thin thin thin thin */
H1 { border−width: thin thick }            /* thin thick thin thick */
H1 { border−width: thin thick medium }     /* thin thick medium thin */
H1 { border−width: thin thick medium thin } /* thin thick medium thin */
```

Les bordures n'ont pas de valeurs négatives.

5.5.16 'border−color'

Valeur : <couleur>{1,4}
Valeur initiale : la valeur de la propriété 'color'
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

La propriété 'border−color' spécifie la couleur des quatre bordures. Elle se compose d'une à quatre valeurs de la même façon que pour 'border−width' au−dessus.

Si on ne spécifie aucune couleur, 'border−color' prend la valeur de la propriété 'color' de l'élément lui−même :

```
P {
  color: black;
  background: white;
  border: solid;
}
```

Ici, la bordure forme un cadre noir.

5.5.17 'border−style'

Valeur : none | dotted | dashed | solid | double | groove | ridge | inset | outset
Valeur initiale : none
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

La propriété 'border−color' spécifie le style des quatre bordures. Elle se compose d'une à quatre valeurs de la même façon que pour 'border−width' au−dessus.

```
#xy34 { border−style: solid dotted }
```

Ici, les bordures horizontales sont en trait plein et les verticales en trait pointillé.

La valeur initiale de la propriété étant 'none', les bordures sont invisibles tant qu'aucune autre valeur n'est spécifiée.

Les valeurs de style de bordure signifient :

none
: aucune bordure n'apparaît (indépendamment de la valeur de 'border−width')

dotted
: la bordure est une ligne pointillée tracée sur le fond de l'élément

dashed
: la bordure est une ligne en tirets tracée sur le fond de l'élément

solid
: la bordure est un trait plein

double
: la bordure est un trait double tracé sur le fond de l'élément. La somme des épaisseurs des deux traits simples et de l'espace entre eux est égale à la valeur de <border−width>

groove
une cannelure en 3D est dessinée d'après la valeur de <color>.
ridge
une strie en 3D est dessinée d'après la valeur de <color>.
inset
une incrustation en 3D est dessinée d'après la valeur de <color>.
outset
une extrusion en 3D est dessinée d'après la valeur de <color>.

CSS1 de base : L'agent utilisateur peut interpréter 'dotted', 'dashed', 'double', 'groove', 'ridge', 'inset' et 'outset' comme un trait plein ('solid').

5.5.18 'border−top'

Valeur : <border−top−width> || <border−style> || <couleur>
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété est un raccourci pour spécifier l'épaisseur, le style et la couleur de la bordure du haut d'un élément.

```
H1 { border−top: thick solid red }
```

La règle ci−dessus indique l'épaisseur, le style et la couleur de la bordure au−dessus de 'H1'. Les valeurs omises prennent leur valeur initiale :

```
H1 { border−top: thick solid }
```

Comme ici la valeur de la couleur est omise, la bordure prend la valeur de 'color' de l'élément lui−même.

Noter que cette propriété n'a qu'une seule valeur de style alors que 'border−style' en accepte jusqu'à quatre.

5.5.19 'border−right'

Valeur : <border−right−width> || <border−style> || <couleur>
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété est un raccourci pour spécifier l'épaisseur, le style et la couleur de la bordure de droite d'un élément. Sinon, elle équivaut à 'border−top'.

5.5.20 'border−bottom'

Value: <border−bottom−width> || <border−style> || <color>
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété est un raccourci pour spécifier l'épaisseur, le style et la couleur de la bordure du bas d'un élément. Sinon, elle équivaut à 'border−top'.

5.5.21 'border−left'

Valeur : <border−left−width> || <border−style> || <couleur>
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété est un raccourci pour spécifier l'épaisseur, le style et la couleur de la bordure de gauche d'un élément. Sinon, elle équivaut à 'border−top'.

5.5.22 'border'

Valeur : <border−width> || <border−style> || <couleur>
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : tous les éléments
Pourcentage : s.o.

La propriété 'border' est un raccourci pour donner la même valeur d'épaisseur, de couleur et de style aux quatre bordures d'un élément. Ainsi dans l'exemple, la première règle produit le même effet que le jeu de quatre règles qui la suit :

```
P { border: solid red }
P {
  border−top: solid red;
  border−right: solid red;
  border−bottom: solid red;
  border−left: solid red
}
```

À l'inverse des propriétés raccourcies 'margin' et 'padding', 'border' ne peut pas donner des valeurs différentes aux quatre bordures. Pour ce faire, il faut utiliser une ou plusieurs des autres propriétés de bordure.

Les fonctions de ces propriétés peuvent se recouper, aussi l'ordre d'écriture des règles devient important. Considérons ce qui suit :

```
BLOCKQUOTE {
  border−color: red;
  border−left: double;
  color: black;
}
```

La bordure gauche est noire tandis que les autres ne sont pas visibles. Ceci vient de 'border−left' qui, la valeur de la couleur n'étant pas spécifiée, va prendre la valeur de la propriété 'color'. Le fait que celle−ci soit fixée après la propriété 'border−left' ne fait pas de différence.

La propriété 'border' n'accepte qu'une seule valeur, bien que 'border−width' puisse en avoir jusqu'à quatre.

5.5.23 'width'

Valeur : <longueur> | <pourcentage> | auto
Valeur initiale : auto
S'applique à : ceux des éléments de type bloc et remplacés
Héritée : non
Pourcentage : relatif à la largeur du parent

On peut appliquer cette propriété au texte, elle est cependant bien plus utile avec les éléments remplacés comme les images. Si nécessaire, on peut contraindre la largeur d'une image. Dans un changement d'échelle, le ratio de l'image rendue est conservé si la propriété 'height' a la valeur 'auto'.

Exemple :

```
IMG.icon { width: 100px }
```

Pour un élément remplacé, les dimensions intrinsèques de l'objet qui le remplace sont assignées aux propriétés 'width' et 'height' si leur valeur initiale était 'auto'.

Les valeurs négatives ne sont pas admises.

Voir le modèle de mise en forme (chapitre 4) pour les relations existants entre cette propriété et la marge et l'espacement.

5.5.24 'height'

Valeur : <longueur> | auto
Valeur initiale : auto
S'applique à : ceux des éléments de type bloc et remplacés
Héritée : non
Pourcentage : s.o.

On peut appliquer cette propriété au texte, elle est cependant bien plus utile avec les éléments remplacés comme les images. Si nécessaire, on peut contraindre la hauteur d'une image. Dans un changement d'échelle, le ratio de l'image rendue est conservé si la propriété 'width' a la valeur 'auto'.

Exemple :

```
IMG.icon { height: 100px }
```

Quand les valeurs des propriétés 'width' et 'height' d'un élément remplacé sont à leur valeur par défaut 'auto', celles−ci adoptent les dimensions intrinsèques de l'objet qui le remplace.

On peut l'appliquer à un élément textuel et ainsi contraindre sa hauteur (ex. barre de défilement).

Les valeurs négatives ne sont pas admises.

CSS1 de base : L'agent utilisateur peut ignorer la propriété 'height' (c.−à−d., considérer sa valeur étant 'auto'), si elle est appliquée à un élément qui ne soit pas remplacé.

5.5.25 'float'

Valeur : left | right | none
Valeur initiale : none
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Pour une valeur 'none', l'élément s'affiche dans le flux normal du texte. Pour une valeur 'left' (ou 'right'), l'élément se place sur la gauche (ou droite) et le texte épouse son côté droit (ou gauche). Avec les valeurs 'left' ou 'right', l'élément acquiert un type bloc (c.−à−d. que la propriété 'display' est ignorée). Voir le chapitre 4.1.4 pour une explication complète.

```
IMG.icon {
  float: left;
  margin−left: 0;
}
```

Dans cet exemple, les éléments 'IMG' avec un attribut 'CLASS="icon"' sont positionnés au côté gauche de leur parent.

On utilise le plus souvent 'float' avec des images en ligne, mais on l'applique aussi aux éléments avec du texte.

5.5.26 'clear'

Valeur : none | left | right | both
Valeur initiale : none
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété spécifie si un élément accepte des éléments flottants contre ses bords. Plus précisément, la valeur de la propriété indique les côtés contre lesquels ceux−ci ne peuvent venir. L'élément dont la propriété 'clear' a la valeur 'left' ira se placer en−dessous de tout élément flottant qui serait à sa gauche. Avec une valeur 'none', les éléments flottants sont admis sur chacun de ses côtés. Exemple :

```
H1 { clear: left }
```

5.6 Propriétés de classification

Ces propriétés ont plus d'intérêt pour classer des éléments en catégories que pour la spécification de paramètres visuels.

Les propriétés de liste décrivent la mise en forme des items d'une liste (les éléments avec une valeur 'list−item' pour la propriété 'display'). Ces propriétés s'appliquent à tout élément, leurs descendants en héritent mais uniquement ceux dont la propriété 'display' a pour valeur 'list−item'. En HTML, il s'agit typiquement de l'élément 'LI'.

5.6.1 'display'

Valeur : block | inline | list−item | none
Valeur initiale : block
S'applique à : tous les éléments
Héritée : non
Pourcentage : s.o.

Cette propriété spécifie si et comment un élément doit s'afficher dans le canevas (que ce soit une page, un moniteur, etc.).

Un élément dont la valeur pour 'display' est 'block' crée une nouvelle boîte. Celle−ci se positionne par rapport aux boîtes adjacentes selon le modèle de mise en forme de CSS. Les éléments' 'H1' et 'P' ont typiquement la valeur 'block'. La valeur 'list−item' confère un comportement similaire à celui de la valeur 'block', à la différence qu'une marque de liste se rajoute. En HTML, c'est la valeur typique de l'élément 'LI'.

Un élément dont la valeur pour 'display' est 'inline' crée une boîte dans le prolongement de la ligne du contenu qui précède. La boîte prend la dimension voulue pour la mise en forme de son contenu. Un contenu textuel peut s'étendre sur plusieurs lignes avec autant de boîtes de ligne. Les propriétés de marge, bordure et espacement s'appliquent aux éléments ainsi créés mais n'ont aucun effet à l'endroit des coupures de ligne.

La valeur 'none' supprime l'affichage d'un élément et de sa boîte ainsi que des éléments qu'il contient.

```
P { display: block }
EM { display: inline }
LI { display: list-item }
IMG { display: none }
```

La dernière règle supprime l'affichage des images.

La valeur initiale de 'display' est 'block', cependant l'agent utilisateur peut n'appliquer que des valeurs par défaut typiques à chacun des éléments HTML, en accord avec leur rendu suggéré dans la spécification du HTML [2].

CSS1 de base : L'agent utilisateur peut ignorer 'display' et n'utiliser que ses valeurs par défaut (voir le chapitre 7).

5.6.2 'white−space'

Valeur : normal | pre | nowrap
Valeur initiale : normal
S'applique à : ceux des éléments de type bloc
Héritée : oui
Pourcentage : s.o.

Cette propriété précise la façon dont sont traités les blancs dans un élément : ainsi la valeur 'normal' (les blancs fusionnent), 'pre' (qui se comporte comme l'élément 'PRE' de HTML) ou 'nowrap' (la mise à la ligne a lieu uniquement avec des éléments 'BR') :

```
PRE { white-space: pre }
P   { white-space: normal }
```

La valeur initiale de 'white−space' est 'normal', cependant l'agent utilisateur peut n'appliquer que des valeurs par défaut typiques à chacun des éléments HTML, en accord avec leur rendu suggéré dans la spécification du HTML [2].

CSS1 de base : L'agent utilisateur peut ignorer 'white−space' dans la feuille de style de l'auteur ou du lecteur et n'utiliser que des valeurs par défaut à la place (voir le chapitre 7).

5.6.3 'list−style−type'

Valeur : disc | circle | square | decimal | lower−roman | upper−roman | lower−alpha | upper−alpha | none
Valeur initiale : disc
S'applique à : ceux des éléments dont la propriété 'display' a la valeur 'list−item'
Héritée : oui
Pourcentage : s.o.

Cette propriété détermine l'apparence de la marque de liste dans le cas où la propriété 'list−style−image' a pour valeur 'none' ou quand l'image désignée par celle−ci n'est pas disponible.

```
OL { list-style-type: decimal }       /* 1 2 3 4 5 etc. */
OL { list-style-type: lower-alpha }   /* a b c d e etc. */
OL { list-style-type: lower-roman }   /* i ii iii iv v etc. */
```

5.6.4 'list−style−image'

Valeur : <url> | none
Valeur initiale : none
S'applique à : ceux des éléments dont la propriété 'display' a la valeur 'list−item'
Héritée : oui
Pourcentage : s.o.

Cette propriété désigne l'image utilisée comme marque de liste. Si l'image est disponible, elle remplace le type de marque spécifié dans 'list−type−marker'.

```
UL { list-style-image: url(http://png.com/ellipse.png) }
```

5.6.5 'list−style−position'

Valeur : inside | outside
Valeur initiale : outside
S'applique à : ceux des éléments dont la propriété 'display' a la valeur 'list−item'
Héritée : oui
Pourcentage : s.o.

La valeur de 'list−style−position' détermine la position de la marque de liste en regard de l'item en question. Voir un exemple de mise en forme au chapitre 4.1.3.

5.6.6 'list−style'

Valeur : [disc | circle | square | decimal | lower−roman | upper−roman | lower−alpha | upper−alpha | none] || [inside | outside] || [<url> | none]
Valeur initiale : non définie pour les propriétés raccourcies
S'applique à : ceux des éléments dont la propriété 'display' a la valeur 'list−item'
Héritée : oui
Pourcentage : s.o.

La propriété 'list−style' est un raccourci pour fixer et regrouper les trois propriétés 'list−style−type', 'list−style−image' et 'list−style−position' dans la feuille de style.

```
UL { list-style: upper-roman inside }
UL UL { list-style: circle outside }
LI.square { list-style: square }
```

Des résultats inattendus peuvent survenir quand on applique 'list−style' directement sur l'élément 'LI'. Ainsi :

```
<STYLE TYPE="text/css">
  OL.alpha LI  { list-style: lower-alpha }
  UL LI        { list-style: disc }
</STYLE>
<BODY>
  <OL CLASS=alpha>
    <LI>niveau 1
    <UL>
       <LI>niveau 2
    </UL>
  </OL>
</BODY>
```

La spécificité (telle que définie dans l'ordre de cascade) étant plus grande pour la première règle, celle−ci annule la deuxième règle pour tous les éléments 'LI', seule la valeur 'lower−alpha' est retenue. Il est donc recommandé de n'appliquer 'list−style' que sur les éléments de liste eux−mêmes :

```
OL.alpha  { list-style: lower-alpha }
UL        { list-style: disc }
```

Ci−dessus, les éléments 'LI' héritent des valeurs de 'list−style' des éléments 'OL' et 'UL'.

On peut combiner une valeur d'URL avec chacune des autres valeurs :

```
UL { list-style: url(http://png.com/ellipse.png) disc }
```

Dans cet exemple, la valeur 'disc' est choisie si l'image n'est pas disponible.

6 Unités

6.1 Unités de longueur

Une valeur de longueur est formée d'un caractère optionnel ('+' ou '−', '+' par défaut) immédiatement suivi par un nombre décimal puis immédiatement encore par un identifiant d'unité (abrégé en deux lettres). L'identifiant d'unité est facultatif pour une valeur 0.

Quelques propriétés admettent des unités de longueur négatives, ce qui peut compliquer la mise en forme et qui dépend des limites propres à un agent utilisateur. Si une valeur de longueur négative n'est pas supportée, il faudrait la ramener à la plus proche valeur qui le soit.

Il y a deux genres d'unités de longueur, relatives et absolues. Les relatives se rapportent à une autre propriété de longueur. Les feuilles de styles basées sur ce genre−ci vont s'adapter plus facilement d'un média à un autre (p.ex. d'un moniteur vers une imprimante laser). Les unités de pourcentage (décrites plus bas) ainsi que les valeurs de mot−clé (ex. 'x−large') offrent les mêmes avantages.

Voici les unités relatives supportées :

```
H1 { margin: 0.5em }      /* em : la taille de la police de l'élément */
H1 { margin: 1ex }        /* x-height : la hauteur de la lettre 'x' de la police de l'élément */
P  { font-size: 12px }    /* pixel : relative au canevas */
```

Les unités relatives 'em' et 'ex' sont relatives à la taille de la police de l'élément lui−même. La seule exception à cette règle de CSS1 est la propriété 'font−size' où les valeurs en 'em' et 'ex' sont calculées à partir de l'élément parent.

Les valeurs en pixels, telle que dans la dernière règle, sont relatives à la résolution du canevas, c.−à−d. le plus souvent celle d'un moniteur. Si la densité en pixels d'un périphérique de sortie est trop différente de celle d'un moniteur habituel, l'agent utilisateur doit adapter l'échelle des valeurs de pixel. On suggère un *pixel de référence* calculé ainsi : c'est l'angle visuel formé par un pixel sur un appareil de densité 90dpi (ppp) situé à une longueur de bras. Ce qui donne, pour une longueur nominale de bras de 28 pouces (71 cm), un angle de 0.0227 degrés.

L'élément enfant hérite de la valeur calculée, non de la valeur relative :

```
BODY {
  font-size: 12pt;
  text-indent: 3em;  /* c.-à-d. 36pt */
}
H1 { font-size: 15pt }
```

Dans l'exemple précédent, la valeur de 'text−indent' pour 'H1' est de 36pt et non 45pt.

Les unités de longueur absolues ne sont vraiment utiles que si on connaît les propriétés physiques du média de sortie. Voici celles qui sont supportées :

```
H1 { margin: 0.5in }      /* inch (pouce) 1in = 2.54cm */
H2 { line-height: 3cm }   /* centimètre */
H3 { word-spacing: 4mm }  /* millimètre */
H4 { font-size: 12pt }    /* point 1pt = 1/72in */
H4 { font-size: 1pc }     /* pica 1pc = 12pt */
```

Si l'agent utilisateur ne supporte pas les longueurs spécifiées, il doit essayer d'en calculer une valeur par approximation. Pour chacune des propriétés CSS1, les calculs ultérieurs ainsi que l'héritage doivent s'appuyer sur cette valeur.

6.2 Unités de pourcentage

Une valeur en pourcentage est formée d'un caractère optionnel ('+' ou '−', '+' par défaut) immédiatement suivi par un nombre avec ou sans point décimal puis immédiatement encore par '%'.

Les valeurs de pourcentage sont toujours relatives à une autre valeur, par exemple une unité de longueur. Chacune des propriétés admettant une valeur en pourcentage définit aussi la valeur à laquelle se rapporte celui−ci. La plupart du temps, c'est la taille de la police de l'élément lui−même :

```
P { line−height: 120% }   /* 120% de la propriété 'font−size' de 'P' */
```

En CSS1 pour toutes les propriétés héritées, si leur valeur est spécifiée en pourcentage, les enfants héritent de la valeur résultante et non du pourcentage.

6.3 Unités de couleur

Une couleur est représentée soit par un mot−clé soit par une valeur numérique de RGB.

Voici la liste suggérée des mots−clés pour les couleurs : 'aqua', 'black', 'blue', 'fuchsia', 'gray', 'green', 'lime', 'maroon', 'navy', 'olive', 'purple', 'red', 'silver', 'teal', 'white' et 'yellow'. Ils sont issus de la palette VGA de Windows, leurs valeurs RGB ne sont pas définies dans la présente spécification.

```
BODY {color: black; background: white }
H1 { color: maroon }
H2 { color: olive }
```

Le modèle de couleur RGB est utilisé pour les valeurs de couleur numériques. Ces règles produisent la même couleur :

```
EM { color: #f00 }              /* #rgb */
EM { color: #ff0000 }           /* #rrggbb */
EM { color: rgb(255,0,0) }      /* un entier de 0 à 255 */
EM { color: rgb(100%, 0%, 0%) } /* un pourcentage de 0.0% à 100.0% */
```

En notation hexadécimale, le format d'une valeur RGB se compose d'un '#' immédiatement suivi par trois ou bien six caractères hexadécimaux. La notation en trois caractères (#rgb) est convertie vers la forme en six caractères (#rrggbb) par réplication et non par l'ajout de zéros. Ainsi '#fb0' devient '#ffbb00'. Ceci assure la notation du blanc (#ffffff) en trois caractères (#fff) et en supprime la dépendance avec la profondeur de couleur de l'écran.

En notation fonctionnelle, le format d'une valeur RGB se compose de 'rgb(' suivi d'une séquence de trois valeurs numériques séparée par des virgules (soit trois entiers ayant des valeurs de 0 à 255, soit trois valeurs de pourcentage de 0 à 100%) avec un ')' final. Les blancs sont admis avant ou après les valeurs numériques.

On ramène les valeurs numériques hors−limites dans la plage autorisée. Ainsi, les trois règles suivantes sont équivalentes :

```
EM { color: rgb(255,0,0) }       /* entier dans la plage 0 − 255 */
EM { color: rgb(300,0,0) }       /* ramené à 255 */
EM { color: rgb(110%, 0%, 0%) }  /* ramené à 100% */
```

Les couleurs RGB sont spécifiées dans l'espace de couleur sRGB [9]. La conformité des agents utilisateurs dans la représentation des couleurs est variable, aussi l'utilisation de l'espace sRGB en fournit une définition objective et sans ambiguïté qui est liée à des standards internationaux [10].

Les agents utilisateurs peuvent se limiter à effectuer une correction du gamma sur les couleurs. sRGB spécifie un gamma de 2.2 pour un écran selon certaines conditions. L'agent utilisateur ajuste les couleurs spécifiées en CSS en relation avec le gamma produit "naturellement" par un dispositif d'affichage pour obtenir un gamma effectif de 2.2. L'Appendice D donne plus de détails. Noter que seules les couleurs spécifiées dans la feuille de style sont affectées, ainsi les images sont sensées contenir leurs propres informations de couleur.

6.4 URL

On identifie un URL (Uniform Resource Locator) par une notation fonctionnelle :

```
BODY { background: url(http://www.bg.com/pinkish.gif) }
```

Le format d'une valeur d'URL se compose de la chaîne 'url(' suivie d'un blanc optionnel puis d'un guillemet simple (') ou double (") optionnel aussi, vient ensuite l'URL proprement dit (tel que défini dans [11]), puis à nouveau les guillemets simple ou double éventuels, un blanc optionnel et la parenthèse ')' finale. On doit apparier les marques de citation qui ne font pas partie de l'URL lui−même.

On doit échapper les parenthèses (()), les virgules (,), les blancs ( ) et les guillemets simples (') ou doubles (") contenus dans un URL à l'aide du caractère oblique inverse : '\(', '\)', '\,', etc.

Un URL partiel s'entend relativement à la source de la feuille de style et non au document :

```
BODY { background: url(yellow.png) }
```

7 Conformité au CSS1

Pour être déclaré conforme à la spécification CSS1, un agent utilisateur qui se sert de CSS1 pour le rendu d'un document doit :

- charger toutes les feuilles référencées dans le document et les interpréter en accord avec cette spécification
- classer les déclarations selon l'ordre de cascade
- utiliser les fonctionnalités de CSS1 dans les limites imposées par le média (voir l'explication plus bas).

Un agent utilisateur qui produit des feuilles de style est conforme à la spécification CSS1 s'il :

- produit des feuilles de style CSS1 valides

Un agent utilisateur qui utilise CSS1 pour le rendu de documents *et* pour la production de feuilles de style CSS1 est conforme à cette spécification s'il satisfait à ces deux conditions.

Un agent utilisateur n'est pas tenu de supporter toutes les fonctions de CSS1 mais est néanmoins considéré conforme s'il en assure les fonctions de base. Les fonctionnalités de base consistent en la totalité de la spécification CSS1 à l'exception de ce qui est explicitement exclus. Dans ce document, les passages marqués "*CSS1 de base :*" précisent les fonctions exclues. On les appelle aussi fonctions avancées de CSS1.

Ce chapitre ne fait que définir la conformité par rapport à CSS1. Les évolutions des feuilles de style pourront requérir le support d'un autre jeu de fonctions de la part des agents utilisateurs pour prétendre à une nouvelle conformité.

Quelques exemples de contraintes liées au média : des ressources limitées (polices, couleur) et une résolution réduite (ainsi les marges peuvent être imprécises). Dans ces cas, l'agent utilisateur prendra les valeurs les plus approchantes à celles de la feuille de style. Également, les différents paradigmes de l'interface utilisateur peuvent avoir des contraintes propres : un navigateur en Réalité Virtuelle peut redimensionner un document selon la perspective de l'utilisateur.

Les agents utilisateurs peuvent offrir des alternatives de présentation au lecteur. Par exemple, l'agent utilisateur peut fournir des options aux lecteurs handicapés visuels ou encore donner la possibilité de désactiver le clignotement.

Noter que CSS1 ne contrôle pas tous les aspects de la mise en forme. Par exemple, l'agent utilisateur est libre d'interpréter l'interlettrage.

Cette spécification recommande mais sans l'imposer qu'un agent utilisateur :

- permette au lecteur de choisir une feuille de style personnelle
- permette que les feuilles de style individuelles puissent être activées ou inactivées.

Les règles de conformité ci−dessus ne décrivent que des fonctionnalités et non l'interface utilisateur.

7.1 Compatibilité ascendante de l'interprétation

Il s'agit ici de la spécification CSS de niveau 1. Une évolution des niveaux de CSS incorporant des fonctions supplémentaires est très probable. Ce chapitre décrit ce qu'un agent utilisateur, simplement conforme à CSS1, doit faire quand celui−ci rencontrera des constructions non valides pour CSS1 dans une feuille de style.

- une déclaration contenant une propriété inconnue est ignorée. Par exemple, pour une telle feuille de style :

  ```
  H1 { color: red; rotation: 70deg }
  ```

  l'agent utilisateur devra réagir comme si elle avait été :

  ```
  H1 { color: red; }
  ```

- une déclaration contenant une valeur illégale *ou une partie de celle−ci étant illégale* est ignorée :

  ```
  IMG { float: left }       /* CSS1 */
  ```

```
IMG { float: left top }   /* "top" n'est pas une valeur de 'float' */
IMG { background: "red" } /* les mots−clés n'ont pas de guillemets en CSS1 */
IMG { border−width: 3 }   /* une valeur de longueur requiert une unité */
```

Dans cet exemple, la première règle est interprétée en CSS1 et les autres ignorées, comme si la feuille de style avait été :

```
IMG { float: left }
IMG { }
IMG { }
IMG { }
```

Un agent utilisateur qui voudrait se conformer à une future spécification peut accepter une ou plusieurs des autre règles suivantes.

- un mot−clé−at [ndt. at−keyword] invalide est ignoré comme tout ce qui le suit jusqu'au prochain point−virgule (;) inclus ou encore la prochaine paire d'accolades ({...}), selon ce qui survient en premier. Par exemple, supposons la feuille de style suivante :

```
@three−dee {
  @background−lighting {
    azimuth: 30deg;
    elevation: 190deg;
  }
  H1 { color: red }
}
H1 {color: blue}
```

Pour CSS1 '@three−dee' n'est pas valide. En conséquence, la règle−at [ndt. at−rule] en entier est ignorée (jusqu'à la troisième accolade droite inclue). L'agent utilisateur CSS1 passe sur cette instruction, ce qui réduit en fait la feuille de style à :

```
H1 {color: blue}
```

En détail :

Quelle qu'en soit la version CSS, une feuille de style est une liste de *déclarations*. Celles−ci sont de deux sortes : les *règles−at [ndt. at−rule]* et les *ensembles de règles [ndt. ruleset]*. Il peut y avoir des blancs (espaces, caractères de tabulation ou de nouvelle ligne) autour des déclarations.

Les feuilles de style sont souvent insérées dans le document HTML. Il peut être nécessaire de les cacher aux agents utilisateurs anciens en les englobant dans des commentaires HTML. Les marques de commentaire HTML "<!−−" et "−−>" peuvent apparaître avant, après ou entre les déclarations. Il peut y avoir des blancs autour des commentaires.

Les règles−at commencent avec un *mot−clé−at*, un identifiant avec un '@' initial (ex. '@import', '@page'). Un identifiant est formé de lettres, chiffres, tirets et caractères échappés (voir plus bas).

Une règle−at consiste en tout ce qui suit jusqu'au premier point−virgule (;) inclus ou au premier bloc (défini plus loin) qui se présente. L'interprétateur CSS1 ignore l'ensemble d'une règle−at qui ne commence pas par le mot−clé−at '@import' et poursuit le traitement après celle−ci. Si '@import' n'apparaît pas au début d'une feuille de style, cette règle−at n'est pas interprétée, c.−à−d. si celle−ci se présente après d'autres règles (même si celles−ci sont ignorées).

Supposons un interpréteur CSS1 face à la feuille de style :

```
@import "subs.css";
H1 { color: blue }
@import "list.css";
```

Le deuxième '@import' n'est pas valide en CSS1. L'interprétation élimine celui−ci, réduisant en fait la feuille de style à :

```
@import "subs.css";
H1 {color: blue}
```

Un *bloc* commence par une accolade gauche ({) et se termine par une accolade correspondante droite (}). Tous caractères peuvent survenir entre celles−ci, cependant les parenthèses (()), crochets ([]) et accolades ({}) doivent toujours être appariés. On peut aussi les inclure dans une construction. De même pour les guillemets simples (') et

doubles (") dont le contenu est considéré comme une *chaîne* (voir l'atomiseur dans l'appendice B pour la définition d'une chaîne). Voici un exemple de bloc, noter que l'accolade droite entre les guillemets ne va pas de pair avec l'accolade ouvrante du bloc et que le deuxième guillemet simple est échappé, il ne correspond donc pas au guillemet ouvrant :

```
{ causta: "}" + ({7} * '\'') }
```

Une règle se compose d'une *chaîne de sélection [ndt. selector−string]* suivie d'un *bloc de déclaration*. La chaîne de sélection consiste en tout ce qui précède la première accolade gauche ({), celle−ci exclue. La règle n'est pas exécutée si sa chaîne de sélection n'est pas valide pour CSS1.

Par exemple, un interpréteur CSS1 avec cette feuille de style :

```
H1 { color: blue }
P[align], UL { color: red; font−size: large }
P EM { font−weight: bold }
```

La deuxième ligne a une chaîne de sélection invalide pour CSS1. Cette instruction est sautée, réduisant la feuille de style à :

```
H1 { color: blue }
P EM { font−weight: bold }
```

Un bloc de déclaration commence par une accolade gauche ({) et finit avec l'accolade correspondante droite (}). Celui−ci contient ou non une liste de *déclarations* séparées par des points−virgules (;).

Une déclaration est formée d'une *propriété*, du caractère deux−points (:) et d'une *valeur*. Il peut y avoir des blancs entre chacun d'eux. Une propriété est un identifiant tel que défini plus haut. La valeur peut contenir n'importe quel caractère, cependant les parenthèses (()), crochets ([]), accolades ({}) et guillemets (' ") doivent aller de pair. Les parenthèses, crochets et accolades peuvent en faire partie. Les caractères entre guillemets sont considérés comme des chaînes.

Pour que l'ajout de futures propriétés et valeurs à celles existantes soit possible, un agent utilisateur doit ignorer une déclaration dont la propriété ou la valeur sont invalides. Les valeurs de chaque propriétés CSS1 sont soumises à des restrictions syntaxiques et sémantiques.

Pour un agent utilisateur CSS1, supposons la feuille de style :

```
H1 { color: red; font−style: 12pt }
P { color: blue;  font−vendor: any;  font−variant: small−caps }
EM EM { font−style: normal }
```

La seconde déclaration de la première règle comporte une valeur '12pt' incorrecte. La deuxième déclaration de la règle suivante indique une propriété 'font−vendor' indéfinie. L'agent utilisateur CSS1 saute ces instructions, réduisant dans les faits la feuille de style à :

```
H1 { color: red; }
P { color: blue;  font−variant: small−caps }
EM EM { font−style: normal }
```

On peut placer des commentaires (voir le chapitre 1.7) partout où les blancs sont autorisés, ils seront considérés comme des blancs. CSS1 définit ces emplacements (comme à l'intérieur des valeurs) qui concernent aussi les commentaires.

Les règles suivantes sont toujours vérifiées :

- aucune feuille de style n'est sensible à la casse, sauf pour les parties qui ne sont pas régies par CSS, c.−à−d. pour CSS1, les noms des familles de police et les URLs. Cela ne s'applique pas non plus aux attributs CLASS et ID qui dépendent de HTML [2].
- en CSS1, les sélecteurs (nom de l'élément, classes et ID) ne peuvent être formés que des caractères allant de A à Z et 0 à 9, des caractères Unicode 161 à 255, du tiret (−) ainsi que des caractères échappés et tous les caractères Unicode sous forme de code numérique (voir suivant). Ils ne peuvent pas commencer par un tiret ou un chiffre.
- une barre oblique inverse suivie par au plus quatre hexadécimaux (0..9A..F) équivaut au caractère Unicode qui revêt ce nombre.
- chaque caractère sauf un hexadécimal peut être échappé avec une barre oblique inverse pour annuler son sens particulier. Exemple : "\"" est une chaîne composée d'un seul guillemet double.

- les deux articles précédents définissent *l'échappement avec une oblique inverse*. L'échappement fait toujours partie d'un identifiant, sauf à l'intérieur d'une chaîne (c.−à−d., "\7B", la représentation hexadécimale du caractère ({), n'est pas une ponctuation alors que "{" l'est, tout comme "\32" est permis au début du nom d'une classe et "2" ne l'est pas).

Note : l'attribut CLASS en HTML admet plus de caractères pour le nom de la classe que pour celui des sélecteurs. En CSS1, ces caractères doivent être échappés ou prendre le modèle d'écriture Unicode : "B&W?" doit s'écrire "B\&W\?" ou "B\26W\3F" et "" ("kouros" en grec) "\3BA\3BF\3C5\3C1\3BF\3C2". Les versions ultérieures de CSS autoriseront sans doute l'introduction directe d'un plus grand nombre de caractères.

La grammaire de CSS1 se trouve à l'appendice B.

8 Références

[1] W3C resource page on web style sheets (http://www.w3.org/Style)

[2] "HTML 4.0 Specification", D. Raggett, A. Le Hors, I. Jacobs, December 1997. Available at http://www.w3.org/TR/REC−html40/.

[3] T Berners−Lee, D Connolly: "Hypertext Markup Language − 2.0", RFC1866, MIT/W3C, November 1995. The specification is also available in hypertext form (http://www.w3.org/MarkUp/html−spec/html−spec_toc.html)

[4] F Yergeau, G Nicol, G Adams, M Dürst: "Internationalization of the Hypertext Markup Language" (ftp://ds.internic.net/rfc/rfc2070.txt).

[5] ISO 8879:1986. Information Processing − Text and Office Systems − Standard Generalized Markup Language (SGML)

[6] ISO/IEC 10179:1996 Information technology −− Processing languages −− Document Style Semantics and Specification Language (DSSSL).

[7] ISO/IEC 9899:1990 Programming languages −− C.

[8] The Unicode Consortium, "The Unicode Standard −− Worldwide Character Encoding −− Version 1.0", Addison−Wesley, Volume 1, 1991, Volume 2, 1992.

[9] "A Standard Default color Space for the Internet", version 1.10, M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta, 5 November 1996.

[10] CIE Publication 15.2−1986, "Colorimetry, Second Edition", ISBN 3−900−734−00−3 (http://www.hike.te.chiba−u.ac.jp/ikeda/CIE/publ/abst/15−2−86.html)

[11] T Berners−Lee, L Masinter, M McCahill: "Uniform Resource Locators (URL)", RFC 1738, CERN, Xerox Corporation, University of Minnesota, December 1994

[12] "PNG (Portable Network Graphics) Specification, Version 1.0 specification" (http://www.w3.org/TR/REC−png−multi.html)

[13] Charles A. Poynton: "Gamma correction on the Macintosh Platform" (ftp://ftp.inforamp.net/pub/users/poynton/doc/Mac/Mac_gamma.pdf)

[14] International Color Consortium: "ICC Profile Format Specification, version 3.2", 1995 (ftp://sgigate.sgi.com/pub/icc/ICC32.pdf)

[15] S C Johnson: "YACC − Yet another compiler compiler", Technical Report, Murray Hill, 1975

[16] "Flex: The Lexical Scanner Generator", Version 2.3.7, ISBN 1882114213

9 Remerciements

Dans la courte existence de HTML, il y a eu plusieurs propositions de feuilles de style, la présente en est l'aboutissement. Les avis de Robert Raisch, Joe English et Pei Wei ont été déterminants.

De nombreuses personnes ont contribué au développement de CSS1. Parmi celles−ci, nous remercions particulièrement : Terry Allen, Murray Altheim, Glenn Adams, Walter Bender, Tim Berners−Lee, Yves Bertot, Scott Bigham, Steve Byrne, Robert Cailliau, James Clark, Daniel Connolly, Donna Converse, Adam Costello, Todd Fahrner, Todd Freter, Roy Fielding, Neil Galarneau, Wayne Gramlich, Phill Hallam−Baker, Philipp Hoschka, Kevin Hughes, Scott Isaacs, Tony Jebson, William Johnston, Gilles Kahn, Philippe Kaplan, Phil Karlton, Evan Kirshenbaum, Yves Lafon, Murray Maloney, Lou Montulli, Colas Nahaboo, Henrik Frystyk Nielsen, David Perrell, William Perry, Scott Preece, Paul Prescod, Liam Quin, Vincent Quint, Jenny Raggett, Thomas Reardon, Cécile Roisin, Michael Seaton, David Seibert, David Siegel, David Singer, Benjamin Sittler, Jon Smirl, Charles Peyton Taylor, Irène Vatton, Daniel Veillard, Mandira Virmani, Greg Watkins, Mike Wexler, Lydja Williams, Brian Wilson, Chris Wilson, Lauren Wood et Stephen Zilles.

Trois personnes méritent des mentions spéciales : Dave Raggett (pour ses encouragements et son travail sur HTML3), Chris Lilley (pour ses contributions incessantes, surtout dans le domaine de la couleur et des polices) et Steven Pemberton (pour ses talents d'organisateur et sa créativité).

Appendices

Appendice A: Exemple de feuille de style pour HTML 2.0

(Cet appendice est ici à titre informatif et non normatif).

Cet exemple de feuille de style a été écrit en accord avec les principes de rendu de la spécification du HTML 2.0 [3]. Quelques styles ont été rajoutés par souci d'exhaustivité. Pour établir la feuille de style par défaut d'un agent utilisateur, il est suggéré de s'en inspirer.

```
BODY {
  margin: 1em;
  font-family: serif;
  line-height: 1.1;
  background: white;
  color: black;
}

H1, H2, H3, H4, H5, H6, P, UL, OL, DIR, MENU, DIV,
DT, DD, ADDRESS, BLOCKQUOTE, PRE, BR, HR, FORM, DL {
  display: block }

B, STRONG, I, EM, CITE, VAR, TT, CODE, KBD, SAMP,
IMG, SPAN { display: inline }

LI { display: list-item }

H1, H2, H3, H4 { margin-top: 1em; margin-bottom: 1em }
H5, H6 { margin-top: 1em }
H1 { text-align: center }
H1, H2, H4, H6 { font-weight: bold }
H3, H5 { font-style: italic }

H1 { font-size: xx-large }
H2 { font-size: x-large }
H3 { font-size: large }

B, STRONG { font-weight: bolder }  /* relatif au parent */
I, CITE, EM, VAR, ADDRESS, BLOCKQUOTE { font-style: italic }
PRE, TT, CODE, KBD, SAMP { font-family: monospace }

PRE { white-space: pre }

ADDRESS { margin-left: 3em }
BLOCKQUOTE { margin-left: 3em; margin-right: 3em }

UL, DIR { list-style: disc }
OL { list-style: decimal }
MENU { margin: 0 }              /* formatage serré */
LI { margin-left: 3em }

DT { margin-bottom: 0 }
DD { margin-top: 0; margin-left: 3em }

HR { border-top: solid }        /* 'border-bottom' est aussi possible */

A:link { color: blue }          /* liens non visités */
A:visited { color: red }        /* liens visités */
A:active { color: lime }        /* liens activés */

/* sélecteurs contextuels requis pour la bordure indiquant un lien autour des éléments IMG */

A:link IMG { border: 2px solid blue }
A:visited IMG { border: 2px solid red }
A:active IMG { border: 2px solid lime }
```

Appendice B: Grammaire de CSS1

(Cette appendice est normatif).

La grammaire minimale CSS (pour toutes les versions de CSS) que doit suivre chacune des implémentations est définie au chapitre 7. La grammaire ci−dessous définit en des termes beaucoup plus réduits le langage qui décrit la syntaxe de CSS1.

C'est en quelque sorte encore un dérivé de CSS1, les contraintes de sémantique venant en plus ne sont pas exprimées dans celle−ci. Un agent utilisateur conforme doit aussi adhérer aux règles de compatibilité ascendante (chapitre 7.1), de notation des propriétés et des valeurs (chapitre 5) et de notation des unités (chapitre 6). En plus des restrictions imposées par HTML, ex. les valeurs possibles de l'attribut CLASS.

La grammaire ci−dessous est LL(1) (noter que la plupart des agents utilisateurs ne devraient pas l'utiliser directement car elle ne suit pas les conventions d'interprétation mais seulement la syntaxe CSS1). Le format des productions facilite un usage humain et quelques raccourcis de notation ne faisant pas partie de yacc [15] sont employés :

```
*  : 0 ou plus
+  : 1 ou plus
?  : 0 ou 1
|  : alternatives
[] : regroupement
```

Les productions sont :

```
stylesheet
 : [CDO|CDC]* [ import [CDO|CDC]* ]* [ ruleset [CDO|CDC]* ]*
 ;
import
 : IMPORT_SYM [STRING|URL] ';'          /* ex., @import url(fun.css); */
 ;
unary_operator
 : '−' | '+'
 ;
operator
 : '/' | ',' | /* vide */
 ;
property
 : IDENT
 ;
ruleset
 : selector [ ',' selector ]*
   '{' declaration [ ';' declaration ]* '}'
 ;
selector
 : simple_selector+ [ pseudo_element | solitary_pseudo_element ]?
 | solitary_pseudo_element
 ;
       /* Un "id" est un ID attaché à un type d'élément
       ** sur sa gauche comme dans : P#p007.
       ** Un "solitary_id" est un ID moins lié
       ** comme dans : #p007
       ** De même pour les classes et pseudo−classes.
       */
simple_selector
 : element_name id? class? pseudo_class?        /* ex. : H1.subject */
 | solitary_id class? pseudo_class?             /* ex. : #xyz33 */
 | solitary_class pseudo_class?                 /* ex. : .author */
 | solitary_pseudo_class                        /* ex. : :link */
 ;
element_name
 : IDENT
 ;
pseudo_class                                    /* comme dans :  A:link */
 : LINK_PSCLASS_AFTER_IDENT
 | VISITED_PSCLASS_AFTER_IDENT
 | ACTIVE_PSCLASS_AFTER_IDENT
 ;
solitary_pseudo_class                           /* comme dans :  :link */
 : LINK_PSCLASS
 | VISITED_PSCLASS
 | ACTIVE_PSCLASS
 ;
```

```
class                                   /* comme dans :  P.note */
 : CLASS_AFTER_IDENT
 ;
solitary_class                          /* comme dans :  .note */
 : CLASS
 ;
pseudo_element                          /* comme dans :  P:first-line */
 : FIRST_LETTER_AFTER_IDENT
 | FIRST_LINE_AFTER_IDENT
 ;
solitary_pseudo_element                 /* comme dans :  :first-line */
 : FIRST_LETTER
 | FIRST_LINE
 ;
        /* Il y a une contrainte sur id et solitary_id telle que
        ** la partie située après "#" soit une valeur ID HTML valide ;
        ** ex., "#x77" est OK, mais pas "#77".
        */
id
 : HASH_AFTER_IDENT
 ;
solitary_id
 : HASH
 ;
declaration
 : property ':' expr prio?
 | /* empty */                          /* Évite les erreurs de syntaxe... */
 ;
prio
 : IMPORTANT_SYM                        /* !important */
 ;
expr
 : term [ operator term ]*
 ;
term
 : unary_operator?
   [ NUMBER | STRING | PERCENTAGE | LENGTH | EMS | EXS
   | IDENT | hexcolor | URL | RGB ]
 ;
        /* Il y a une contrainte sur la couleur telle que celle-ci
        ** ait 3 ou 6 hexadécimaux (c.-à-d., [0-9a-fA-F])
        ** après "#"; ex., "#000" est OK, mais non "#abcd".
        */
hexcolor
 : HASH | HASH_AFTER_IDENT
 ;
```

Suit l'atomiseur, écrit en notation flex [16]. Noter que cela suppose une implémentation 8 bits de flex. L'atomiseur est insensible à la casse (option -i en ligne de commande flex).

```
unicode         \\[0-9a-f]{1,4}
latin1          [¡-ÿ]
escape          {unicode}|\\[ -~¡-ÿ]
stringchar      {escape}|{latin1}|[ !#$%&(-~]
nmstrt          [a-z]|{latin1}|{escape}
nmchar          [-a-z0-9]|{latin1}|{escape}
ident           {nmstrt}{nmchar}*
name            {nmchar}+
d               [0-9]
notnm           [^-a-z0-9\\]|{latin1}
w               [ \t\n]*
num             {d}+|{d}*\.{d}+
string          \"({stringchar}|\')*\"|\'({stringchar}|\")*\'

%x COMMENT
%s AFTER_IDENT

%%
"/*"                            {BEGIN(COMMENT);}
<COMMENT>"*/"                   {BEGIN(0);}
<COMMENT>\n                     {/* ignore */}
<COMMENT>.                      {/* ignore */}
@import                         {BEGIN(0); return IMPORT_SYM;}
"!"{w}important                 {BEGIN(0); return IMPORTANT_SYM;}
{ident}                         {BEGIN(AFTER_IDENT); return IDENT;}
{string}                        {BEGIN(0); return STRING;}

{num}                           {BEGIN(0); return NUMBER;}
```

```
{num}"%"                        {BEGIN(0); return PERCENTAGE;}
{num}pt/{notnm}                 {BEGIN(0); return LENGTH;}
{num}mm/{notnm}                 {BEGIN(0); return LENGTH;}
{num}cm/{notnm}                 {BEGIN(0); return LENGTH;}
{num}pc/{notnm}                 {BEGIN(0); return LENGTH;}
{num}in/{notnm}                 {BEGIN(0); return LENGTH;}
{num}px/{notnm}                 {BEGIN(0); return LENGTH;}
{num}em/{notnm}                 {BEGIN(0); return EMS;}
{num}ex/{notnm}                 {BEGIN(0); return EXS;}

<AFTER_IDENT>":"link            {return LINK_PSCLASS_AFTER_IDENT;}
<AFTER_IDENT>":"visited {return VISITED_PSCLASS_AFTER_IDENT;}
<AFTER_IDENT>":"active  {return ACTIVE_PSCLASS_AFTER_IDENT;}
<AFTER_IDENT>":"first-line      {return FIRST_LINE_AFTER_IDENT;}
<AFTER_IDENT>":"first-letter    {return FIRST_LETTER_AFTER_IDENT;}
<AFTER_IDENT>"#"{name}          {return HASH_AFTER_IDENT;}
<AFTER_IDENT>"."{name}          {return CLASS_AFTER_IDENT;}

":"link                         {BEGIN(AFTER_IDENT); return LINK_PSCLASS;}
":"visited                      {BEGIN(AFTER_IDENT); return VISITED_PSCLASS;}
":"active                       {BEGIN(AFTER_IDENT); return ACTIVE_PSCLASS;}
":"first-line                   {BEGIN(AFTER_IDENT); return FIRST_LINE;}
":"first-letter                 {BEGIN(AFTER_IDENT); return FIRST_LETTER;}
"#"{name}                       {BEGIN(AFTER_IDENT); return HASH;}
"."{name}                       {BEGIN(AFTER_IDENT); return CLASS;}

url\({w}{string}{w}\)                                   |
url\({w}([^ \n\'\")]|\\\ |\\\'|\\\"|\\\))+{w}\)         {BEGIN(0); return URL;}
rgb\({w}{num}%?{w}\,{w}{num}%?{w}\,{w}{num}%?{w}\)      {BEGIN(0); return RGB;}

[-/+{};,#:]                     {BEGIN(0); return *yytext;}
[ \t]+                          {BEGIN(0); /* ignore les blancs */}
\n                              {BEGIN(0); /* ignore les blancs */}
\<\!\-\-                        {BEGIN(0); return CDO;}
\-\-\>                          {BEGIN(0); return CDC;}
.                               {fprintf(stderr, "%d: Illegal character (%d)\n",
                                 lineno, *yytext);}
```

Appendice C: Encodage

(Cet appendice est ici à titre informatif et non normatif).

Les documents HTML peuvent inclure tous les caractères Unicode parmi les quelques 30000 définis. Beaucoup de documents n'en utilisent que quelques centaines. Beaucoup de polices n'ont aussi que quelques centaines de glyphes. Cet appendice explique de pair avec le chapitre 5.2 la correspondance entre les caractères d'un document et les glyphes d'une police.

Encodage des caractères

Le contenu d'un document HTML est une séquence de *caractères* avec un balisage. Pour la transmettre "au travers du fil", elle est encodée dans une séquence d'octets à l'aide d'un ou de plusieurs *encodages* possibles. Le document HTML passe par une étape de décodage pour obtenir le caractère. Par exemple, en Europe occidentale, l'octet 224 est habituellement utilisé pour représenter un a−accent grave (à), mais en hébreu, cet octet représente couramment un aleph. En japonais, la signification d'un octet dépend de celui qui le précède. Dans certains encodages, deux (ou plus) octets sont nécessaires pour représenter un caractère.

L'agent utilisateur s'appuie sur le paramètre de "charset" (jeu de caractères) dans l'en−tête HTTP pour décoder l'octet. "ASCII" (pour l'anglais), "ISO−8859−1" (pour l'Europe occidentale), "ISO−8859−8" (pour l'hébreu), "Shift−JIS" (pour le japonais) sont des encodages (ou valeurs de charset) typiques.

HTML [2][4] admet quelques 30000 caractères, nommément ceux définis par Unicode. Peu de documents utilisent autant de caractères différents et le choix du bon encodage est généralement suffisant pour indiquer que le document ne nécessite qu'un octet par caractère. Certains caractères hors du champs d'encodage peuvent encore apparaître sous une forme numérique référençant le caractère : 'Π' représentera toujours la lettre Pi majuscule, indépendamment de l'encodage utilisé. Cela sous−entend que les agents utilisateurs doivent reconnaître chacun des caractères Unicode, même s'ils ne peuvent honorer que peu d'encodages.

Encodage de police

Une police ne contient pas de *caractères* mais des images de ceux−ci qu'on nomme *glyphes*. Sous la forme de contours ou de points, les glyphes constituent une représentation particulière du caractère. De façon implicite ou explicite, chaque police a une table, la *table d'encodage de la police*, qui lui est associée et qui indique pour quel caractère chaque glyphe est une représentation. Pour les polices de Type 1, on parle de la table comme du *vecteur d'encodage*.

En fait, de nombreuses polices ont plusieurs glyphes pour le même caractère. Le glyphe qui est utilisé dépend soit des règles d'usage de la langue ou soit des préférences de l'auteur.

En arabe, par exemple, chaque lettre dispose de quatre formes différentes, selon qu'elle est utilisée au début, au milieu, à la fin d'un mot ou bien isolément. Il s'agit du même caractère dans tous les cas, c'est pourquoi un seul caractère apparaît dans le document HTML, mais il semble à chaque fois différent à l'impression.

Il existe aussi des polices dont le graphiste est libre de choisir une forme parmi des alternatives variées. Malheureusement, CSS1 n'offre pas de moyens pour la sélection de celles−ci. Seule la forme standard de ces polices est retenue pour le moment.

Jeux de polices

CSS1 autorise l'usage de *jeux de polices* pour obtenir l'affichage de tous les caractères dans un document ou même dans un seul élément, pour lequel parfois, une seule police ne suffit pas.

En CSS1, un jeu de police est une liste de polices, toutes de même style et taille et qui sont essayées l'une à la suite de l'autre pour y trouver un glyphe correspondant à un caractère donné. Un élément contenant du texte et des symboles mathématiques peut avoir besoin d'un jeu de deux polices, l'une avec des chiffres et des lettres, l'autre avec des symboles mathématiques. Voir au chapitre 5.2 pour une explication en détail du mécanisme de sélection des jeux de polices.

Voici un exemple de jeu de polices pour un texte sensé contenir des caractères latins, japonais et des symboles mathématiques :

```
BODY { font−family: Baskerville, Mincho, Symbol, serif }
```

Les caractères latins disponibles dans la police Baskerville seront tirés de celle−ci, les japonais de Mincho et les symboles de Symbol. Tout autre caractère sera issu (heureusement) de la famille de police générique 'serif'. Cette famille de police 'serif est utilisée si une ou plus des autres polices ne sont pas disponibles.

Appendice D: Correction du gamma

(Cet appendice est ici à titre informatif et non normatif).

Voir le tutoriel sur le gamma dans la spécification de PNG [12] pour ceux qui ne sont pas familiers avec ces questions.

Lors des calculs pour l'affichage sur un moniteur [ndt. CRT], l'agent utilisateur peut considérer un moniteur idéal et ignorer les effets du gamma apparent dus au dithering. Cela veut dire que le calcul minimum requis pour les plates−formes actuelles est :

PC avec MS−Windows
: aucun

Unix avec X11
: aucun

Mac avec QuickDraw
: appliquer un gamma 1.39 [13] (Les applications travaillant avec ColorSync peuvent simplement passer le profil ICC sRGB [14] à celui−ci pour effectuer la correction de couleur)

SGI avec X
: appliquer la valeur de gamma avec `/etc/config/system.glGammaVal` (la valeur par défaut étant 1.70, les applications tournant sur Irix, 6.2 ou plus, peuvent simplement passer le profil ICC sRGB au système de gestion de couleur)

NeXT avec NeXTStep
: appliquer un gamma 2.22

"Appliquer le gamma" signifie la conversion de chacune des valeurs R, G et B en $R'=R^{gamma}$, $G'=G^{gamma}$, $B'=B^{gamma}$ avant de les passer au système.

Cela peut être fait rapidement en construisant une table de lookup de 256 éléments au lancement du navigateur ainsi :

```
for i := 0 to 255 do
  raw := i / 255;
  corr := pow (raw, gamma);
  table[i] := trunc (0.5 + corr * 255.0)
end
```

ce qui évite des calculs astronomiques sur chaque attribut de couleur, et à plus forte raison sur chaque pixel.

Appendice E: Application et évolution de CSS1

(Cet appendice est ici à titre informatif et non normatif).

Le travail effectué pour CSS1 a eu pour but la création d'un mécanisme simple de feuille de style pour les documents HTML. La spécification actuelle représente un équilibre entre la simplicité voulue pour réaliser des feuilles de style pour le web et la pression des auteurs pour un contrôle visuel plus riche. CSS1 offre :

- le remplacement du balisage visuel : les feuilles de style CSS1 remplace facilement les extensions HTML, ex. "CENTER", "FONT" et "SPACER".
- un balisage plus élégant : au lieu d'utiliser l'élément "FONT" pour obtenir des petites capitales, une déclaration dans la feuille de style suffit. Comparons le balisage visuel :

  ```
  <H1>G<FONT SIZE=−1>ROS TITRE</FONT></H1>
  ```

  avec la feuille de style :

  ```
  H1 { font−style: small−caps }

  <H1>Gros titre</H1>
  ```

- divers niveaux d'intégration : les règles de style CSS1 peuvent provenir d'une feuille de style externe, être inclues dans l'élément 'STYLE' ou mises dans l'attribut 'STYLE'. Cette dernière option pour faciliter la transition à partir des extensions HTML vers CSS.
- des nouveaux effets : quelques effets visuels ont été ajoutés pour la joie des utilisateurs : les pseudo−éléments typographiques et les autres valeurs de la propriété 'background' en font partie.
- l'adaptabilité : CSS1 agit utilement sur une gamme d'appareils allant du terminal en mode texte à la station de travail en couleur haute résolution. Les auteurs peuvent écrire une feuille de style dont on peut raisonnablement penser qu'elle s'appliquera au mieux de la façon voulue.

CSS1 n'offre pas :

- un contrôle au pixel : CSS1 favorise la simplicité sur le niveau de contrôle. Bien que l'utilisation conjointe des images de fond et des styles soit forte de possibilités, le contrôle au pixel près n'est pas possible.
- un contrôle par l'auteur : l'auteur ne peut pas imposer l'utilisation de telle feuille de style, seulement la suggérer.
- un langage de mise en page : CSS1 n'offre pas le multi−colonnage avec le transvasement automatique du texte, des cadres qui se superposent, etc.
- un langage de recherche riche dans l'arborescence du document : CSS1 peut seulement remonter aux éléments parents dans l'arborescence, même si d'autres langages de feuille de style (ex. DSSSL [6]) sont dotés de capacités de recherche complètes.

CSS va s'étendre dans plusieurs directions :

- le papier : une meilleure prise en compte de l'impression des documents HTML
- le support des médias non visuels : l'ajout de nouvelles propriétés et leurs valeurs sont étudiées pour le support des sorties vocales et braille
- les noms de couleur : la liste actuelle va peut être s'allonger
- les polices : on attend des systèmes de spécification des polices plus précis pour compléter les propriétés de police existantes.
- les valeurs et propriétés : on attend que les éditeurs proposent des améliorations au jeu des propriétés et valeurs de CSS1. La spécification peut être facilement étendue, mais l'interopérabilité entre différents agents utilisateurs est une autre histoire
- un langage de mise en page : prise en compte de la mise en page en deux dimensions dans la tradition des applications de publication numérique.
- d'autres DTDs : CSS1 comporte des parties spécifiquement HTML (ex. le statut particulier des attributs 'CLASS' et 'ID') qui aussi pourraient être facilement portés sur d'autres DTDs.

Ce que CSS ne risque pas de devenir :

- un langage de programmation.

Appendice F: Changements depuis la version du 17 décembre 1996

[N.d.T] : L'appendice F présente un intérêt pour le moins réduit à la compréhension de CSS1, c'est pourquoi celui−ci n'est pas traduit.

(Cet appendice est ici à titre informatif et non normatif).

Ce document est une révision de la Recommandation CSS1 publiée initialement le 17 décembre 1996 et la liste ci−dessous mentionne les changements apportés. En choisissant la feuille de style alterne "errata", les changements apparaissent surlignés.

Nos remerciements à Komachi Yushi, Steve Byrne, Liam Quinn, Kazuteru Okahashi, Susan Lesch et Tantek Çelik pour leur aide dans cette version révisée.

Erreurs orthographiques et typographiques

- typo1 [section 1.1] The sentence:

  > As as result, old UAs will ignore the 'STYLE' element, but its content will be treated as part of the document body, and rendered as such.

  has been changed to:

  > As a result, old UAs will ignore the 'STYLE' element, but its content will be treated as part of the document body, and rendered as such.

- typo2 [section 1.6] The sentence:

  > The second selector matches matches all 'H1' elements that have an ancestor of class 'reddish'.

  has been changed to:

  > The second selector matches all 'H1' elements that have an ancestor of class 'reddish'.

- typo3 [section 2.1] The sentence:

  > E.g., a style sheet can legally specify that the 'font−size' of an 'active' link should be larger that a 'visited' link, but the UA is not required to dynamically reformat the document when the reader selects the 'visited' link.

  has been changed to:

  > E.g., a style sheet can legally specify that the 'font−size' of an 'active' link should be larger than a 'visited' link, but the UA is not required to dynamically reformat the document when the reader selects the 'visited' link.

- typo4, typo5 [section 2.3−2.4] A trailing quote mark has been added to 'vertical−align'.
- typo6 [section 4] A missing right parenthesis has been added.
- typo7 [section 4.1] A missing comma has been added.
- typo8 [section 4.1.2] The text:

  > "If more than one of the three is 'auto', and one of them is 'width', than the others..."

  has been changed to:

  > "If more than one of the three is 'auto', and one of them is 'width', then the others..."

- typo9 [section 5.3.6] The word "Examples" has been capitalized.
- typo10 [section 5.4.4] An entity gone astray has been corrected (from "<length&t;" to "<length>").
- typo11 [section 5.5] The text:

  > The margin properties properties set the margin of an element.

has been changed to:

> The margin properties set the margin of an element.

- typo12 [section 5.5.25] Superfluous quote marks have been removed.

Erreurs

- error1 [section 2.3] A previously invalid declaration (`font−style: small−caps`) has been replaced by a valid one.
- error2 [section 4] This sentence:

> CSS1 assumes a simple box−oriented formatting model where each element results in one or more rectangular boxes.

has been replaced with:

> CSS1 assumes a simple box−oriented formatting model where each formatted element results in one or more rectangular boxes.

- error3 [section 4.1] In this sentence:

> The top is the top of the object including any padding, border and margin; it is only defined for inline and floating elements, not for non−floating block−level elements.

the word "object" has been replaced with "element" to use consistent terminology.

- error4 [section 4.1.3] The alignment of the list items has been corrected.
- error5 [section 4.1.4] There is only one P element in the example, and this sentence:

> Note that the margin of the 'P' elements enclose the floating 'IMG' element.

has therefore been corrected to:

> Note that the margin of the 'P' element enclose the floating 'IMG' element.

- error6, error7 [section 4.5] The window size can only influence one axis of the canvas, either the width or the height.
- error8 [section 5.4.1] The text inside the parenthesis now refers to a CSS1 property.
- error9 [section 5.4.1] To correspond with the following paragraph, the example has been corrected.
- error10 [section 5.4.8] The section "The height of lines" is now correctly identified as section 4.4, not 4.7.
- error11 [section 5.5] This sentence:

> The 'margin' property sets the border for all four sides while the other margin properties only set their respective side.

has been corrected to:

> The 'margin' property sets the margin for all four sides while the other margin properties only set their respective side.

- error12, error13, error14, error15, error16, error17, error18, error19, error20, error22 [section 5.5.1−5.5.10] Percentage values refer to width of the closest block−level ancestor, not the parent element (which can be inline) as previously stated.
- error21 [section 5.5.10] Shorthand properties don't have initial values, and the previously specified '0' has therefore been corrected.
- error23 [section 5.5.15] The last rule in the example previously contained an illegal value ('none').
- error24 [section 5.6.6] The value specification on the 'list−style' property had been corrected, but there are no syntactic or semantic changes.
- error25 [section 7.1] The numeric character references used to encode the Greek word "kouros" have been corrected.
- error26 [Appendix A] These element types have been added to the list of selectors attached to the 'display: block' declaration: FORM DL.

- error27 [Appendix B] A dead link to section 7 has been fixed.
- error28 [Appendix D] This text:

> "Applying gamma" means that each of the three R, G and B must be converted to $R'=R^{gamma}$, $G'=G^{gamma}$, $G'=B^{gamma}$, before handing to the OS.

has been changed to:

> "Applying gamma" means that each of the three R, G and B must be converted to $R'=R^{gamma}$, $G'=G^{gamma}$, $B'=B^{gamma}$, before handing to the OS.

Structure et Organisation

- Appendix F, which lists all changes since the 17 December 1996 version has been added.
- A paragraph in the status section has been added to inform readers that this is a revised version.
- The style sheet has changed.
- The reference to a future HTML specification with support for style sheets has been updated to reference HTML 4.0.
- URLs in the References have been updated, and dead links have been removed.
- The underlying HTML markup has been revised.

Notes