まえがき

この規格は，工業標準化法第 12 条第 1 項の規定に基づき，財団法人日本規格協会(**JSA**)から，工業標準原案を具して日本工業規格を制定すべきとの申出があり，日本工業標準調査会の審議を経て，経済産業大臣が制定した日本工業規格である。

JIS X 4168 には，次に示す附属書がある。

附属書 A（参考）HTML2.0 用のスタイルシート例

附属書 B（規定）CSS1 文法

附属書 C（規定）符号化

附属書 D（参考）ガンマ補正

附属書 E（参考）CSS1 の適用可能性及び拡張可能性

原勧告の標題及びまえがきの翻訳
段階スタイルシート 水準 1(CSS1)

W3C 勧告 1996 年 12 月 17 日，改訂 1999 年 1 月 11 日

この規格の原規定は，W3C Recommendation 17 Dec 1996 の改訂版の W3C Recommendation 11 Jun 1999 であって，次のウェブ上で公開されている。

http://www.w3.org/pub/WWW/TR/REC-CSS1

原規定は，次の 2 名によって編集された。

Hakon Wium Lie (howcome@w3.org)

Bert Bos (bert@w3.org)

この規格の原規定は，W3C 勧告である。それは W3C (http://www.w3.org/)メンバによってレビューされ，その規定が利用に適するとの一般的合意が得られた。それは安定した文書であって，参考資料として利用したり，他の文書から引用規格として引用してもよい。W3C は，この勧告の広範囲な展開を推進する。

現在の W3C の勧告及びその他の技術文書のリストは，http://www.w3.org/pub/WWW/TR/で見ることができる。

この規格は，W3C による 1996 年 12 月 17 日の最初の公表（TR X 0011:1998 に一致）の改訂版である。この改訂版の既に明らかになっている誤り（JIS X 4168 に反映済み）は，http://www.w3.org/Style/CSS/Errata/REC-CSS1-19990111-errata から入手できる。

目　次

日本工業規格（案）　　JIS X 4168：2004

段階スタイルシート 水準 1(CSS1)

Cascading Style Sheets, level 1

序文　この規格は，1999 年 1 月に World Wide Web Consortium(W3C)から公表された勧告 Cascading Style Sheets, level 1 及びそれに対する訂正票[Errata for the CSS1 specification (revision of 11 Jan 1999)]を翻訳し，技術的内容を変更することなく作成した日本工業規格である。

0. 一般

0.1　適用範囲　この規格は，段階スタイルシート(CSS)の水準 1（以降，CSS1)の機構を規定する。CSS1 は，簡単なスタイルシート機構であって，文書作成者及び読者が，フォント，カラー，スペースなどのスタイルを HTML 文書に付与することを可能にする。CSS1 言語は，人が読み書きでき，通常のデスクトップ出版用語によってスタイルを表記する。

CSS の基本的な特徴の一つは，スタイルシートが段階的であることにある。つまり，文書作成者は希望するスタイルシートを付与できるが，読者は，人的又は技術的ハンディキャップを調整するために個人的なスタイルシートをもってもよい。異なるスタイルシート間の競合を解決するための規則を，この規格が規定する。

この規格の原規定は，スタイルシート分野の W3C の活動によって作成された。スタイルシートの背景情報については，**8.**の[1]を参照されたい。

0.2　定義

0.2.1　属性 (attribute)　HTML の属性を示す。

0.2.2　文書作成者 (author)　HTML 文書を作成する人。

0.2.3　ブロックレベル要素 (block-level element)　前後に改行をもつ要素。例えば，HTML における'H1'。

0.2.4　キャンバス (canvas)　UA(利用者エージェント)が描画する面のその部分であって，そこに文書を可視化する。

0.2.5　子要素 (child element)　SGML（**8.**の[5]を参照）用語における下位要素。

0.2.6　文脈依存選択子 (contextual selector)　文書構造の中での位置に基づいて要素との一致をとる選択子。文脈依存選択子は，幾つもの単純選択子から成る。例えば，文脈依存選択子'H1.initial B'は，二つの単純選択子'H1.initial'及び'B'から成る。

0.2.7　CSS　段階スタイルシート(Cascading Style Sheets)の短縮形。

0.2.8　CSS1　段階スタイルシート水準 1(Cascading Style Sheets, level 1)の短縮形。この規格は，ウェブ用の簡単なスタイルシート機構である CSS1 を規定する。

0.2.9　CSS1 上位機能 (CSS1 advanced features)　この規格には規定されるが，CSS1 コア機能には分類されない機能。

0.2.10 CSS1 コア機能 (CSS1 core features) すべての CSS1 適合の UA（利用者エージェント）に必要とされる CSS1 の部分。

0.2.11 CSS1 パーサ (CSS1 parser) CSS1 スタイルシートを読む UA（利用者エージェント）。

0.2.12 宣言 (declaration) 特性（例えば'font-size'）及び対応する値（例えば'12pt'）。

0.2.13 設計者 (designer) スタイルシートを設計する者。

0.2.14 文書 (document) HTML 文書を示す。

0.2.15 要素 (element) HTML の要素を示す。

0.2.16 要素型 (element type) SGML（**8.**の[5]を参照）用語における共通識別子(generic identifier)。

0.2.17 仮想タグ列 (fictional tag sequence) 疑似クラス及び疑似要素の振る舞いを記述するツール。

0.2.18 フォントサイズ (font size) フォントの設計寸法。通常は，フォントのサイズは，ディセンダをもつ最も高さの低い字の下部から，アセンダをもち（オプションとして）ダイアクリティカルマークをもつ最も高さの高い字の上部までの距離に近似的に等しい。

0.2.19 HTML ハイパテキストマーク付け言語(Hypertext Markup Language)（**8.**の[2]を参照）の短縮形であって，SGML の応用。

0.2.20 HTML 拡張 (HTML extension) 多くの場合ある可視的効果をサポートするために，UA ベンダが導入するマーク付け。HTML 拡張の例として，"BGCOLOR"属性と同様に，"FONT"要素，"CENTER"要素及び"BLINK"要素がある。CSS の目的の一つは，別の HTML 拡張を提供することにある。

0.2.21 行内要素 (inline element) 前後に改行をもたない要素。例えば，HTML における'STRONG'。

0.2.22 固有寸法 (intrinsic dimensions) 周囲に無関係に要素そのものによって定義される幅及び高さ。この規格は，すべての置換要素が（置換要素だけが）固有寸法となることを前提とする。

0.2.23 親要素 (parent element) SGML（**8.**の[5]を参照）用語における親要素(containing element)。

0.2.24 疑似要素 (pseudo-element) 構造的な要素ではなく表示上の項目（例えば，要素の最初の行）を指定するために，CSS 選択子の中で用いる記述単位。

0.2.25 疑似クラス (pseudo-class) HTML ソースに対する外部情報（例えば，アンカをたどったか否かという事実）で要素を分類することを可能にするために，CSS 選択子の中で用いる記述単位。

0.2.26 特性 (property) CSS によって影響を与えることができるスタイル上のパラメタ。この規格は，特性及び対応する値のリストを定義する。

0.2.27 読者 (reader) 文書可視化の対象となる人。

0.2.28 置換要素 (replaced element) CSS フォーマタだけが固有寸法を知る要素。HTML においては, 'IMG', 'INPUT', 'TEXTAREA', 'SELECT'及び'OBJECT'の各要素が，置換要素の例になり得る。例えば，'IMG'要素の内容は，SRC 属性が指定する画像によって置換されることが多い。CSS1 は，固有寸法を知る方法については規定しない。

0.2.29 規則 (rule) 宣言（例えば 'font-family: helvetica'）及びその選択子（例えば'H1'）。

0.2.30 選択子 (selector) 対応する規則をどの要素に適用するかを特定する文字列。選択子は，単純選択子（例えば 'H1'）であるか，複数の単純選択子から成る文脈依存選択子（例えば'H1 B'）であるかのどちらかとする。

0.2.31 SGML 標準一般化マーク付け言語(Standard Generalized Markup Language（**8.**の[5]を参照)の短縮形のことであって，HTML はその応用。

0.2.32 単純選択子 (simple selector) 要素型及び/又は属性に基づいて要素との一致をとる選択子であって，文書構造における要素の位置に基づくものではない。単純選択子の 1 例に，'H1.initial'がある。

0.2.33　スタイルシート (style sheet)　規則の集まり。

0.2.34　UA　利用者エージェント(User Agent)の短縮形であって，ウェブブラウザ又はウェブクライアントであることが多い。

0.2.35　利用者 (user)　文書可視化の対象となる人。読者(**0.2.27**)と同義。

0.2.36　重み (weight)　規則の優先度。

この規格においては，一重引用符('...')は，HTML 及び CSS の抜粋を引用する。

1.　基本概念　簡単なスタイルシートを設計することは容易であり，HTML を少し知っていて，幾つかの基本的なデスクトップ出版用語を知っているだけでよい。例えば，'H1'要素のテキストカラーを青にするには，次のとおり記述すればよい。

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

この例は，簡単な CSS 規則であって，規則は二つの主要部分，つまり選択子('H1')及び宣言('color: blue')から成る。宣言は二つの部分，つまり特性('color')及び値('blue')をもつ。この例は，HTML 文書を可視化するために必要な特性の一つだけに影響を与えようとするものであるが，それだけで一つのスタイルシートとして特徴付けられている。他のスタイルシートと組み合わせれば（CSS の基本的な特徴の一つが，スタイルシートを組み合わせること。），文書の最終表示を決定する。

選択子は，HTML 文書とスタイルシートとの間のリンクであり，すべての HTML 要素型は，選択子になり得る。HTML 要素型は，HTML 規定（**8.**の[2]を参照）の中で定義される。

'color'特性は，HTML 文書の表示を決定する約 50 個の特性の一つとする。この規格は，特性及びそれがとる値のリストを定義する。

HTML の文書作成者は，その文書に固有のスタイルを推奨しようとする場合だけ，スタイルシートを書く必要がある。各 UA（"ウェブブラウザ"又は"ウェブクライアント"であることが多い。）は，適当ではあるがありきたりの方法で文書を表示するデフォルトのスタイルシートをもっている。**附属書 A**は，HTML 2.0 規定（**8.**の[3]を参照）が推奨するとおりに HTML 文書を表示するスタイルシートの例を示す。

CSS1 言語の形式文法については，**附属書 B**が規定する。

1.1　HTML への組込み　スタイルシートが表示に影響を与えるためには，UA は，スタイルシートの存在を知らなければならない。HTML 規定（**8.**の[2]を参照）が，どのようにして HTML をスタイルシートとをリンクさせるかを規定する。したがって **1.1** は，参考であって規定ではない。

```
<HTML>
  <HEAD>
    <TITLE>title</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>Headline is blue</H1>
```

```
    <P STYLE="color: green">While the paragraph is green.
  </BODY>
</HTML>
```

この例は，スタイルと HTML とを組み合わせる次の四つの方法を示す。

a) 外部スタイルシートをリンクする'LINK'要素を用いる。

b) 'HEAD'要素の中の'STYLE'要素を用いる。

c) CSS の'@import'記法を使用してインポートしたスタイルシートを用いる。

d) 'BODY'の中の要素の'STYLE'属性を用いる。

オプション d)は，スタイルを内容と混在させ，これまでのスタイルシートにあった利点を損ねる。

'LINK'要素は，読者が選択できる別のスタイルシートを参照し，インポートしたスタイルシートは，スタイルシートの残りと自動的にマージされる。

従来より，UA は，未知のタグに対しては何もせずに無視してきた。その結果，古い UA は，'STYLE'要素を無視するが, その内容は文書本体の部分として扱われ, そのまま可視化される。切り替えの際には，'STYLE'要素を，SGML コメントを用いて隠してもよい。

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

'STYLE'要素は，（**8.**の[2]で定義するとおり）DTD の中で"CDATA"として宣言されるので，適合 SGML パーサは，このスタイルシートを削除されるはずのコメントであるとは判断しない。

1.2 グループ化 スタイルシートのサイズを小さくするために，選択子をコンマ区切りのリストでグループ化できる。

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

同様に，宣言もグループ化できる。

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

固有のグループ化構文をもつ特性もある。

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

この例は，その前の例に等価となる。

1.3 継承 最初の例（**1.**の例）において，'H1'要素のカラーは青に設定された。'H1'要素の中に強調要素があるとしよう。

```
<H1>The headline <EM>is</EM> important!</H1>
```

'EM'要素にカラーが指定されていなければ，強調された"is"は，親要素のカラーを継承する。つまり，それも青く表示される。'font-family', 'font-size'などの他のスタイル特性も同様に継承される。

"デフォルト"のスタイル特性を文書に設定するために，すべての可視要素を下位にもつ要素に対して特

性を設定できる。HTML 文書では，'BODY'要素がこの機能を果たす。

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

たとえ文書作成者が'BODY'タグを省略しても（文法には違反しない），これは動作する。それは，HTML パーサが欠けたタグを推定することによる。この例は，テキストカラーを黒に，背景をある画像に設定する。もしその画像がなければ，背景は白になる。（これに関する詳細は，**5.3** を参照されたい。）

親要素から子要素に継承されないスタイル特性もある。ほとんどの場合，それは直感に基づく。例えば，'background'特性は継承しないが，親要素の背景はデフォルトで透けて見える。

特性の値が，他の特性に対するパーセントの値になることも多い。

```
P { font-size: 10pt }
P { line-height: 120% }   /* relative to 'font-size', i.e. 12pt */
```

パーセントの値を使える各特性に関しては，それがどの特性を参照するかを定義する。'P'の子要素は，percentageではなく，'line-height'の計算された値（つまり 12pt）を継承する。

1.4　選択子としてのクラス　要素に対する制御の粒度を高めるために，新しい属性'CLASS'が HTML に追加された（**8.**の[2]を参照）。'BODY'要素の中のすべての要素はクラスに分類でき，スタイルシートの中でそのクラスを指定できる。

```
<HTML>
 <HEAD>
  <TITLE>Title</TITLE>
  <STYLE TYPE="text/css">
    H1.pastoral { color: #00FF00 }
  </STYLE>
 </HEAD>
 <BODY>
  <H1 CLASS=pastoral>Way too green</H1>
 </BODY>
</HTML>
```

通常の継承規則は，クラス分けされた要素に適用する。それらは，文書構造の親要素から値を継承する。

選択子の中でタグ名を省略することによって，同一クラスのすべての要素を指定できる。

```
.pastoral { color: green }   /* all elements with CLASS pastoral */
```

選択子ごとに一つだけのクラスを指定できる。したがって 'P.pastoral.marine'は，CSS1 では無効な選択子となる。（**1.6** に示す文脈依存選択子は，単純選択子ごとに一つのクラスをもつことができる。）

CSS は，CLASS 属性に大きな力を与えているので，多くの場合，どの HTML 要素にそのクラスが設定されているかはまったく問題でない。どの要素に対してもそれ以外の要素をエミュレートさせることができる。この力の引継ぎは推奨しない。それは，普遍的な意味をもつ構造（HTML 要素）の水準をそれが削除することによる。CLASS に基づく構造は，制限付きドメインの中だけで有効であり，そこではクラスの意味は相互に承認されてきた。

1.5　選択子としての ID　HTML （**8.**の[2]を参照）は，文書に関して一意な値をもつことが保証されている'ID'属性も導入している。したがってそれは，スタイルシートの選択子として特に重要であり得る。それは，'#'を前置して指定できる。

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

<P ID=z98y>Wide text</P>
```

この例では，最初の選択子は，'ID'属性の値のために，'P'要素と一致する。2 番目の選択子は，要素型('H1')及び ID 値の両方を指定し，その結果，'P'要素とは一致しない。

ID 属性を選択子として用いることによって，スタイル特性を要素ごとに設定できる。スタイルシートは文書構造を活用して設計されてきたが，この機能は，HTML の構造要素の利点を用いることなく，キャンバスにうまく表れる文書を，文書作成者が生成することを可能にする。スタイルシートのこの利用は，推奨しない。

1.6　文脈依存選択子　継承は，CSS 設計者のタイプ入力を削減する。すべてのスタイル特性を設定する代わりに，デフォルトを生成し，例外をリストできる。'H1'の中の'EM'要素に別のカラーを与えるために，次の指定を行える。

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

このスタイルシートが有効である場合，'H1'の内外のすべての強調節は赤になる。'H1'の中の'EM'要素だけを赤にしたければ，次の指定が可能になる。

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

選択子は開いている要素のスタック上での検査パターンであり，この形式の選択子は*文脈依存選択子*と呼ばれる。文脈依存選択子は，空白によって分離された複数の単純選択子（これまでに記述した選択子は，すべて単純選択子であった。）から成る。最後の単純選択子に一致する要素（この場合，'EM'要素）だけが，検査パターンが一致する場合に限って，指定される。CSS1 の文脈依存選択子は，先祖の関係を捜すが，他の関係（例えば，親子関係）は，今後の改訂で導入するかもしれない。この例では，'EM'が'H1'の子孫である場合，つまり'EM'が'H1'要素の内部にある場合に，検査パターンが一致する。

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

ここでは，最初の選択子は，少なくても一つの先祖'UL'をもつ'LI'要素に一致する。2 番目の選択子は，最初の選択子のサブセット，つまり少なくても二つの先祖'UL'をもつ'LI'要素に一致する。その競合は，2 番目の選択子が，もっと長い検査パターンをもつので固有性がより高いことによって解決される。これに関する詳細は，段階順序(**3.2**)を参照されたい。

文脈依存選択子は，要素型，CLASS 属性，ID 属性又はこれらの組合せを探すことができる。

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

最初の選択子は，先祖の中に'DIV'をもつすべての'P'要素と一致する。2 番目の選択子は，クラス'reddish'である先祖をもつすべての'H1'要素と一致する。3 番目の選択子は，'ID=x78y'をもつ要素の子孫であるすべ

ての'CODE'要素と一致する。4 番目の選択子は，クラス'sidenote'である先祖'DIV'をもつすべての'H1'要素と一致する。

幾つかの文脈依存選択子は，グループにまとめることができる。

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

これは，次に等価となる。

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

1.7　コメント　CSS スタイルシートにおけるテキストのコメントは，C プログラム言語（**8.**の[7]を参照）のコメントに類似している。

```
EM { color: red }   /* red, really red!! */
```

コメントは，入れ子にできない。CSS1 パーサにとっては，コメントは空白に等価とする。

2.　疑似クラス及び疑似要素　CSS1 においては，スタイルは通常，文書構造の中の要素位置に基づいて要素に付与される。多様なスタイルにもこの簡単なモデルで充分であるが，幾つかのよく使う組み効果はカバーできない。疑似クラス及び疑似要素の概念は，CSS1 での指定を拡張して，外部情報がフォーマット化処理に影響を与えることを可能にする。

疑似クラス及び疑似要素は，CSS 選択子で用いることができ，HTML ソースには存在しない。スタイルシートにおける指定に用いるために，幾つかの条件のもとで，UA が疑似クラス及び疑似要素を"挿入"する。それらは"クラス"及び"要素"とよばれるが，それは，その振る舞いを記述する便利な方法であることによる。もっと細かくは，*仮想タグ列*がその振る舞いを定義する。

疑似要素は，要素の一部を指定するのに用い，疑似クラスは，スタイルシートをいろいろな要素型の中で区別可能にする。

2.1　アンカ疑似クラス　UA は通常，新しくたどられたアンカを古いものと区別して表示する。CSS1 では，これを'A'要素の中の疑似クラスが扱う。

```
A:link { color: red }          /* unvisited link */
A:visited { color: blue }      /* visited links */
A:active { color: lime }       /* active links */
```

'HREF'属性をもつすべての'A'要素は，これらのグループの一つ（一つだけ）に入る（つまり，目標アンカは影響を受けない。）。UA は，要素を'visited（たどり済み）'から'link（リンク）'に，ある時間の後に移ることを選択してよい。'active（活性）'リンクは，現在読者が（例えば，マウスボタン押下によって）選択中であるものとする。

アンカ疑似クラスのフォーマット化は，クラスが手動で挿入された場合に類似している。UA は，アンカ疑似クラスの遷移によって現在表示している文書を，再フォーマット化する必要はない。例えば，スタイルシートは，文法に違反せずに，'active'リンクの'font-size'は'visited'リンクのそれより大きくすることを指定できるが，読者が'visited'リンクを選択するとき，UA が動的に文書を再フォーマット化する必要はない。

疑似クラス選択子は，通常のクラスと一致せず，逆も成り立つ。したがって，次の例のスタイル規則はどんな影響も与えない。

A:link { color: red }

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

CSS1 では，アンカ疑似クラスは，'A'以外の要素には影響を与えない。したがって，要素型を選択子から省略できる。

A:link { color: red }

:link { color: red }

この二つの選択子は，CSS1 の中の同一要素を選択する。

疑似クラスの名前では，大文字・小文字を区別しない。

疑似クラスは，文脈依存選択子の中で使用できる。

A:link IMG { border: solid blue }

疑似クラスは，通常のクラスと組み合わせることができる。

A.external:visited { color: blue }

<A CLASS=external HREF="http://out.side/">external link</A>

この例におけるリンクがたどられると，それは青に可視化される。通常クラスの名前は，選択子の中で疑似クラスの前に置くことに注意されたい。

2.2 表示上の擬似要素 幾つかの表示上のよく使う組み効果は，構造要素とは関連しないが，キャンバスにフォーマット化される表示上の項目と関連する。CSS1 においては，これらの二つの表示上の項目は，疑似要素，つまり要素の最初の行及び最初の字，によって指定できる。

CSS1 コア: UA は，選択子の中の':first-line'又は':first-letter'をもつすべての規則を無視してよい。そうでなければ，これらの疑似要素の特性のサブセットをサポートするだけとする（**7.**を参照）。

2.3 'first-line'擬似要素 'first-line'擬似要素は，キャンバスにフォーマット化される最初の行に特別なスタイルを適用するために使用する。

<STYLE TYPE="text/css">

P:first-line { font-variant: small-caps }

</STYLE>

<P>The first line of an article in Newsweek.

テキストベースの UA では，これは次のとおりフォーマット化される。

THE FIRST LINE OF AN

article in Newsweek.

この例の仮想タグ列を，次に示す。

<P>

<P:first-line>

The first line of an

</P:first-line>

article in Newsweek.

</P>

'first-line'終了タグは，キャンバスにフォーマット化される最初の行の行末に挿入される。

'first-line'疑似要素は，ブロックレベル要素だけに付与することができる。

'first-line'疑似要素は，行内要素に類似しているが，制限を伴う。次の特性だけが，'first-line'要素に適用される。フォント特性(**5.2**)，カラー及び背景特性(**5.3**)，'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**)。

ある最初の行があるブロックレベル要素 A 及び A の先祖 B の最初の行であるとき，仮想タグ列は次のとおりとなる。

```
<B>...
  <A>...
    <B:first-line>
      <A:first-line>
        This is the first line
      </A:first-line>
    </B:first-line>
    ...
  </A>
  ...
</B>
```

最初の行のすべての仮想タグは, 最も小さい囲みブロックレベル要素の内側にあり, 仮想タグ A:first-line 及び B:first-line の入れ子順は，要素 A 及び B のそれと同じになる。

ブロックレベル要素の“最初のフォーマット済み行”は，要素フローにおける最初の行であり，つまりどんな浮動をも無視している。例えば,

```
<div>
  <p style="float: left">Floating paragraph...</p>
  <p>First line starts here...</p>
</div>
```

において，最初の p はフローから外れるので，選択子'div:first-line'は 2 番目の p の最初の行に適用される。

2.4　'first-letter'擬似要素　'first-letter'擬似要素は, 表示上のよく使う組み効果である"イニシャルキャップ"及び"ドロップキャップ"に使用する。それは，'float'特性が'none'であれば，行内要素に類似し，そうでなければ，浮動要素に類似する。'first-letter'疑似要素に適用される特性を次に示す。フォント特性(**5.2**)，カラー及び背景特性(**5.3**)，'text-decoration'(**5.4.3**)，'vertical-align'('float' が 'none' の場合だけ，**5.4.4**)，'text-transform'(**5.4.5**)，'line-height'(**5.4.8**)，余白特性(**5.5.1**～**5.5.5**)，パディング特性(**5.5.6**～**5.5.10**)，境界特性(**5.5.11**～**5.5.22**)，'float'(**5.5.25**)，並びに'clear'(**5.5.26**)。

どのようにしてドロップキャップ頭文字を 2 行にわたらせるかを次に示す。

```
<HTML>
 <HEAD>
  <TITLE>Title</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>The first</SPAN> few words of an article in The Economist.</P>
 </BODY>
</HTML>
```

UA が'first-letter'疑似要素をサポートする場合（多くの場合サポートしないが），この例は次のとおりフォーマット化される。

```
——
 | HE FIRST few
 | words of an
article in the
Economist.
```

仮想タグ列は，次のとおり。

```
<P>
<SPAN>
<P:first-letter>
T
</P:first-letter>he first
</SPAN>
few words of an article in the Economist.
</P>
```

'first-line'疑似要素の開始タグは，それが付いている要素の開始タグの後の右に挿入されるが，'first-letter'疑似要素は，内容（つまり，最初の文字）についてタグ付けすることに注意されたい。

UA は，どの文字が'first-letter'要素の中にあるかを定義する。通常は，最初の文字に前置する引用が含まれなければならない。

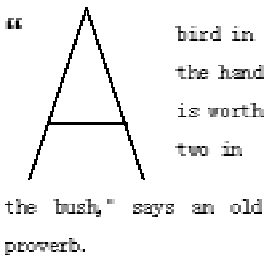

図 2.1　ドロップキャップ

段落が他の句読点（例えば，括弧及び省略点），又は通常は字としては扱われない他の文字（例えば，小数点及び数学記号）で始まるとき，'first-letter'疑似要素は通常無視される。

字の組合せ(letter combination)の扱い方に関する固有の規則をもつ言語がある。例えばオランダ語では，字の組合せ"ij"が語の始めに現れると，それらはどちらも'first-letter'疑似要素として考慮される。

'first-letter'疑似要素は，ブロックレベル要素だけに付与される。

2.5　選択子中の擬似要素　文脈依存選択子において，疑似要素は選択子の最後だけに置くことができる。

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

疑似要素は，選択子の中でクラスと組み合わせることができる。

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

<P CLASS=initial>First paragraph</P>
```

この例は，'CLASS=initial'であるすべての'P'要素の最初の字を赤にする。クラス又は疑似クラスと組み合わされるとき，疑似要素は，選択子の最後で指定されなければならない。選択子ごとに一つだけの疑似要素を指定できる。

2.6　複数疑似要素　疑似要素は幾つか組み合わせることができる。

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

<P>Some text that ends up on two lines</P>
```

この例では，各'P'要素の最初の字は，24 ポイントのフォントサイズで緑になる。段落の残りは赤であるが，（キャンバスにフォーマット化される）最初の行の残り（最初の字以外）は青になる。語"ends"の前で改行する場合，仮想タグ列は次のとおりとなる。

```
<P>
<P:first-line>
<P:first-letter>
S
</P:first-letter>ome text that
</P:first-line>
ends up on two lines
</P>
```

'first-letter'要素は，'first-line'要素の内部にあることに注意されたい。'first-line'に設定された特性は，'first-letter'に継承されるが，同じ特性が'first-letter'に設定されると上書きされる。

疑似要素が実要素を割るとき，必要な追加タグを仮想タグ列に再生成しなければならない。例えば，SPAN 要素が</P:first-line>タグをまたぐとき，SPAN 終了タグ及び SPAN 開始タグが再生成されて，仮想タグ列は次のとおりとなる。

```
<P>
<P:first-line>
<SPAN>
This text is inside a long
</SPAN>
</P:first-line>
<SPAN>
span element
</SPAN>
```

3. 段階 CSS においては，複数のスタイルシートが同時に表示に影響を与えることができる。この機能は，モジュール化，及び文書作成者と読者とのバランスという二つの主要な理由による。

a) モジュール化 スタイルシートの設計者は，冗長性を小さくするために，幾つかの（部分）スタイルシートを組み合わせることができる。

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

H1 { color: red }        /* override imported sheets */
```

b) 文書作成者と読者とのバランス 文書作成者及び読者のどちらも，スタイルシートによって表示に影響を与えることができる。それを行うために，彼らは同じスタイルシート言語を用いて，だれもが出版者になれるというウェブの基本機能を反映する。UA は，個人のスタイルシートを参照する機構を自由に選択できる。

表示に影響を与えるスタイルシートの間で競合が生じることが，時々ある。競合は，重みをもつ各スタイル規則に基づいて，解決される。デフォルトでは，読者の規則の重みは，文書作成者の文書における規則の重みより小さい。つまり，受領文書のスタイルシートと読者の個人シートとの間で競合があると，文書作成者の規則が採用される。読者の規則及び文書作成者の規則のどちらも，UA のデフォルト値を上書きする。

インポートしたスタイルシートも，インポートした順序に，次に規定する段階規則に従って，互いに段階的となる。スタイルシートそのものの中で指定された規則はすべて，インポートされたスタイルシートの規則を上書きする。つまり，インポートされたスタイルシートは，スタイルシートそのものの中の規則より，段階順序が小さい。インポートされたスタイルシートは，再帰的に他のスタイルシートをインポートし上書きすることができる。

CSS1 においては，'@import'ステートメントは，スタイルシートの最初に，すべての宣言の前に現れなければならない。その結果，スタイルシートそのものの中の規則が，インポートされたスタイルシートの中の規則を上書きすることを容易に見ることができる。

3.1 'important' スタイルシート設計者は，その宣言の重みを増加できる。

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

この例では，最後の宣言は通常の重みをもつが，最初の三つの宣言は増加した重みをもつ。

重要宣言をもつ読者の規則は，通常宣言をもつ文書作成者の規則を上書きする。重要宣言をもつ文書作成者の規則は，重要宣言をもつ読者の規則を上書きする。

3.2 段階順序 競合する規則は，CSS 機構にとって本質的であり，要素と特性との組合せに関する値を見つけるために，次のアルゴリズムに従わなければならない。

1) 問題となっている要素と特性との組合せに適用する宣言を見つける。問題の要素に選択子が一致すれば，宣言が適用される。適用される宣言がなければ，継承された値を用いる。継承された値がない場合（'HTML'要素の場合及び継承しない特性の場合）は，初期値を用いる。

2) 明示的な重みによって宣言をソートする。'!important'のマークのある宣言は，マークなしの（通常の）宣言より重みが大きい。

3) 出典によってソートする。文書作成者のスタイルシートは読者のスタイルシートを上書きし，読者の

スタイルシートは UA のデフォルト値を上書きする。インポートされたスタイルシートは，インポート先のスタイルシートと同じ出典をもつ。

4) 選択子の固有性によってソートする。固有性の高い選択子は，一般の選択子を上書きする。固有性を見つけるために，選択子の中の ID 属性の個数を数え(a)，選択子中の CLASS 属性の個数を数え(b)，選択子中のタグ名の個数を数える(c)。（大きな基数をもつ数値表記システムで）これらの三つの個数を結合したものが，固有性を与える。例を次に示す。

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

疑似要素及び疑似クラスは，それぞれ通常の要素及び通常のクラスとして数えられる。

5) 指定された順序でソートする。二つの規則が同一の重みをもつ場合には，後で指定されたものが勝つ。インポートされたスタイルシートにおける規則は，スタイルシートそのものにおけるどんな規則よりも前にあるとする。

一つの規則が，同じ要素と特性との組合せに適用される他の規則よりも大きい重みをもつときはいつでも，特性の検査を終了できる。

この方策は，文書作成者のスタイルシートに，読者のそれよりもかなり大きい重みを与える。したがって，読者があるスタイルシートの影響を，例えばプルダウンメニューによって遮断できるということが，重要になる。

要素の'STYLE'属性の中の宣言は，スタイルシートの終わりに指定される ID ベースの選択子をもつ宣言と同じ重みをもつ。

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

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

この例において，'P'要素のカラーは赤になる。固有性は両方の宣言について同一であるが，'STYLE'属性における宣言は，段階規則 **5)**によって，'STYLE'要素のそれを上書きする。

UA は，他の HTML スタイル属性，例えば'ALIGN'，を尊重することを選択してよい。その場合，これらの属性は，1 に等しい固有性をもって，対応する CSS 規則に翻訳される。その規則は，文書作成者のスタイルシートの開始点にあることを前提とし，後続のスタイルシート規則によって上書きされ得る。遷移フェーズにおいては，この方策は，スタイル属性がスタイルシートと共存することを容易にする。

4. フォーマット化モデル CSS1 は，簡単なボックス指向のフォーマット化モデルを前提とする。そのモデルでは，フォーマット化される各要素は，1 個以上のく（矩）形のボックスに帰着する。('display'値が'none'である要素は，フォーマット化されず，したがってボックスには帰着しない。) すべてのボックスは，一つのコア内容領域をもち，それにはオプションの周囲を囲むパディング領域，境界領域及び余白領域を伴う。

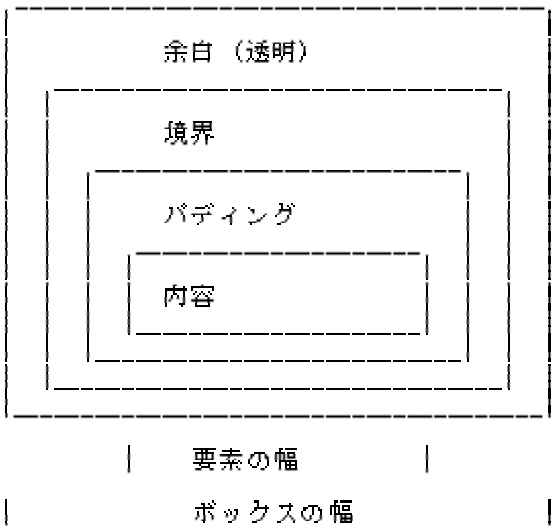

図 4.1　フォーマット化モデル

余白，境界及びパディングのサイズは，それぞれ余白特性(**5.5.1**～**5.5.5**)，パディング特性(**5.5.6**～**5.5.10**)，及び境界特性(**5.5.11**～**5.5.22**)を用いて設定される。パディング領域は，その要素そのものと同じ背景を用いる（背景特性(**5.3.2**～**5.3.7**)を設定する。）。境界のカラー及びスタイルは，境界特性を用いて設定される。余白は常に透明であり，親要素が透けて見える。

ボックスのサイズは，要素の幅（つまり，フォーマット済みのテキスト又は画像），パディング領域，境界領域及び余白領域の合計になる。

フォーマタの観点では，二つの主要な要素の型，つまりブロックレベル及び行内がある。

4.1　ブロックレベル要素　'display'値が'block'又は'list-item'である要素を*ブロックレベル*要素とする。浮動要素（'float'値が'none'ではない要素）も，ブロックレベル要素と見なせる。

次の例は，余白及びパディングが，二つの子をもつ'UL'要素をどのようにフォーマット化するかを示す。図を簡単にするため，境界はない。この例における 1 字の'constants'は，CSS1 構文に文法的に合わないが，スタイルシートの値を図に結び付けるのにつごうがよい。

```
<STYLE TYPE="text/css">
  UL {
    background: red;
    margin: A B C D;
    padding: E F G H;
  }
  LI {
    color: white;
    background: blue;        /* so text is white on blue */
    margin: a b c d;
```

```
      padding: e f g h;
   }
</STYLE>
..
<UL>
   <LI>1st element of list
   <LI>2nd element of list
</UL>
```

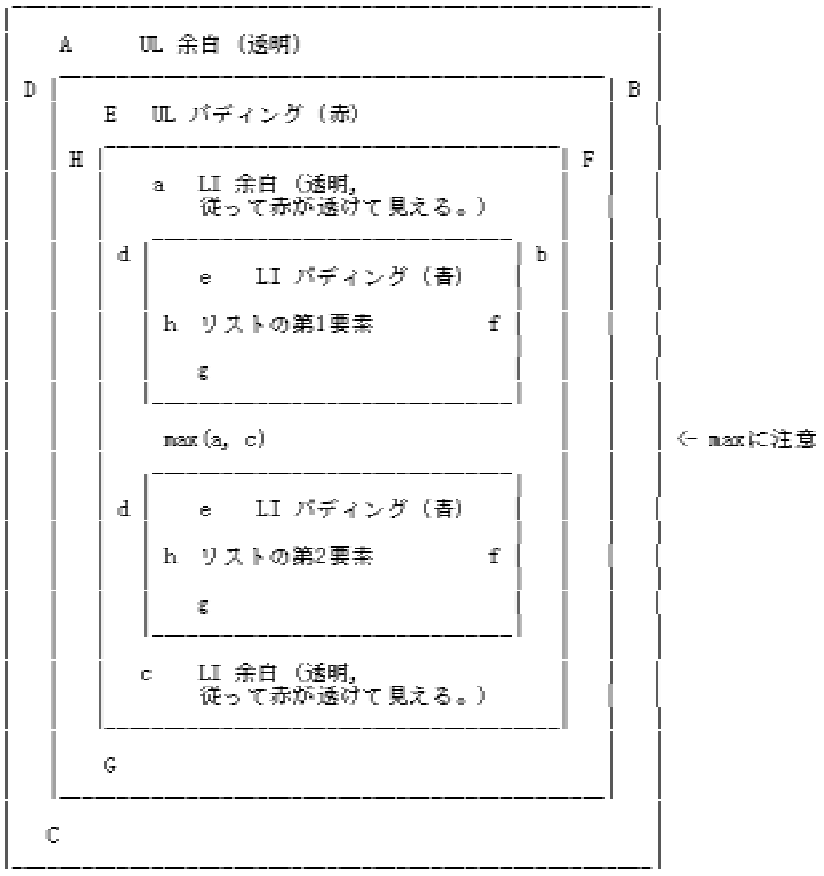

図 4.2　二つの子をもつ'UL'要素のフォーマット

技術的には，パディング特性及び余白特性は継承されない。しかしこの例が示すとおり，要素の配置は先祖及び兄弟に対して相対的であり，これらの要素のパディング特性及び余白特性はそれらの子に影響を与える。

もしこの例に境界があったら，それはパディングと余白との間に現れる。

次の図は，幾つかのよく使う用語を導入している。

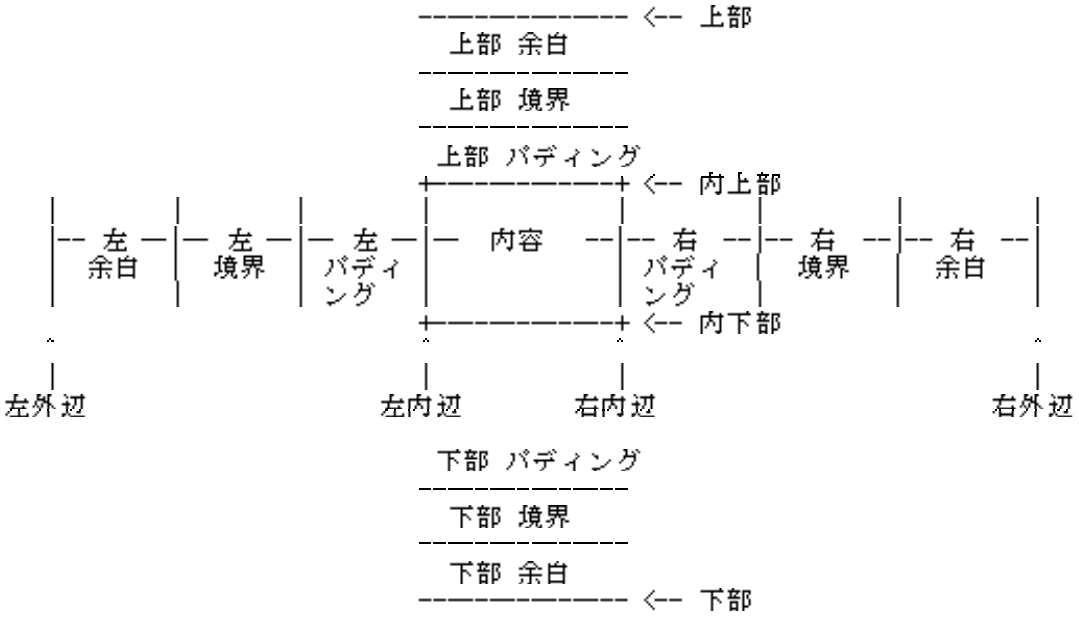

図 4.3　よく使う用語

*左外辺*は，パディング，境界及び余白を考慮した要素の端とする。*左内辺*は，パディング，境界又は余白のすべての内側にある内容だけの端とする。右側についても同様とする。*上部*は，パディング，境界及び余白のすべてを含む要素の上端とする。それは，行内要素及び浮動要素だけのために定義され，非浮動のブロックレベル要素に対しては定義されない。*内上部*は，パディング，境界又は余白の内側にある内容の上端とする。*下部*は，要素の下端であって，パディング，境界及び余白のすべての外側とする。それは，行内要素及び浮動要素だけのために定義され，非浮動のブロックレベル要素に対しては定義されない。*内下部*は，パディング，境界及び余白のすべての内側にある要素の下端とする。

要素の*幅*は，内容の幅であって，左内辺と右内辺との間の距離とする。*高さ*は，内容の高さであって，内上部と内下部との間の距離とする。

4.1.1　上下方向フォーマット化　非浮動ブロックレベル要素の余白の幅は，周囲のボックスの端への最小距離を指定する。二つ以上の隣接する上下方向の余白（つまりそれらの間には，境界，パディング又は内容はない。）は，余白の最大値を使用すると，つぶれる。多くの場合，上下方向の余白をつぶすと，視覚的には好まれる結果となり，設計者の期待するものに近づく。**図 4.2** の例では，最初の LI 要素の'margin-bottom'及び 2 番目の LI 要素の'margin-top'の最大値を使用することによって，二つの'LI'要素の間の余白は，つぶされている。同様に，'UL'要素と最初の'LI'要素（'E'定数）との間のパディングがゼロだと，UL 要素及び最初の LI 要素の余白はつぶされる。

余白が負の場合，負の隣接余白の絶対最大値は，正の隣接余白の最大値から差し引かれる。正の余白がない場合には，負の隣接余白の絶対最大値は，ゼロから差し引かれる。

4.1.2　左右方向フォーマット化　非浮動ブロックレベル要素の左右方向の位置及び大きさは，次の七つの特性で決定する。七つの特性とは，'margin-left'（左余白），'border-left'（左境界），'padding-left'（左パディング），'width'（幅），'padding-right'（右パディング），'border-right'（右境界）及び 'margin-right'（右余白）とする。これら七つの和は，常に，親要素の'width'に等しい。

デフォルトで，要素の'width'は，'auto'とする。要素が置換要素でない場合には，ここは'width'が UA によって計算され，先の七つの特性の和が親の幅に等しくなることを意味する。要素が置換要素の場合には，'width'に対する'auto'の値は，自動的に，その要素の基本幅によって置き換えられる。

七つの特性の内，'margin-left'，'width'及び'margin-right'の三つは，'auto'に設定できる。置換要素に対しては，'width'についての'auto'の値は，基本の幅で置き換えられる。そこで，置換要素に対しては，二つの'auto'値だけが存在できる。

'width'は，負でない UA 定義の最小値をもつ。（ただし，この UA 定義の最小値は，要素ごとに変わってもよいし，他の特性に依存してもよい。）'width'を明示的に設定した，又は'width'が'auto'であって以下の規則によって'width' が非常に小さくなった，のいずれかの理由のために，'width'がこの限界を下回った場合は，その値は，代わりに，最小値で置き換えられる。

'margin-left'，'width'又は'margin-right'の*一つだけ*が'auto'の場合，UA は，その特性に，七つの特性の和を親の幅に等しくする値を割り当てる。

'margin-left'，'width'又は'margin-right'の一つも'auto'*でない*場合，'margin-right'を'auto'に割り当てる。

'margin-left'，'width'又は'margin-right'の*二つ以上*が'auto'の場合であって，その一つが'width'の場合，'width'以外のもの（'margin-left'及び／又は'margin-right'）は，ゼロに設定し，'width'は，七つの和を親の幅に等しくするのに必要な値をとる。

そうでない場合であって，'margin-left'及び'margin-right'の両方が'auto'の場合，これらを等しい値に設定する。これによって，親の中で要素が中心に位置する。

行内又は浮動とする要素において，七つの特性の一つに対する値として，'auto'を設定する場合，ゼロと設定したものとして取り扱う。

上下方向の余白とは違い，左右方向の余白は，つぶれることはない。

4.1.3　リスト項目要素　'list-item'の'display'特性値をもつ要素は，ブロックレベル要素としてフォーマット化するが，リスト項目マーカが先行する。マーカの型は，'list-style'特性で決定する。マーカは，'list-style' 特性の値に従って，位置付ける。

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

<UL>
  <LI>first list item comes first
  <LI>second list item comes second
</UL>

<UL CLASS=COMPACT>
  <LI>first list item comes first
  <LI>second list item comes second
</UL>
```

この例は，次のとおりにフォーマット化される。

```
* first list item
```

comes first
* second list item
comes second
* first list
item comes first
* second list
item comes second

左から右へと書く横書きのテキストでは，マーカは，ボックスの右側に存在する。

4.1.4 浮動要素 'float' 特性を使用して, 要素が, 要素の正規フローの外側に存在する宣言を可能とする。このとき，要素は，ブロックレベル要素としてフォーマット化される。例えば，画像の'float'特性を'left'に設定することによって，他のブロックレベル要素の余白，パディング又は境界に達するまで，その画像は左に移動する。正規フローは，右側に巻き込まれる。要素それ自体の余白，境界及びパディングは，そのまま残り，余白は，隣接する余白でつぶれることはない。

浮動要素は，次の制約に従って位置付けされる（用語の定義は，**4.1** を参照されたい。）。

a) 左浮動要素の左外辺は，その親要素の左内辺の左側とならなくともよい。右浮動要素に対しても同様とする。

b) 左浮動要素の左外辺が，（HTML のテキストにおける出現順序で,）それより前のすべての左浮動要素の右外辺の右側となるか，左浮動要素の上部が，それより前のすべての左浮動要素の下部よりも下とならなければならない。右浮動要素に対しても，同様とする。

c) 左浮動要素の右外辺は, その左浮動要素の右側にある右浮動要素の左外辺の右側とならなくともよい。右浮動要素に対しても，同様とする。

d) 浮動要素の上部は，その親の内部の上部よりも高くなくともよい。

e) 浮動要素の上部は，それより前の浮動要素又はブロックレベル要素の上部よりも高くなくともよい。

f) 浮動要素の上部は，HTML ソースで，その浮動要素に先行する内容をもつ*ラインボックス*（**4.4** を参照）の上部よりも高くなくともよい。

g) 浮動要素は，可能な限り高く位置付けなければならない。

h) 左浮動要素は，可能な限り左側の遠くに置かなければならない。右浮動要素は，可能な限り右側の遠くに置かなければならない。より高い位置の方が，より左又はより右の位置よりも望ましい。

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

<BODY>
  <P>
    <IMG SRC=img.gif>
    Some sample text that has no other...
</BODY>
```

この例は，次のとおりにフォーマット化される。

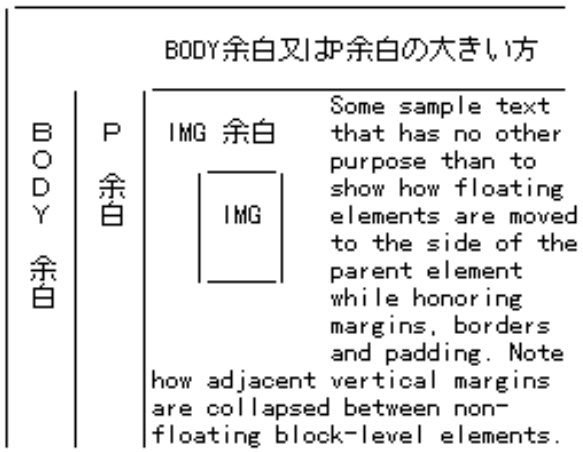

図 4.4　浮動要素の表示例

'P'要素の余白は，浮動な'IMG'要素を囲い込むことに注意すること。

浮動要素が，他の要素の余白領域，境界領域及びパディング領域と重なることが可能な場合には，次の二つの状況がある。

a) 浮動要素が，負の余白をもつ場合。浮動要素の負の余白は，他のブロックレベル要素のものとみなされる。

b) 浮動要素が，それを内部に含む要素よりも左右又は上下に大きい場合。

4.2　行内要素　ブロックレベル要素としてフォーマット化されない要素は，*行内*要素とする。行内要素は，他の要素と行間を共有できる。次に例を示す。

```
<P>Several <EM>emphasized</EM> words <STRONG>appear</STRONG>.</P>
```

'P'要素は，通常，ブロックレベルとなる。一方，'EM'及び'STRONG'は，行内要素となる。'P'要素が，一つの行の全要素をフォーマット化するのに十分の幅をもつ場合，その行に二つの行内要素が存在することになる。

Several *emphasized* words **appear**.

一つの行に十分な余地がない場合は，一つの行内要素がいくつかのボックスに分離する。

```
<P>Several <EM>emphasized words</EM> appear here.</P>
```

この例は，次のとおりにフォーマット化される。

Several *emphasized*

words appear here.

行内要素が，余白，境界，パディング又は付加されたテキスト修飾をもつ場合，これらは，要素が分断された場所では，いかなる効果ももたない。

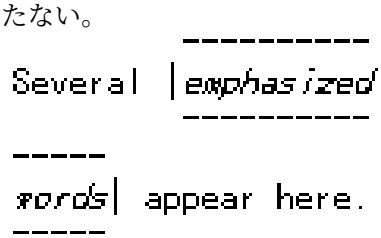

図 4.5　行内要素の表示例

（この"図"は，ASCII による図形を使用したために，多少ともゆがんでしまっている。行の高さを計算

する方法の規定に関しては，**4.4** を参照のこと。）

4.3　置換要素　置換要素は，その要素から指し示された要素で置き換えられた要素とする。例えば，HTML では，'IMG'要素は，'SRC'属性によって指し示された画像で置換えられる。置換要素は，それ自体の基本寸法をもってくると仮定できる。'width'特性の値が'auto'の場合，要素の幅として基本幅を使用する。スタイルシートで，'auto'以外の値を指定している場合は，この値を使用し，置換要素は，それに従って，再度大きさを調整する。（この方法は，メディア型による。）'height'特性も，同じ方法で使用する。

置換要素は，ブロックレベル又は行内の要素となることができる。

4.4　行の高さ　すべての要素は，原理的には，テキストの行の高さ全体を与える，'line-height'特性をもつ。行の高さに達するために，行のテキストの上下に，スペースを追加する。例えば，テキストが 12 ポイントの高さをもち，'line-height'が'14pt'に設定されている場合，2 ポイントの余分なスペースを，通常は，行の上に 1 ポイント及び行の下に 1 ポイント，追加する。空要素は，内容をもつ要素としてこの計算に影響する。

フォントサイズと'line-height'との差を，*リーディング(leading)*という。リーディングの半分を，*半リーディング*という。フォーマット化の後では，各行は，長方形の*行ボックス*を形成する。

（行にいくつかの行内要素が存在するために，）テキストの行が，違った'line-height' 値をもつセクションを含む場合，これらのセクションの各々は，それ自体の半リーディングを上下にもつ。行ボックスの高さは，最も高さの高いセクションの上部から最も高さの低いセクションの下部までとする。上部及び下部は必ずしも最も背の高い要素に対応するとは限らないことに注意すること。これは，要素は，'vertical-align'特性で，上下に位置づけできることによる。段落を形成するために，各行ボックスは，その前の行のすぐ下に積み上げる。

非置換行内要素の上下の，パディング，境界又は余白のいずれも，行の高さには影響しないことに注意しなければならない。言い換えると，'line-height'が，選択したパディング又は境界に対して小さ過ぎる場合は，他の行のテキストに重なる。

行における置換要素（例えば，画像）は，その置換要素の上部（つまり，そのパディング，境界及び余白のすべてを含む。）が，最も背の高いテキストセクションの上にあるか，下部が最も背の低いものの下にある場合には，行ボックスをより大きくすることができる。

通常の場合で，段落を通して'line-height'の値がただ一つだけ存在し，背の高い画像が存在しないときには，これまでの定義によって，続く行の基底線は，ちょうど'line-height'だけ離れることが確実になる。これは，例えば表における，違ったフォントを使うテキストの行をそろえなければならないときに重要となる。

このことは，二つの隣接する行のテキストが，互いに重なり合わないことを排除するわけでは無いことに注意。'line-height'は，テキストの高さよりも小さくともよい。この場合には，リーディングは，負となる。これは，テキストが，（例えば，大文字だけを含むので，）下方に伸びる文字を含まない場合に役に立ち，行を互いにより近づけることが可能となる。

4.5　キャンバス　キャンバスは，UA の，文書を可視化する描画の表面の一部とする。文書のいかなる構造的な要素もキャンバスには対応しない。このことは，文書をフォーマット化するときに，次の二つの問題を生じる。

a)　キャンバスの寸法を，どの位置から設定するのがよいか。

b)　文書がキャンバス全体を覆わないない場合に，この領域をどのように可視化するのがよいか。

最初の問題に対する答は，合理的には，キャンバスの初期幅はウィンドウサイズに基づくとするものに

なる。しかし，CSS1 では，この問題は，UA に決定を任せることとする。ウィンドウの大きさが変わった場合には，UA がキャンバス幅を変えると期待するのも理にかなっている。しかし，これも CSS1 の適用範囲外とする。

HTML 拡張は，2 番目の問題に対して優先度を設定する。すなわち，'BODY'要素の属性が，キャンバス全体の背景を設定する。設計者の期待をサポートするために，CSS1 では，キャンバス背景を見つけるために，特別な規則を導入する。

'HTML'要素の'background'値が'transparent'と違う場合，その値を使用し，そうでない場合には，'BODY'要素の'background'値を使用する。その結果の値が'transparent'のとき，描写は不定とする。

この規則は，次を許す。

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

この例では，キャンバスは，"marble"で覆われる。'BODY'要素の背景は，（キャンバスを完全に覆うかもしれないし，覆わないかもしれないが,）赤となる。

キャンバスに情報を伝える他の方法が利用可能となるまで，キャンバス特性を'BODY'要素で設定することが望ましい。

4.6 'BR'要素 現在の CSS1 特性及び値は，'BR'要素の振る舞いを示すことができない。HTML では，'BR'要素は，語の間の行替えを指定する。実効的には，この要素は，行替えで置き換えられる。CSS の将来の版では，追加内容及び置換内容を処理するかもしれない。しかし，CSS1 に基づくフォーマタは，'BR'を特別に処理しなければならない。

5. CSS1 特性 スタイルシートは，値をスタイル特性に割り当てることによって，文書の表示に影響する。5.では，CSS1 で定義済みのスタイル特性を示し，それら特性の可能な値の対応する一覧を示す。

5.1 特性値の記法 この規格の以降においては，各特性に対して許される値を次の構文で示す。

値： N | NW | NE
値： [<length> | thick | thin]{1,4}
値： [<family-name> ,]* <family-name>
値： <url>? <color> [/ <color>]?
値： <url> || <color>

"<"と">"との間の語は，値の型を示す。最も共通の型は，<length>，<percentage>，<url>，<number>及び<color>とする。これらは，**6.**で示す。より特殊な型（<font-family>及び <border-style>）は，対応する特性のもとで示す。

それ以外の語は，キーワードとし，引用符なしでそのまま出現しなければならない。スラッシュ(/)及びコンマ(,)も，そのまま出現しなければならない。

いくつかの並置されたものは, それらすべてが, 与えられた順番で出現しなければならないことを示す。縦棒(|)は，選択肢を分離する。それらの一つが出現しなければならない。二重縦棒(A || B)は，A 若しくは B 又は両方が，任意の順番で，出現しなければならないことを示す。角括弧([])は，グループ化のために使用する。並びは，二重縦棒よりも優先し，二重縦棒は，縦棒よりも優先する。そこで，"a b | c || d e"は，"[a b] | [c || [d e]]に等価になる。

型，キーワード又は括弧付けしたグループのすべてに，次の修飾子の一つが続いてもよい。

a) アスタリスク(*)は，先行する，型，語又はグループが，ゼロ回以上繰り返すことを示す。

b) プラス(+)は，先行する，型，語又はグループが，1 回以上繰り返すことを示す。

c) 疑問符(?)は，先行する，型，語又はグループをオプションとすることを示す。

d) 波括弧({A,B})内の数字の対は，先行する，型，語又はグループが，少なくとも A 回，多くとも B 回，繰り返すことを示す。

5.2 フォント特性 フォント特性の設定は，スタイルシートを共通的に使用するときに使われる。残念なことに，フォントを分類する，正しく定義され普遍的に受け入れられた分類学は，存在しない。一つのフォントファミリに適用される用語が，他のフォントファミリには適切でないこともある。例えば，'イタリック(italic)'は，傾斜したテキストをラベル付けするために，共通に使用するが，傾斜したテキストは，*Oblique，Slanted，Incline，Cursive* 又は *Kursiv* としてラベル付けすることもできる。そこで，典型的なフォント選択特性を，特定のフォントに対応付けることは，単純な問題ではない。

CSS1 は，次の各特性，font-family，'font-style'，'font-variant'，'font-weight'，'font-size'及び'font'を定義する。

5.2.1 フォント一致化 フォント特性についての普遍的に受け入れられた分類学は存在しないので，書体に対する特性の一致化は，注意深く行わなければならない。特性は，正しく定義された順番で一致化する。これによって，この一致化過程の結果を，UA 間でできるだけ一貫性を保つことを保証する。ただし，書体の同じライブラリを，UA の各々が所持していることを仮定する。

a) **ステップ 1** UA は、その UA が知っているフォントすべての関連する CSS1 特性データベースを作成（又はアクセス）する。UA は，あるフォントがローカルにインストールされている場合，又は以前にウェブからダウンロードされた場合には，そのフォントを知っていることもある。全く同じ特性をもつ二つのフォントが存在する場合は，どちらかを無視する。

b) **ステップ 2** 要素が与えられると，その要素における各文字に対して，UA は，その要素に適用するフォント特性を組み立てる。特性の完全な集合を使って，UA は，暫定的なフォントファミリを選択するために，'font-family'特性を使用する。残りの特性は，各特性とともに示される一致化基準に従って，そのファミリに対して，試験する。残りの特性すべてが一致する場合は，その暫定的なフォントファミリを，与えられた要素に対する一致化書体とする。

c) **ステップ 3** ステップ 2 で処理される'font-family'内の一致化書体が存在せず，そのフォント集合にその次の代替'font-family'が存在する場合は，その代替'font-family' でステップ 2 を繰り返す。

d) **ステップ 4** ステップ 2 で選択した暫定的なフォントファミリの書体は存在するが，着目している文字に対するグリフを含まず，フォント集合にその次の代替'font-family'が存在する場合は，その代替'font-family' でステップ 2 を繰り返す。フォント及び文字符号化の規定に関しては，**附属書 C**を参照すること。

e) **ステップ 5** ステップ 2 で選択したフォントファミリ内にフォントが存在しない場合は，UA に依存するデフォルト'font-family'を使用し，デフォルトフォント内で取得可能な最も一致するものを使用して，ステップ 2 を繰り返す。

このアルゴリズムは，各文字に対する CSS1 特性の再調査を回避する最適化を可能とする。

このステップ 2 からの特性ごとの一致化規則は，次による。

a) 最初に，'font-style'を試験する。CSS キーワード'italic'（こちらが望ましい）又は'oblique'のラベルをもつ UA フォントデータベースで，いずれかの書体が存在する場合は，フォントスタイルの値は満たされて，'italic'となる。そうでないときには，'font-style'の値それ自体が，一致しなければならないか，又はフォントスタイルは満たされない。

b) 次に，'font-variant'を試験する。'normal'は，'small-caps'というラベル付けされていないフォントと一致する。'small-caps'は，次のいずれかと一致する。

1) 'small-caps'とラベル付けされているフォント。

2) スモールキャップを合成するフォント。

3) すべての小文字を大文字に置き換えた場合のフォント。

スモールキャップフォントは，本文フォントから大文字の大きさを電子的に変えて合成してもよい。

c) 次に，'font-weight' を一致させる。これは，必ず満たされる。（**5.2.5** 'font-weight'を参照すること。）

d) UA に依存する許容余白内で，'font-size'を一致しなければならない。（通常は，大きさを変えられるフォントのサイズは，最も近い画素の大きさに丸められる。一方，ビットマップ化したフォントに対する許容限度は，大きくとも，20%となる。）他の特性における'em'値などの更なる計算は，使用する'font-size'値に基づくものであって，指定しない。

5.2.2 'font-family'

値： [[<family-name> | <generic-family>],]* [<family-name> | <generic-family>]

初期値： UA 固有

適用対象： すべての要素

継承： yes

パーセント値： N/A

値は，フォントファミリ名及び／又は一般ファミリ名の優先度付けされたリストとする。多くの他のCSS1 特性とは違い，値はコンマで分離する。これは，分離したものが選択肢となることを示す。

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

リストの値には，次の二つの型がある。

a) **<family-name>** 選択したフォントファミリの名前。例では，"gill"及び"helvetica"がフォントファミリとなる。

b) **<generic-family>** 例では，最後の値が一般ファミリ名となる。次の一般ファミリを定義する。

1) 'serif'（例えば，Times）

2) 'sans-serif'（例えば，Helvetica）

3) 'cursive'（例えば，Zapf-Chancery）

4) 'fantasy'（例えば，Western）

5) 'monospace'（例えば，Courier）

スタイルシートには，一般フォントファミリを最後の選択肢として提供するのがよい。

空白を含むフォント名は，引用符で囲むことが望ましい。

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


<BODY STYLE="font-family: 'My own font', fantasy">
```

引用符を省略した場合には，フォント名の前後のいかなる空白文字も無視し，フォント名の中の空白文字の列は，一つのスペースに変換する。

5.2.3 'font-style'

値： normal | italic | oblique

初期値： normal

適用対象： すべての要素

継承： yes

パーセント値： N/A

'font-style'特性は，一つのフォントファミリ内での，本文書体（normal face のこと。"roman"又は"upright"として参照することがある。），イタリック体（italic face のこと。）及び斜体（oblique face のこと。）のうちから選択する。

'normal'の値は，UA のフォントデータベースで'normal'と分類されるフォントを選択する。一方，'oblique'は，'oblique'とラベル付けされたフォントを選択する。'italic'の値は，'italic'とラベル付けされたフォントを選択するか，それが使用できない場合は，'oblique'とラベル付けされたフォントを選択する。

UA のフォントデータベースで，'oblique'とラベル付けされたフォントは，実際には，本文書体を電子的に傾斜することで生成してもよい。

名前に, Oblique, Slanted 又は Incline をもつフォントは, UA のフォントデータベースでは, 通常は, 'oblique'とラベル付けされている。名前に，*Italic，Cursive* 又は *Kursiv* をもつフォントは，通常は，'italic'とラベル付けされている。

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

この例では，'H1'内の強調されたテキストは，本文書体で現れる。

5.2.4 'font-variant'

値： normal | small-caps

初期値： normal

適用対象： すべての要素

継承： yes

パーセント値： N/A

フォントファミリ内の変化の型には，スモールキャップもある。スモールキャップフォントでは，小文字が大文字と同じに見えるが，字の大きさが小さく，字の比率が少し違う。'font-variant'特性が，そのフォントを選択する。

'normal'の値は，スモールキャップフォントではないフォントを選択する。'small-caps'は，スモールキャップフォントを選択する。CCS1 では，通常フォントをとり，大きさを変えた大文字で小文字を置き換えて生成し，スモールキャップとしている場合を 許容する。（ただし，これを要求するわけではない。）最後の手段としては，大文字を，スモールキャップフォントの代わりとして使用する。

次の例は，斜体スモールキャップで強調した語をもつ，スモールキャップの'H3'要素を生じる。

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

同様に，旧様式の数字，スモールキャップの数字，凝縮した字，拡張した字などをもつフォントの フォントファミリの他の変種が存在してもよい。CSS1 は，これらを選択する特性はもたない。

CSS1 コア： この特性によって，テキストの大文字への変換が生じる限りは，'text-transform'に対するのと同じ考慮が必要となる。

5.2.5 'font-weight'

値： normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

初期値： normal

適用対象： すべての要素

継承： yes

パーセント値： N/A

'font-weight'特性は，フォントの重みを選択する。値'100'から'900'までは，順番のある列を形成し，そこでは，各数は一つ前の数と少なくとも同じ暗さの重みを示す。キーワード'normal'は，'400'の同意語とし，'bold'は，'700'の同意語とする。'normal'及び'bold'以外のキーワードは，フォント名と混同することが多いことが知られており，そこで，九つの値のリストの数値化尺度を選択した。

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

'bolder'及び'lighter'の値は，親から継承する重みとの相対的なフォントの重みを選択する。

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

子要素は，結果として生ずる重みを継承し，キーワードの値を継承しない。

フォント（フォントデータ）は，通常は，フォントの "重み" を示す名前が値となる一つ以上の特性をもつ。これらの重みの名前に対して，受け入れられた普遍的な意味は存在しない。これらの基本的な役割は，一つのフォントファミリ内での違った暗さを区別することにある。フォントファミリ間での利用は，全く一定しない。例えば，ボールド(Bold)と思われるフォントが，フォントの "本文" 書体を設計でどのくらい黒くするかに依存して，*Regular-Bold*，*Roman-Bold*，*Book-Bold*，*Medium-Bold*，*Semi-Bold* 若しくは *DemiBold*, *Bold* 又は *Black* として記述されるかもしれない。名前の標準的な使用法が存在しないので, CSS1における重み特性値は，数値化尺度で与えることとする。これによって，値'400'（又は'normal'）は，そのファミリに対する "本文" 書体に対応する。その書体に関連する重みの名前は，通常は，*Book*，*Regular*，*Roman* 又は *Normal*，時には，*Medium* となる。

一つのファミリ内での他の重みを数字の重み値に関連させるには，そのファミリ内での暗さの順番を保存することだけを意図することになる。しかし, 次の経験的な方法が, 通常の場合の割当て方法を与える。

a) フォントファミリが，（例えば，*OpenType* がそうだが,）既に九つの値をもつ数値化尺度を使用している場合には，そのフォント重みを直接に写像するのがよい。

b) *Medium* というラベルをもつ書体及び *Book*，*Regular*，*Roman* 又は *Normal* というラベルをもつ書体が存在する場合は，*Medium* を通常'500'に割り当てる。

c) "Bold"というラベルが付いたフォントを，重み値'700'に対応させることが多い。

d) ファミリに九つよりも少なくい重みが存在する場合は，"穴"を埋めるためのデフォルトのアルゴリズムを 次のとおりとする。'500'を割り当てていない場合は, それを, '400'と同じフォントに割り当てる。値'600'，'700'，'800'又は'900'のいずれかを割り当てていない場合で，次に暗い割当てキーワードが存在するときには，それと同じ書体に割り当て，そうでないときには，それ以外の次に明るいものに割り当てる。'300'，'200'又は'100'を割り当てていない場合で，次に明るい割当てキーワードが存在するときには，それに割り当て，そうでないときには，それ以外の，次に暗いものに割り当てる。

次の二つの例は，この処理過程を説明する。"Example1"ファミリには，明るい方から暗い方へと，四つの重み，*Regular*，*Medium*，*Bold* 及び *Heavy* が存在すると仮定する。一方，"Example2"ファミリには，六つの重み，*Book*，*Medium*，*Bold*，*Heavy*，*Black* 及び *ExtraBlack* が存在すると仮定する。2 番目の例では，いかにして"Example2 ExtraBlack"をいずれにも割り当てないと決断したかに注意すること。

使用可能書体	割当て	穴の埋め方

"Example1 Regular"	400	100, 200, 300
"Example1 Medium"	500	
"Example1 Bold"	700	600
"Example1 Heavy"	800	900

使用可能書体	割当て	穴の埋め方
"Example2 Book"	400	100, 200, 300
"Example2 Medium"	500	
"Example2 Bold"	700	600
"Example2 Heavy"	800	
"Example2 Black"	900	
"Example2 ExtraBlack"	(none)	

相対的なキーワード'bolder'及び'lighter'の意図は，*そのファミリ内の*書体を暗く又は明るくすることにあり，ファミリは，すべての記号的な重み値とそろ（揃）う書体をもたなくともよいので，'bolder'の一致化は，そのファミリ内のクライアント上で使用可能な次に暗い書体に対して行い，'lighter'の一致化は，そのファミリ内の次に明るい書体に対して行う。正確には，相対的なキーワード'bolder'及び'lighter'の意味は，次による。

a) 'bolder'は，継承したものよりも暗いフォントに割り当てたその次の重みを選択する。その重みが存在しない場合は，継承した値が'900'でないときには，単に，次に暗い数値となる。（そして，フォントはそのままとする。）継承した値が'900'の場合は，結果の重みも'900'とする。

b) 'lighter'も同様とする。ただし，反対の方向となる。継承したものとは違うフォントをもつ，次に明るいキーワードを選択する。ただし，そのフォントが存在する場合に限る。存在しない場合は，次に明るい数値を選択する。（そして，フォントはそのままとする。）

'font-weight'値の各々に対して，より暗い書体が存在することは保証されない。例えば，あるフォントは，本文書体及びボールド体だけをもち，他のフォントは，9 個の違った書体の重みをもつかもしれない。UA が，一つのファミリ内で，書体を重み値に対応付ける方法についての保証はない。与えられた値の書体が，より明るい値の書体よりも暗くない，ということだけを保証する。

5.2.6 'font-size'

値： <absolute-size> | <relative-size> | <length> | <percentage>

初期値： medium

適用対象： すべての要素

継承： yes

パーセント値： 親要素のフォントサイズに相対的

c) **<absolute-size>** <absolute-size>キーワードは，UA が計算し保持するフォントサイズの表へのインデクスとする。可能な値は，[xx-small | x-small | small | medium | large | x-large | xx-large] とする。コンピュータスクリーン上では，隣接するインデクス間は，1.5 のスケール倍率が提案される。すなわち，'medium'フォントが 10pt の場合，'large'フォントは，15pt となる。違った媒体では，違ったスケール倍率を必要とするかもしれない。UA が表を計算するときには，フォントの品質及び利用可能性を考慮することも望ましい。表は，フォントファミリごとに違っていてもよい。

d) **<relative-size>** <relative-size>キーワードは，フォントサイズの表及び親要素のフォントサイズに対し

て，相対的に解釈する。可能な値は，[larger | smaller]とする。例えば，親要素が'medium'のフォントサイズをもつ場合，'larger'の値は，現在の要素のフォントサイズを'large'とする。親要素のサイズが表エントリに近くない場合には，UA は，自由に表エントリの間を内挿するか，最も近いエントリに丸めるかする。数値がキーワードを超える場合は，UA は，表エントリを外挿しなければならないかもしれない。

要素のフォントサイズを計算する場合，長さ及びパーセントの値は，フォントサイズ表を考慮しないほうがよい。

負の値は，許されない。

すべての他の特性において，'em'及び'ex'の長さ値は，現在の要素のフォントサイズを参照する。'font-size'特性において，これらの長さ単位は，親要素のフォントサイズを参照する。

応用は，その文脈に依存して，明示的なサイズを再度解釈し直してもよい。例えば，VR シーンの内部では，フォントは，光景のゆがみのために，違ったサイズを取るかもしれない。

次に例を示す。

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

提案されたスケール倍率 1.5 を使用する場合は，最後の三つの宣言は，同じになる。

5.2.7 'font'

値： [<font-style> || <font-variant> || <font-weight>]? <font-size> [/ <line-height>]? <font-family>

初期値： 短縮形の特性に対して定義されない

適用対象： すべての要素

継承： yes

パーセント値： <font-size>及び<line-height>で許される

'font'特性は，スタイルシート内の同じ位置での，'font-style', 'font-variant', 'font-weight', 'font-size', 'line-height'及び'font-family'を設定するための，短縮形の特性とする。この特性の構文は，フォントに関係した複数の特性を設定するための，既存の印刷上の短縮形記法に基づく。

可能な初期値の定義については，既に定義した特性を参照すること。値を与えない特性は，それらの初期値に設定する。

```
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 }
```

この内の 2 番目の規則では，フォントサイズパーセントの値('80%')が，親要素のフォントサイズを参照する。3 番目の規則では，行の高さのパーセントが，要素それ自体のフォントサイズを参照する。

最初の三つの規則では，'font-style'，'font-variant'及び'font-weight'を明示的に与えていない。これは，これら三つすべてを，それらの初期値('normal')に設定することを意味する。4 番目の規則は，'font-weight'を'bold'に設定し，'font-style'を'italic'に設定し，'font-variant'を暗黙的に'normal'に設定する。

5 番目の規則は，'font-variant'（を，'small-caps'に），'font-size'（を，親のフォントの 120%に），'line-height'

（を，フォントサイズの 120%に）及び'font-family'を（，'fantasy'に,）設定する。さらに，キーワード'normal'を，残りの二つの特性'font-style'及び'font-weight'に適用する。

5.3　カラー及び背景特性　これらの特性は，要素のカラー（*前景の色*ということが多い。）及び要素の背景（つまり，内容を可視化する面）を記述する。背景色及び／又は背景画像を設定できる。画像の位置，画像が繰り返されるかどうか,画像の繰返しの方法及び 画像がキャンバスに対して固定しているのか又はスクロールするのかも，設定できる。

'color'特性は，通常は継承する。背景特性は継承しないが，'background-color'の初期値は 'tansparent'なので，親要素の背景は，デフォルトで，透けて輝くことになる。

5.3.1　'color'

値： <color>
初期値： UA 固有
適用対象： すべての要素
継承： yes
パーセント値： N/A

この特性は，要素のテキストカラー（*前景*の色として参照されることが多い。）を示す。赤を指定するためのいくつかの方法がある。

```
EM { color: red }                  /* natural language */
EM { color: rgb(255,0,0) }         /* RGB range 0-255     */
```

可能なカラーの値の記述に関しては，**6.3**を参照すること。

5.3.2　'background-color'

値： <color> | transparent
初期値： transparent
適用対象： すべての要素
継承： no
パーセント値： N/A

この特性は，要素の背景の色を設定する。

```
H1 { background-color: #F00 }
```

5.3.3　'background-image'

値： <url> | none
初期値： none
適用対象： すべての要素
継承： no
パーセント値： N/A

この特性は，要素の背景画像を設定する。背景画像を設定するときには，その画像が使用不可能なときに使用する背景色も設定するほうがよい。画像が使用可能なときには，背景色の上に画像を重ねる。

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

5.3.4　'background-repeat'

値： repeat | repeat-x | repeat-y | no-repeat
初期値： repeat

適用対象： すべての要素

継承： no

パーセント値： N/A

背景画像を指定する場合は，'background-repeat'の値が，その画像を繰り返す方法又は その画像を繰り返すかどうかを決定する。

'repeat'の値は，画像が左右方向及び上下方向に繰り返されることを示す。'repeat-x'（又は'repeat-y'）の値は，画像を左右方向（又は上下方向）に繰り返し，片側から他の片側へと続く画像の一つの帯を生成する。'no-repeat'の値を使うと，画像は繰り返さない。

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

この例では，画像は上下方向にだけ繰り返す。

5.3.5 'background-attachment'

値： scroll | fixed

初期値： scroll

適用対象： すべての要素

継承： no

パーセント値： N/A

背景画像を指定する場合，'background-attachment'の値で，その画像がキャンバスに関して固定しているか，又は内容に沿ってスクロールするかを決定する。

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

CSS1 コア： UA は，'fixed'を'scroll'として扱ってもよい。しかし，少なくとも HTML 要素及び BODY 要素に関しては，'fixed'を正しく解釈することが望ましい。これは，'fixed'をサポートするこれらブラウザに対してだけ，作成者が画像を提供する方法がないことによる（**7.**を参照）。

5.3.6 'background-position'

値： [<percentage> | <length>]{1,2} | [top | center | bottom] || [left | center | right]

初期値： 0% 0%

適用対象： ブロックレベル要素及び置換要素

継承： no

パーセント値： 要素それ自体のサイズを参照

背景画像を指定した場合，'background-position'の値がその初期位置を指定する。

'0% 0%'の値の対で，画像の上左隅を，要素の内容を囲むボックス（つまり，パディング，境界 又は余白を囲むボックスではない。）の上左隅に置く。'100% 100%'の値の対で，画像の下右隅を要素の下右隅に置く。'14% 84%'の値の対で，画像の右 14%下 84%の点を，要素の右 14%下 84%の点に置く。

'2cm 2cm'の値の対で，画像の上左隅を，要素の上左隅から，2cm 右及び 2cm 下に置く。

一つのパーセントの値又は一つの長さの値だけを与えた場合，左右方向の位置だけを設定し，上下方向の位置は 50%とする。二つの値を与えた場合には，左右方向の位置を最初とする。長さの値及びパーセントの値の結合を許す。例えば，'50% 2cm'。負の位置を許す。

キーワードの値を使用し，背景画像の位置を指定することもできる。キーワードは，パーセントの値又は長さの値と結合することはできない。キーワードの可能な結合及びその解釈は，次による。

a) 'top left'及び'left top'の両方は，'0% 0%'と同じことを意味する。

b) 'top，top center'及び'center top'は，'50% 0%'と同じことを意味する。

c) 'right top'及び'top right'は，'100% 0%'と同じことを意味する。

d) 'left，left center'及び'center left'は，'0% 50%'と同じことを意味する。

e) 'center'及び'center center'は，'50% 50%'と同じことを意味する。

f) 'right'，'right center'及び'center right'は，'100% 50%'と同じことを意味する。

g) 'bottom left'及び'left bottom'は，'0% 100%'と同じことを意味する。

h) 'bottom'，'bottom center'及び'center bottom'は，'50% 100%'と同じことを意味する。

i) 'bottom right'及び'right bottom'は，'100% 100%'と同じことを意味する。

次に例を示す。

```
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% */
```

背景画像をキャンバスに関して固定した場合（**5.3.5** 'background-attachment'特性を参照），要素の代わりに，その画像が，キャンバスに対して相対的に置かれる。次に例を示す。

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

この例では，画像は，キャンバスの下右隅に位置する。

5.3.7 'background'

値： <background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>

初期値： 短縮形特性に対しては定義しない

適用対象： すべての要素

継承： no

パーセント値： <background-position>に関して許される

'background'特性は，スタイルシート内の同じ位置で，個々の背景特性（つまり，'background-color'，'background-image'，'background-repeat'，'background-attachment'及び'background-position'）を設定するための短縮形特性とする。

'background'特性に関する可能な値は，個々の特性に関するすべての可能な値の集合とする。

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

'backgroud'特性は，常に，個々の背景特性すべてを設定する。この例の最初の規則では，'backgound-color'に対する値だけが与えられ，他の個々の特性は，それらの初期値を設定する。2 番目の規則では，すべての個々の特性を指定している。

5.4 テキスト特性

5.4.1 'word-spacing'

値: normal | <length>

初期値: normal

適用対象: すべての要素

継承: yes

パーセント値: N/A

length 単位は，語間のデフォルトスペースへの追加を示す。負の値も可能であるが，実装固有の制限があってもよい。UA は，正確なスペース空けアルゴリズムを選択してもよい。語間スペースは，('text-align'特性の値である）justification（均等割り）によって影響を受けることもある。

H1 { word-spacing: 1em }

この場合，'H1'要素の中の各語の語間スペースは，'0.4em'だけ増加する。

CSS1 コア: UA は，どんな'word-spacing'の値も'normal'と解釈してよい（**7.**を参照）。

5.4.2 'letter-spacing'

値: normal | <length>

初期値: normal

適用対象: すべての要素

継承: yes

パーセント値: N/A

length 単位は，文字間のデフォルトスペースへの追加を示す。負の値も可能であるが，実装固有の制限があってもよい。UA は, 正確なスペース空けアルゴリズムを選択してもよい。字間スペースは，('text-align'特性の値である）justification によって影響を受けることもある。

BLOCKQUOTE { letter-spacing: 0.1em }

この場合，'BLOCKQUOTE'要素の中の各文字の字間スペースは，'0.1em'だけ増加する。

値が'normal'のとき，UA は，テキストの均等割りを行うために字間スペースを変えてもよい。'letter-spacing'が明示的に<length>の値に設定される場合には，これが起きることはない。

BLOCKQUOTE { letter-spacing: 0 }

BLOCKQUOTE { letter-spacing: 0cm }

結果としての 2 字間のスペースがデフォルトスペースと同じでない場合，UA はリガチャを使用してはならない。

CSS1 コア: UA は，どんな'letter-spacing'の値も'normal'と解釈してよい（**7.**を参照）。

5.4.3 'text-decoration'

値: none | [underline || overline || line-through || blink]

初期値: none

適用対象: すべての要素

継承: なし（次の詳細記述を参照）

パーセント値: N/A

この特性は，要素のテキストに加える修飾を記述する。要素にテキストがない場合（HTML の'IMG'要素など），又は要素が空要素の場合（<EM></EM&gt など），この特性は効果をもたない。値'blink'は，テキストを点滅させる。

テキスト修飾に必要なカラーは，'color'特性の値からもってくる。

この特性は，継承されないが，要素はその親と一致していなければならない。例えば，ある要素に下線が引かれていると，その線は子の要素にも及ぶ。下線のカラーは，子孫の要素が別の'color'値をもっていたとしても，同じとなる。

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

この例は，すべてのリンク（つまり，'HREF'属性をもつすべての'A'要素）のテキストに下線を引く。

UA は，キーワード'blink'を認識できなければならないが，点滅効果をサポートする必要はない。

5.4.4 'vertical-align'

値: baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage>

初期値: baseline

適用対象: 行内要素

継承: no

パーセント値: 要素そのものの'line-height'に対する値

この特性は，要素の上下方向の位置決めに影響を与える。次のキーワードの集合は，親要素に対して相対とする。

a) **'baseline'** 要素のベースライン（要素がベースラインをもたない場合には，下部）を，親のベースラインにそろ（揃）える。

b) **'middle'** 要素（通常は，画像）の上下方向の中点を，親のベースラインに x-height の半分を加えた位置にそろ（揃）える。

c) **'sub'** 要素を下付き添字にする。

d) **'super'** 要素を上付き添字にする。

e) **'text-top'** 要素の上部を親要素のフォントの上部にそろ（揃）える。

f) **'text-bottom'** 要素の下部を親要素のフォントの下部にそろ（揃）える。

もう一つの特性集合は，その要素を含むフォーマット済みの行に対して相対とする。

a) **'top'** 要素の上部を，行の中の高さが最も高い要素にそろ（揃）える。

b) **'bottom'** 要素の下部を，行の中の高さが最も低い要素にそろ（揃）える。

'top'及び'bottom'のそろ（揃）えを使用する場合，要素の依存関係がループを形成して，解決できない状態が起こる可能性がある。

パーセント値は，要素そのその'line-height'特性の値に対するものとする。それは，指定された量だけ親のベースラインよりも上に要素のベースライン（ベースラインがない場合には，下部）を上げる。負の値も可能とする。例えば，-100%の値は，次の行のベースラインがくるはずのところに要素のベースラインが帰着するまで，要素を下げる。これによって，ベースラインをもたない要素（字の代わりに用いる画像など）の上下方向の位置を正確に制御可能にする。

CSS の将来の版では，この特性の値として<length>を使用可能にすることが期待される。

5.4.5 'text-transform'

値: capitalize | uppercase | lowercase | none

初期値: none

適用対象: すべての要素

継承: yes

パーセント値: N/A

a) **'capitalize'** 各語の最初の文字を大文字にする。

b) **'uppercase'** 要素のすべての字を大文字にする。

c) **'lowercase'** 要素のすべての字を小文字にする。

d) **'none'** 継承された値を無効にする。

それぞれの場合の実際の変換は，自然言語依存とする。要素の言語を見つける方法については，**8.**の[4]を参照されたい。

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

この例は，'H1'要素を大文字のテキストにする。

CSS1 コア: UA は，Latin-1 レパートリには含まれない文字について，及び Unicode（**8.**の[8]を参照）の大文字小文字変換表によって指定された変換とは異なる変換に関する言語の要素については，'text-transform'を無視してよい（つまり，'none'として扱う。）。

5.4.6 'text-align'

値: left | right | center | justify

初期値: UA specific

適用対象: ブロックレベル要素

継承: yes

パーセント値: N/A

この特性は，要素内でのテキストのそろ（揃）え方を記述する。実際に用いる調整アルゴリズムは，UA 及び自然言語に依存する。

例

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

'text-align'は継承されるので，'CLASS=center'が指定された'DIV'要素の中のすべてのブロックレベル要素のすべての行ボックスは，中央そろ（揃）えとなる。そろ（揃）えは，キャンバスではなく要素の幅に対しての相対指定であることに注意されたい。'justify'がサポートされていない場合，UA が代替を提供する。西洋言語では，これは'left'となることが多い。

CSS1 コア: UA は，要素のデフォルトの表記方向が右向きであるか左向きであるかに応じて，それぞれ'left'又は'right'として'justify'を扱ってよい。

5.4.7 'text-indent'

値: <length> | <percentage>

初期値: 0

適用対象: ブロックレベル要素

継承: yes

パーセント値: 親要素の幅に対する値

この特性は，フォーマット済みの最初の行の前に現れる字下げを指定する。'text-indent'の値は負であってよいが，実装固有の制限があることもある。字下げは，他のもの（HTML の'BR'など）で分割された要素の中央には挿入されない。

例

```
P { text-indent: 3em }
```

5.4.8 'line-height'

値: normal | <number> | <length> | <percentage>

初期値: normal

適用対象: すべての要素

継承: yes

パーセント値: 要素そのもののフォントサイズに対する値

この特性は，二つの隣接する行のベースラインの間の距離を設定する。

数値が指定された場合，現在の要素のフォントサイズにその数値を乗算したものが，行の高さとなる。これは，継承の仕方において，パーセント値とは異なる。つまり数値が指定されると，子供の要素は，（パーセント値及び他の単位の場合と同様の）結果の値ではなく，その係数そのものを継承する。

負の値は使用できない。

次の例の三つの規則は，結果として同じ行の高さとなる。

```
DIV { line-height: 1.2; font-size: 10pt }         /* number */
DIV { line-height: 1.2em; font-size: 10pt }      /* length */
DIV { line-height: 120%; font-size: 10pt }       /* percentage */
```

値'normal'は，'line-height'を要素のフォントに関して適切な値に設定する。UA は，'normal'値を 1.0 から 1.2 の範囲の数値にすることが，提案される。

'line-height'が, ブロックレベル要素にどのように影響を与えるかの記述については, **4.4**を参照されたい。

5.5 ボックス特性 ボックス特性は，要素を表現するボックスのサイズ，周囲及び位置を設定する。ボックス特性の使用法の例については，フォーマット化モデル(**4.**)を参照されたい。

余白特性は，要素の余白を設定する。'margin'特性は，4 辺すべての余白を設定する。他の余白特性は，対応する辺の設定だけを行う。

パディング特性は，境界と内容（テキスト又は画像）との間にどれほどのスペースを入れるかを記述する。'padding'特性は，4 辺すべてのパディングを設定する。他のパディング特性は，対応する辺の設定だけを行う。

境界特性は，要素の境界を設定する。各要素は，四つ（各辺に一つ）の境界をもち，それは，幅，カラー，スタイルによって定義される。

'width'特性及び'height'特性は，ボックスのサイズを設定する。'float'特性及び'clear'特性は，要素の位置を変えることができる。

5.5.1 'margin-top'

値: <length> | <percentage> | auto

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の上部余白を設定する。

```
H1 { margin-top: 2em }
```

負の値も指定可能であるが，実装固有の制限があることがある。

5.5.2 'margin-right'

値: <length> | <percentage> | auto

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の右余白を設定する。

```
H1 { margin-right: 12.3% }
```

負の値も指定可能であるが，実装固有の制限があることがある。

5.5.3 'margin-bottom'

値: <length> | <percentage> | auto

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の下部余白を設定する。

```
H1 { margin-bottom: 3px }
```

負の値も指定可能であるが，実装固有の制限があることがある。

5.5.4 'margin-left'

値: <length> | <percentage> | auto

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の左余白を設定する。

```
H1 { margin-left: 2em }
```

負の値も指定可能であるが，実装固有の制限があることがある。

5.5.5 'margin'

値: [<length> | <percentage> | auto]{1,4}

初期値: 簡略表記特性については定義されない

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

'margin'特性は，'margin-top', 'margin-right', 'margin-bottom'及び'margin-left'を，スタイルシートの同じ場所に設定するための簡略表記特性とする。

四つの length 値が指定されたると，それらは，それぞれ上部，右，下部及び左に適用される。値が一つだけの場合，それがすべての辺に適用される。二つ又は三つの場合，指定のない値には，反対側の辺の値を用いる。

```
BODY { margin: 2em } /* all margins set to 2em */
BODY { margin: 1em 2em } /* top & bottom = 1em, right & left = 2em */
BODY { margin: 1em 2em 3em } /* top=1em, right=2em, bottom=3em, left=2em */
```

この例の最後の規則は，次の例と同じとする。

```
BODY {
  margin-top: 1em;
  margin-right: 2em;
  margin-bottom: 3em;
  margin-left: 2em;              /* copied from opposite side (right) */
}
```

負の余白値も指定可能であるが，実装固有の制限があることがある。

5.5.6 'padding-top'

値: <length> | <percentage>

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の上部パディングを設定する。

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

パディングの値は，負に指定できない。

5.5.7 'padding-right'

値: <length> | <percentage>

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の右パディングを設定する。

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

パディングの値は，負に指定できない。

5.5.8 'padding-bottom'

値: <length> | <percentage>

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の下部パディングを設定する。

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

パディングの値は，負に指定できない。

5.5.9 'padding-left'

値: <length> | <percentage>

初期値: 0

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

この特性は，要素の左パディングを設定する。

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

パディングの値は，負に指定できない。

5.5.10 'padding'

値: [<length> | <percentage>]{1,4}

初期値: 簡略表記特性については定義されない。

適用対象: すべての要素

継承: no

パーセント値: 直近のブロックレベルの先祖の幅に対する値

'padding'特性は，'padding-top', 'padding-right', 'padding-bottom'及び'padding-left'を，スタイルシートの同じ場所に設定するための簡略表記特性とする。

四つの値が指定されると，それらは，それぞれ上部，右，下部及び左に適用される。値が一つだけの場合，それがすべての辺に適用される。二つ又は三つの場合，指定のない値には反対側の辺の値を用いる。

パディング領域の面は，'background'特性で設定される。

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

この例は，'1em'のパディング（'padding-top'及び'padding-bottom'）を上下に設定し，'2em'のパディング（'padding-right'及び'padding-left'）を左右に設定する。'em'単位は，要素のフォントサイズに対して相対とする。'1em'は，使用しているフォントのサイズに等しい。

パディングの値は，負に指定できない。

5.5.11 'border-top-width'

値: thin | medium | thick | <length>

初期値: 'medium'

適用対象: すべての要素

継承: no

パーセント値: N/A

この特性は，要素の上部境界を設定する。キーワード値の幅は UA 依存とするが，'thin' <= 'medium'<= 'thick'の関係は保つ。

キーワードの幅は，文書を通して一定とする。

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

この例では，'H1'要素及び'P'要素は，フォントサイズに関係なく，同じ境界の幅をもつ。相対的な幅を達成するために，'em'単位を使用できる。

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

境界の幅は，負に指定できない。

5.5.12 'border-right-width'

値: thin | medium | thick | <length>

初期値: 'medium'

適用対象: すべての要素

継承: no

パーセント値: N/A

この特性は，要素の右境界の幅を設定する。それ以外は，'border-top-width'と同じとする。

5.5.13 'border-bottom-width'

値: thin | medium | thick | <length>

初期値: 'medium'

適用対象: すべての要素

継承: no

パーセント値: N/A

この特性は，要素の下部境界の幅を設定する。それ以外は，'border-top-width'と同じとする。

5.5.14 'border-left-width'

値: thin | medium | thick | <length>

初期値: 'medium'

適用対象: すべての要素

継承: no

パーセント値: N/A

この特性は，要素の左境界の幅を設定する。それ以外は，'border-top-width'と同じとする。

5.5.15 'border-width'

値: [thin | medium | thick | <length>]{1,4}

初期値: not defined for shorthand properties

適用対象: すべての要素

継承: no

パーセント値: N/A

この特性は，'border-top-width', 'border-right-width', 'border-bottom-width'及び'border-left-width'を，スタイルシートの同じ場所に設定するための簡略表記特性とする。

一つから四つまでの値を，次の解釈で指定できる。

a) 一つの値: 四つのすべての境界がこの値に設定される。

b) 二つの値: 上部及び下部の境界の幅が最初の値に設定され，右及び左の境界が 2 番目の値に設定される。

c) 三つの値: 上部境界が最初の値に設定され，右及び左の境界が 2 番目の値に設定され，下部境界は 3 番目の値に設定される。

d) 四つの値: 上部，右，下部及び左の境界が，それぞれの値に設定される。

次の例では，コメントが上部，右，下部及び左の境界の結果としての幅を示す。

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

境界の幅は，負の値に指定できない。

5.5.16 'border-color'

値: <color>{1,4}
初期値: 特性 color の値
適用対象: すべての要素
継承: no
パーセント値: N/A

'border-color'特性は，四つの境界のカラーを設定する。'border-color'は，一つから四つまでの値をもつことができ，その値は，前述の'border-width'と同様に，異なる辺に対して設定される。

カラーの値が指定されない場合，要素そのものの'color'特性の値が代用される。

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

この例では，境界は黒い直線となる。

5.5.17 'border-style'

値: none | dotted | dashed | solid | double | groove | ridge | inset | outset
初期値: none
適用対象: すべての要素
継承: no
パーセント値: N/A

'border-style'特性は，四つの境界のスタイルを設定する。この特性は，一つから四つまでの値をもつことができ，その値は，前述の'border-width'と同様に，異なる辺に対して設定される。

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

この例では，左右方向の境界が'solid'（直線）となり，上下方向の境界が'dotted'（点線）となる。

境界のスタイルの初期値が'none'であるので，境界のスタイルが設定されないと，境界は見えない。

境界のスタイルの意味は，次のとおりとする。

a) none （'border-width'値にかかわらず）境界を引かない。
b) dotted 要素の背景の上部に，境界を点線で描く。
c) dashed 要素の背景の上部に，境界を破線で描く。
d) solid 境界を直線とする。
e) double 要素の背景の上部に，境界を二重線で描く。二つの単線と間のスペースとの合計が，<border-width>の値に等しい。
f) groove 3D groove（立体的な溝）を，<color>値に基くカラーで描く。
g) ridge 3D ridge（立体的な峰）を，<color>値に基くカラーで描く。
h) inset 3D inset（立体的な窪み）を，<color>値に基くカラーで描く。
i) outset 3D outset（立体的な出っ張り）を，<color>値に基くカラーで描く。

CSS1 コア: UA は，'dotted', 'dashed', 'double', 'groove', 'ridge', 'inset'及び'outset'をすべて'solid'と解釈してよい。

5.5.18 'border-top'

値: <border-top-width> || <border-style> || <color>
初期値: 簡略表記特性については定義されない
適用対象: すべての要素
継承: no
パーセント値: N/A

これは，要素の上部境界の幅，スタイル及びカラーを設定するための簡略表記特性とする。

H1 { border-bottom: thick solid red }

この規則は，H1 要素の下の境界の幅，スタイル及びカラーを設定する。省略された値は，初期値に設定される。

H1 { border-bottom: thick solid }

この例では，カラー値が省略されているので，境界のカラーは，要素そのものの'color'値と同じになる。

'border-style'特性は，四つの値まで受けとるが，この特性は一つのスタイル値だけを受けとることに留意されたい。

5.5.19 'border-right'

値: <border-right-width> || <border-style> || <color>
初期値: 簡略表記特性については定義されない
適用対象: すべての要素
継承: no
パーセント値: N/A

これは，要素の右境界の幅，スタイル及びカラーを設定するための簡略表記特性とする。それ以外は，'border-top'と同じとする。

5.5.20 'border-bottom'

値: <border-bottom-width> || <border-style> || <color>
初期値: 簡略表記特性については定義されない
適用対象: すべての要素
継承: no
パーセント値: N/A

これは，要素の下部境界の幅，スタイル及びカラーを設定するための簡略表記特性とする。それ以外は，'border-top'と同じとする。

5.5.21 'border-left'

値: <border-left-width> || <border-style> || <color>
初期値: not defined for shorthand properties
適用対象: すべての要素
継承: no
パーセント値: N/A

これは，要素の左境界の幅，スタイル及びカラーを設定するための簡略表記特性とする。それ以外は，'border-top'と同じとする。

5.5.22 'border'

値: <border-width> || <border-style> || <color>
初期値: 簡略表記特性については定義されない

適用対象: すべての要素

継承: no

パーセント値: N/A

'border'特性は，要素の四つの境界すべてに同じ幅，同じスタイル及び同じカラーを設定するための簡略表記特性とする。例えば，次の最初の規則は，その後に示す四つの規則の集合と等価となる。

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

簡略表記特性'margin'及び'padding'とは異なり，'border'特性は，四つの境界に異なる値を設定することはできない。それを行うためには，一つ以上の他の境界特性を使用する必要がある。

特性は, ある程度重なった機能をもつので, 規則が指定される順序が重要になる。次の例を考えてみる。

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

この例では，左境界のカラーは黒となる。これは，幅，スタイル及びカラーを設定する'border-left'による。'border-left'特性では，カラー値が指定されないので，'color'特性の値が使用される。'color'特性が'border-left'特性の後に設定されることとは，関係ない。

'border-width'特性は，四つの length 値までを受けとるが，この特性は，一つだけを受けとることに注意されたい。

5.5.23 'width'

値: <length> | <percentage> | auto

初期値: auto

適用対象: ブロックレベル要素及び置換要素

継承: no

パーセント値: 親要素の幅に対する値

この特性は, テキスト要素に対して適用できるが, 画像のような置換要素と共に最も有効に用いられる。幅は，必要であれば画像のスケーリング（変倍）によって強制される。スケーリングに際しては，'height'特性が'auto'であると，画像の縦横比が保存される。

例

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

置換要素の'width'及び'height'が両方とも'auto'であれば，これらの特性は，要素の基本寸法に設定される。

負の値は，指定できない。

この特性，余白及びパディングの関係の記述については，フォーマット化モデル(**4.**)を参照されたい。

5.5.24 'height'

値: <length> | auto
初期値: auto
適用対象: ブロックレベル要素及び置換要素
継承: no
パーセント値: N/A

この特性は, テキスト要素に対して適用できるが, 画像のような置換要素と共に最も有効に用いられる。幅は，必要であれば画像のスケーリング（変倍）によって強制される。スケーリングに際しては，'width'特性が'auto'であると，画像の縦横比が保存される。

例

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

置換要素の'width'及び'height'が両方とも'auto'であれば, これらの特性は, 要素の基本寸法に設定される。

テキスト要素に適用される場合，その高さは，例えばスクロールバーを用いて強制される。

負の値は，指定できない。

CSS1 コア: 要素が置換要素でない場合，UA は，'height'特性を無視してよい（つまり，それを'auto'として扱う。）。

5.5.25 'float'

値: left | right | none
初期値: none
適用対象: すべての要素
継承: no
パーセント値: N/A

値が'none'の場合，要素は，それがテキストの中に現れる場所に表示される。値が'left'('right')の場合，要素は左（右）に移動し，テキストが要素の右（左）側で折り返す。値が'left'又は'right'の場合，要素はブロックレベルとして扱われる（つまり，'display'特性は無視される。）。全規定については，**4.1.4** を参照されたい。

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

この例は，親要素の左側に沿って，'CLASS=icon'をもつすべての IMG 要素を配置する。

この特性は，行内画像と共に最も頻繁に用いられるが，テキスト要素にも適用できる。

5.5.26 'clear'

値: none | left | right | both
初期値: none
適用対象: すべての要素
継承: no
パーセント値: N/A

この特性は，要素の横に浮動要素を置いてよいかどうかを指定する。もっと詳しく言うと，この特性の値は，浮動要素が許されない側を列挙する。'left'に設定された'clear'の場合，要素は左側のすべての浮動要素の下に移動する。'None'に設定された'clear'の場合, 浮動要素がすべての側で許可される。例を次に示す。

H1 { clear: left }

5.6　分類特性　これらの特性は，固有の可視化パラメータを設定し，さらに要素を分類する。

list-style 特性は，リスト項目（つまり，'list-item'の'display'値をもつ要素）がフォーマット化される方法を記述する。list-style 特性は，どんな要素にも設定でき，通常はトリーの下方向に継承される。しかし，'list-item'の'display'値をもつ要素だけに効果をもつ。HTML では通常，LI 要素に対して効果をもつ。

5.6.1　'display'

値: block | inline | list-item | none

初期値: block

適用対象: すべての要素

継承: no

パーセント値: N/A

この特性は，キャンバス（印刷ページ，コンピュータ画面など）上に，要素が表示されるかどうか，どのように表示されるかを記述する。

'block'の'display'値をもつ要素は，新規のボックスを開く。ボックスは，CSS のフォーマット化モデルに従い, 隣接するボックスに相対的に位置付けされる。通常, 'H1'及び'P'などの要素が型'block'をもつ。'list-item'の値は，list-item マーカが追加されることを除いて'block'に類似している。HTML では，'LI'が通常この値をもつ。

'inline'の'display'値をもつ要素は, その前の内容と同じ行の新規の行内ボックスとなる。そのボックスは，内容のフォーマット済みサイズに従って，大きさが決る。内容がテキストであれば，それが数行に及ぶこともある。余白，境界及びパディングの特性は，'inline'要素に対して適用されるが，改行に際して効果はない。

値'none'は，子要素及び周囲のボックスを含んで，要素の表示を消す。

P { display: block }

EM { display: inline }

LI { display: list-item }

IMG { display: none }

最後の規則は，画像の表示を消す。

'display'の初期値は'block'とするが，通常，UA は，HTML 規定（**8.**の[2]を参照）において提案されている要素の可視化に従って，すべての HTML 要素に関するデフォルト値をもつ。

CSS1 コア: UA は，'display'を無視し，UA のデフォルト値だけを使用してよい（**7.**を参照）。

5.6.2　'white-space'

値: normal | pre | nowrap

初期値: normal

適用対象: すべての要素

継承: yes

パーセント値: N/A

この特性は，要素内の空白が扱われる方法を宣言する。つまり，'normal'（空白はつぶれる），'pre'（HTML における'PRE'要素に似た振る舞いをする），又は'nowrap'（BR 要素だけによって折り返す）とする。

PRE { white-space: pre }

P　{ white-space: normal }

'white-space'の初期値は'normal'とするが，通常，UA は，HTML 規定（**8.**の[2]を参照）において提案されている要素の可視化に従って，すべての HTML 要素に関するデフォルト値をもつ。

CSS1 コア: UA は，文書作成者及び読者のスタイルシートにおける'white-space'特性を無視し，その代わりに UA のデフォルト値を使用してよい（**7.**を参照）。

5.6.3 'list-style-type'

値: disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none

初期値: disc

適用対象: 'list-item'の'display'値をもつ要素

継承: yes

パーセント値: N/A

この特性は，'list-style-image'が'none'である場合，又は URL によって指定された画像が表示できない場合に，'list-item'マーカの出現を決定するために用いる。

```
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'

値: <url> | none

初期値: none

適用対象: 'list-item'の'display'値をもつ要素

継承: yes

パーセント値: N/A

この特性は，list-item マーカとして使いる画像を設定する。画像が利用可能な場合，それは，マーカ集合を'list-style-type'マーカで置き換える。

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

5.6.5 'list-style-position'

値: inside | outside

初期値: outside

適用対象: 'list-item'の'display'値をもつ要素

継承: yes

パーセント値: N/A

'list-style-position'の値は，内容に関して list-item マーカがどのように描かれるかを決定する。フォーマット化の例については，**4.1.3**を参照されたい。

5.6.6 'list-style'

値: [disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none] || [inside | outside] || [<url> | none]

初期値: 簡略表記特性については定義されない

適用対象: 'list-item'の'display'値をもつ要素

継承: yes

パーセント値: N/A

'list-style'特性は，三つの特性'list-style-type', 'list-style-image'及び'list-style-position'を，スタイルシートの同

じ場所に設定するための簡略表記記法とする。

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

'LI'要素に直接'list-style'を設定すると，予期しない結果が生じ得る。次の場合を考えてみる。

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

この例では，スタイルシートの最初の規則に関して，（段階順序で指定された）固有性が高いので，それが, すべての'LI'要素に関する 2 番目の規則を上書きし, 'lower-alpha'のリストスタイルのだけが使用される。したがって，リスト型要素だけに'list-style'を設定することが推奨される。

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

この例では，継承が，'OL'要素及び'UL'要素から'LI'要素に対して'list-style'の値を転送する。

URL の値は，他のどんな値とも組み合わせることができる。

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

この例では，画像が入手できない場合に'disc'が使用される。

6. 単位

6.1 長さの単位 長さの値のフォーマットは，オプションの符号文字（'+'又は'-'で，'+'がデフォルト。）の直後に数字（小数点付き又は小数点なし）が続き，さらにその直後に単位識別子（2 字の短縮形）が続く。'0'の数字の後の単位識別子はオプションとする。

幾つかの特性は，負の長さの単位を許す。しかしこれは，フォーマット化モデルを複雑にすることがあり，実装固有の制限が存在するかもしれない。負の長さの値をサポートできない場合は，サポート可能な最も近い値に丸める。

長さの単位には，相対的及び絶対的の 2 種類がある。相対単位は，他の長さ特性に相対的な長さを指定する。相対単位を用いたスタイルシートでは，あるメディアから他のメディアへの（例えば，コンピュータディスプレイからレーザプリンタへの）スケール化が可能となる。パーセント単位（**6.2** に示す。）及びキーワード値（例えば，'x-large'）も，同様の利点を提供する。

次の相対単位をサポートする。

```
H1 { margin: 0.5em }          /* ems, the height of the element's font */
```

```
H1 { margin: 1ex }              /* x-height, ~ the height of the letter 'x' */
P   { font-size: 12px }       /* pixels, relative to canvas */
```

相対単位'em'及び'ex'は，要素それ自体のフォントサイズに対して相対的とする。CSS1 におけるこの規則の例外は'font-style'特性であって，'em'及び'ex'の値は，親要素のフォントサイズを参照する。

画素単位は，この例外規則を用い，通常はコンピュータディスプレイとなるキャンバスの解像度に対して相対的とする。出力装置の画素密度が典型的なコンピュータディスプレイの密度と大きく違う場合，UA は，画素値を再スケール化するのが望ましい。推奨*参照画素*は，96dpi の画素密度をもつ読者の腕の長さ離れた装置上の一画素の視角とする。通常の腕の長さ 28 インチに対して，視角は約 0.0213 度となる。

子要素は，相対値ではなく，計算した値を継承する。

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

この例では，要素'H1'の'text-indent'値は 36pt であって，45pt ではない。

絶対長さ単位は，出力媒体の物理特性が分かっている場合にだけ有効となる。次の絶対単位をサポートする。

```
H1 { margin: 0.5in }          /* inches, 1in = 2.54cm */
H2 { line-height: 3cm }     /* centimeters */
H3 { word-spacing: 4mm }    /* millimeters */
H4 { font-size: 12pt }       /* points, 1pt = 1/72 in */
H4 { font-size: 1pc }         /* picas, 1pc = 12pt */
```

指定された長さをサポートできない場合，UA は近似値を試すのがよい。すべての CSS1 特性について，それ以降の計算及び継承は，その近似値を基にすることが望ましい。

6.2　パーセント単位　パーセント値のフォーマットは，オプションの符号文字（'+'又は'-'で，'+'がデフォルト。）の直後に数字（小数点付き又は小数点なし）が続き，さらにその直後に'%'が続く。

パーセント値は，長さ単位など他の値に対して常に相対的とする。パーセント単位を許す各特性は，パーセント値が何を参照するかも定義する。これを，要素それ自体のフォントサイズとすることが多い。

```
P { line-height: 120% }     /* 120% of the element's 'font-size' */
```

すべての継承された CSS1 特性では，値がパーセントとして指定された場合，子要素は結果としての値を継承し，パーセント値を継承しない。

6.3　カラー単位　カラーは，キーワード又は数値的な RGB 規定とする。

カラー名のキーワードの与えられたリストは，aqua（淡緑青），black（黒），blue（青），fuchsia（赤紫色），gray（グレイ），green（緑），lime（ライム），maroon（えび茶），navy（ネービー），olive（オリーブ），purple（紫），red（赤），silver（銀），teal（青緑），white（白）及び yellow（黄）とする。これらの 16 色は，Windows VGA パレットから取られたが，その RGB 値は，この規定では定義しない。

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

RGB カラーモデルは，数値カラー規定を用いる。次の例は，すべて同じ色を指定する。

```
EM { color: #f00 }                  /* #rgb */
EM { color: #ff0000 }               /* #rrggbb */
EM { color: rgb(255,0,0) }          /* integer range 0 - 255 */
EM { color: rgb(100%, 0%, 0%) } /* float range 0.0% - 100.0% */
```

16 進記法での RGB のフォーマットは，'#'の直後に 3 個又は 6 個の 16 進の文字が続く。3 桁の RGB 記法(#rgb)は，0 を追加することではなく数字を複製することで，6 桁の数字の型式(#rrggbb)に変換する。例えば，#fb0 は，#ffbb00 に拡張する。これによって，white(#ffffff)は，確実に短い記法(#fff)で指定でき，ディスプレイの色の深さの依存性を取り除くことができる。

関数的記法の RGB 値のフォーマットは，'rgb（'の後にカンマで区切られた三つの数値（0～255 の範囲の三つの整数値，又は 0.0%～100.0%の範囲の三つのパーセント値）のリストが続き，さらに')'が続く。空白文字を，数値の前後に入れてもよい。

範囲外の数値は，（範囲内の最も近い値に）丸めるのがよい。そこで，次の三つの規則は等価となる。

```
EM { color: rgb(255,0,0) }          /* integer range 0 - 255 */
EM { color: rgb(300,0,0) }          /* clipped to 255 */
EM { color: rgb(110%, 0%, 0%) }   /* clipped to 100% */
```

RGB カラーは，sRGB 色空間（**8.**の[9]を参照）で指定する。UA は，これらの色を表現する際に正確さを犠牲にすることになるかもしれない。しかし，sRGB を用いることで，国際規格（**8.**の[10]を参照）と関連可能な，その色が何かという，あいまい性のない客観的に測定可能な定義ができる。

UA は，ガンマ補正の実行に対して，色表示の努力を制限してもよい。sRGB は，指定された視覚条件下で，表示ガンマを 2.2 と規定する。UA は，出力装置の"自然な"表示ガンマとの組合せで，実効表示ガンマ 2.2 を生成するものとして CSS 内で与えられた色を調整する。この詳細は，**附属書 D**に示す。CSS で規定する色だけが，影響することに注意。例えば，画像は，それ自体の色情報をもち運ぶことを期待する。

6.4　URL　URL(Uniform Resource Locator)は，関数的記法で識別する。

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

URL 値のフォーマットは，'url('の後にオプションの空白が続き，オプションの一重引用符(')又は二重引用符(")が続き，URL 自体（**8.**の[11]で定義）が続き，オプションの一重引用符(')又は二重引用符(")が続き，オプションの空白が続き，さらに')'が続く。引用符文字は，URL 自体の部分ではないが，対になっていなければならない。

URL に出現する，括弧，コンマ，空白文字，一重引用符(')及び二重引用符(")は，¥記号で，'¥(',¥)',¥,'などとしてエスケープしなければならない。

部分的な URL は，文書に対してではなく，スタイルシートのソースに対して相対的に解釈する。

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

7.　CSS1 適合性　文書を表示するために CSS1 を用いる UA は，次を満たす場合に，CSS1 規定に適合する。

a)　すべての参照スタイルシートの取込みを行い，この規格に従って構文解析する。

b)　段階順序に従って宣言をソートする。

c)　表現媒体の制約内（以下に示す。）で，CSS1 の機能を実装する。

CSS1 スタイルシートを出力する UA は，次を満たす場合に，CSS1 規定に適合する。

a)　妥当な CSS1 スタイルシートを出力する。

文書を表示するために CSS1 を用い，*さらに*CSS1 スタイルシートを出力する UA は，これら二つの適合性要件を共に満たす場合に，CSS1 規定に適合する。

UA は，すべての CSS1 機能を実装する必要はない。コア機能を実装することで，CSS1 に適合できる。コア機能は, 明示的に除外された部分を除く CSS1 規定の全体で構成する。この規格では, この部分は"*CSS1 コア:*"として記されており，それに続いてコア機能以外の機能を示す。コア機能から除外された機能の集合を，CSS1 上位機能と呼ぶ。

7.では，CSS1 への適合性だけを定義する。将来，他の水準の CSS が存在するようになった場合には，適合性のために他の機能集合の実装を UA に要求するかもしれない。

表現媒体の制約の例としては，資源の限定（フォント，カラーなど）及び解像度の限定（このために，余白は不正確となるかもしれない。）がある。これらの場合，UA は，スタイルシートの値を近似することが望ましい。異なる利用者インタフェースでは，それ自体の制約があるかもしれない。VR ブラウザは，利用者からの"距離"に基づいて文書を再スケール化するかもしれない。

UA は，読者に，表現に関する付加的な選択を提供してもよい。例えば，UA は，視覚損傷のある読者にオプションを提供したり，ブリンクを不可能とする選択を提供したりしてもよい。

CSS1 は，フォーマット化のすべての様相を規定しているわけではないことに注意すること。例えば，UA は，自由に字間スペース化のアルゴリズムを選択する。

この規格は，UA が次を許容することを推奨するが，要求はしない。

a) 読者が個人スタイルシートを指定すること。

b) 個々のスタイルシートの採用の可否。

適合規則は，機能だけを示すものであって，利用者インタフェースを示さない。

7.1　上位互換の構文解析　この規格は，CSS 水準 1 を定義する。付加機能をもつ上位水準の CSS を将来定義することが期待される。CSS1 だけをサポートする UA が，確実に上位水準の機能を含むスタイルシートを読むことができるために，**7.1** では，CSS 水準 1 で有効でない上位水準の構成に出会った場合，UA は何をするかを定義する。

a) 知らない特性をもつ宣言は無視する。例えば，スタイルシートを次のとおりとする。

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

このとき，UA は，スタイルシートを次のものとして取り扱う。

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

b) 不正な値，*つまり不正な部分をもつ*値は，宣言が存在しなかったものとして取り扱う。

```
IMG { float: left }             /* CSS1 */
IMG { float: left top }     /* "top" is not a value of 'float' */
IMG { background: "red" } /* keywords cannot be quoted in CSS1 */
IMG { border-width: 3 }     /* a unit must be specified for length values */
```

この例では，CSS1 パーサは最初の規則を尊重し，残りは無視する。つまり，スタイルシートは次のとおりであったものとする。

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

将来の CSS 規定に適合する UA は，他の一つ以上の規則を同様に受け入れるかもしれない。

c) 不正な@キーワードは，その後の最初に来る次のセミコロン(;)又は波括弧({...})まで及びこれを含むすべてと共に無視する。例えば，スタイルシート指定が次を読むと仮定する。

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

CSS1 に従うと，'@three-dee'は不正となる。そこで，すべての@規則（3 番目の右波括弧まで及びこれを含む）は，無視する。CSS1 UA は，これをスキップし，実効的にはスタイルシートを縮退して，次のとおりとする。

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

より詳細には，次のとおり。

CSS の任意の版に対して，CSS スタイルシートは，*文*のリストから構成される。文には，*@規則*及び*規則集合*の 2 種類がある。空白（スペース，タブ及び行換え(newline)）が，文の前後に存在してもよい。

CSS スタイルシートは，HTML 文書に埋め込まれることが多いが，より古い UA からスタイルシートを隠すことができるためには，HTML のコメントの中にスタイルシートを置くのが便利となる。HTML のコメントトークン"<!--"及び"-->"が，文の前，後及び中に出現してもよい。その前後に空白があってもよい。

@規則は，*@キーワード*で開始する。@キーワードとは，先頭が'@'で始まる識別子とする（例えば，'@import'，'@page'など。）。識別子は，字，数字，ダッシュ及びエスケープ文字（後で定義する。）から成る。

@規則は，最初に来る次のセミコロン又は次のブロック（すぐ後で定義する。）まで及びこれを含むすべてから構成される。'@import'以外の@キーワードで始まる@規則に出会った CSS1 UA は，@規則全体を無視し，その後，構文解析を続ける。'@import'で始まる@規則であっても，それがスタイルシートの最初にない場合には，つまり，他の規則（たとえそれが無視される規則であっても）の後に出現する場合には，無視される。次に例を示す。

CSS1 パーサが次のスタイルシートに出会ったと仮定する。

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

CSS1 に従うと，2 番目の@import は不正となる。CSS1 パーサは，その@規則全体をスキップし，実効的には，スタイルシートを次のとおりに縮退する。

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

*ブロック*は，左波括弧({)で始まり，対応する右波括弧(})で終わる。その間に，任意の文字が存在してもよい。ただし，丸括弧(())，角括弧([])及び波括弧({})は，常に対応する対で出現し，入れ子になっていてもよい。一重引用符(')及び二重引用符(")も対応する対で出現し，その間の文字は，*文字列*として構文解析される。（文字列の定義については，**附属書 B**のトークン化子を参照されたい。）次にブロックの例を示す。

この例で，引用符の間の右波括弧は，ブロックの開き波括弧とは合致しないこと，2 番目の一重引用符はエスケープ文字となること，そこで開き引用符とは合致しないこと，に注意。

```
{ causta: "}" + ({7} * '¥") }
```

規則集合は，*選択子文字列*とそれに続く*宣言ブロック*とから成る。選択子文字列は，最初の左波括弧({)まで（ただし，これを含まない。）のすべてから構成される。妥当な CSS1 でない，選択子文字列で始まる規則集合は，スキップされる。

例えば，CSS1 パーサが次のスタイルシートに出会うと仮定する。

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

2 行目は，CSS1 では不正な選択子文字列を含む。CSS1 UA は，規則集合をスキップし，スタイルシートを次のとおりに縮退する。

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

宣言ブロックは，左波括弧({)で開始し，対応する右波括弧(})で終了する。この間に，セミコロン(;)で分離された 0 個以上の*宣言*のリストが一つ存在する。

宣言は，*特性*，コロン(:)及び*値*で構成される。これらの前後に空白があってもよい。特性は，先に定義した識別子とする。あらゆる文字が値に出現するが，丸括弧(())，角括弧([])，波括弧({})，一重引用符(')及び二重引用符(")は，対で出現しなければならない。丸括弧，角括弧及び波括弧は，入れ子になっていてもよい。引用符の中では，文字は，文字列として構文解析される。

新たな特性及び既存の特性に対する新たな値を，将来，追加可能とするため，UA は不正な特性名又は不正な値をもつ宣言をスキップしなければならない。すべての CSS1 特性は，それが受容する値について，それ自体の構文及び意味の制約をもつ。

例えば，CSS1 パーサが次のスタイルシートに出会うとする。

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

最初の行の 2 番目の宣言は，不正な'12pt'をもつ。2 行目の 2 番目の宣言は，未定義の特性'font-vendor'を含む。CSS1 パーサは，これらの宣言をスキップし，スタイルシートを次のとおりに縮退する。

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

コメント（**1.7**を参照）は，空白が出現できるすべての箇所に出現できる。CSS1 は，（値の内部などの）空白が出現できる付加的な位置を定義する。同様に，コメントもそこに書いてよい。

次の規則は，常に成立する。

a) すべての CSS スタイルシートは，CSS の管理下にない部分を除き，大文字及び小文字を区別しない。つまり，CSS1 では，フォントファミリ名及び URL は，大文字及び小文字を区別しない。CLASS 属性及び ID 属性の大文字及び小文字の区別は，HTML（**8.**の[2]を参照）の管理下にある。

b) CSS1 では，選択子（要素名，クラス及び ID）は，A～Z，0～9 及び Unicode 文字の 161～255 にダッシュ(-)を加えたものだけを含めることができる。ダッシュ又は数字で開始はできない。エスケープさ

れた文字及び数字コード（次項を参照）としての Unicode 文字も含むことができる。

c) ¥（逆スラッシュ。日本語環境では通貨の円記号。）に続く最大 4 個の 16 進数字(0..9 A..F)は，その番号の Unicode 文字を表す。

d) 16 進数字を除く任意の文字は，¥を前に置くことによって，特別な意味を取り除き，エスケープできる。例えば，"¥""は一つの二重引用符から成る文字列とする。

e) 先の二つの項目は，*逆スラッシュエスケープ(backslash-escape)*と定義する。逆スラッシュエスケープは，文字列の中を除き常に識別子の一部と考える。（つまり，"¥7B"は語を区切るものではないが，"{"はそうで，"¥32"はクラス名の最初に存在してもよいが，"2"は許されない。）

備考 HTML の CLASS 属性は，選択子のために許容される集合以外の多くの文字をクラス名として許す。CSS1 では，これらの文字は，エスケープされるか，又は Unicode 番号として記述されなければならない。"B&W?"は，"B¥&W¥?"又は"B¥26W¥3F"として，"κουρος"（ギリシャ語の"kouros"）は，"¥3BA¥3BF¥3C5¥3C1¥3BF¥3C2"として記述されなければならない。CSS の今後の版では，より多くの文字を直接入力できることが期待される。

附属書 Bに，CSS1 の文法を示す。

8. 引用規格及びその他の文献

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

[2] **TR X 0033**:2002 ハイパテキストマーク付け言語(HTML) 4.0

備考1. "HTML 4.0 Specification", D. Raggett, A. Le Hors, I. Jacobs, December 1997, http://www.w3.org/TR/REC-html40/ が，この規定に一致している。

2. **JIS X 4156**:2000 ハイパテキストマーク付け言語(HTML)は，W3C の HTML 4.0 を洗練化した **ISO/IEC 15445**:2000 に一致している。

[3] T Berners-Lee, D Connolly: "Hypertext Markup Language - 2.0", RFC 1866, MIT/W3C, November 1995. この規定は，ハイパテキスト形式 (http://www.w3.org/pub/WWW/MarkUp/html-spec/html-spec_toc.html)でも得られる。

[4] F Yergeau, G Nicol, G Adams, M Dürst: "Internationalization of the Hypertext Markup Language" (ftp://ietf.org/internet-drafts/draft-ietf-html-i18n-05.txt).

[5] **JIS X 4151**:2001 文書記述言語 SGML

備考 **ISO 8879**:1986 Information Processing - Text and Office Systems - Standard Generalized Markup Language(SGML)に，**ISO 8879**:1986/**Amd.1**:1988 の内容，技術的追加及び編集上の変更を加え，さらに **ISO 8879**:1986/**Cor.1**:1996 及び **ISO 8879**:1986/**Cor.2**:1999 の内容を追加したものが，この規格に対応する。

[6] **JIS X 4153**:2002 文書スタイル意味指定言語(DSSSL)

備考 **ISO/IEC 10179**:1996 Information technology - Processing languages - Document Style Semantics and Specification Language(DSSSL)に **ISO/IEC 10179**:1996/**Cor.1**:2001 の内容を追加したものが，この規格に一致している。**ISO/IEC 10179** は，ハイパテキスト形式 (http://occam.sjf.novell.com:8080/dsssl/dsssl96)でも得られる。

[7] **JIS X 3010**:1993 プログラム言語C

備考 **ISO/IEC 9899**:1990 Programming languages - C.が，この規格に一致している。

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

[9] M Anderson, R Motta, S Chandrasekar, M Stokes: "Proposal for a Standard Color Space for the Internet - sRGB" (http://www.hpl.hp.com/personal/Michael_Stokes/srgb.htm)

[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/pub/WWW/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. CSS1 開発貢献者 HTML の短い歴史の中で，これまでに幾つかのスタイルシート提案が行われた。この提案も，これらに負っている。特に Robert Raisch，Joe English 及び Pei Wei の提案からは影響を受けた。

多くの人々が，CSS1 の開発に貢献した。特に，次の方々に感謝する。

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 and Stephen Zilles

次の 3 名には特に触れておきたい。Dave Raggett からは，HTML3 に関する助力及び作業を，Chris Lilley からは，継続した貢献，特にカラー及びフォント分野での貢献を，Steven Pemberton からは，彼の組織力及び創造的な能力に関して，特に協力を頂いた。

附属書 A（参考）HTML2.0 用のスタイルシート例

この附属書（参考）は，本体及び附属書（規定）に関連する事柄を補足するもので，規定の一部ではない。

次のスタイルシートは，HTML 2.0（**8.**の[3]を参照）規定で提案する表現に従って記述した。幾つかのスタイル, 例えばカラーは, 完全を期すために追加した。次のスタイルシートに類似したスタイルシートを，UA デフォルトとして用いることを勧める。

```
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 }   /* relative to the 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 }                    /* tight formatting */
LI { margin-left: 3em }

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

HR { border-top: solid }              /* 'border-bottom' could also have been used */

A:link { color: blue }                /* unvisited link */
A:visited { color: red }              /* visited links */
A:active { color: lime }              /* active links */

/* setting the anchor border around IMG elements
     requires contextual selectors */

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

附属書 B（規定）CSS1 文法

すべての実装がサポートする必要がある最小の CSS 文法（つまり，CSS 文法の任意の版）は，**7.**で規定する。次の文法は，もっと小さな言語，つまり CSS1 構文を定義する言語を規定する。

しかし，これはある意味では，CSS1 の上位集合でもある。つまり，この文法においては表現されない付加的な意味制約がある。適合する UA は，上位互換の構文解析規則(**7.1**)，特性及び値の記法(**5.**)並びに単位の記法(**6.**)をも守らなければならない。さらに，HTML は，例えば CLASS 属性の可能な値についての制限を課す。

次の文法は，LL(1)を示す。（しかし，多くの UA は，直接これを使わなくともよい。これは構文解析規約を示すものではなく，CSS1 の構文だけを示す。）生成規則のフォーマットは，人が用いるために最適化され，yacc（**8.**の[15]を参照） では扱えない幾つかの簡略表記を用いる。

* : 0 以上
+ : 1 以上
? : 0 又は 1
| : 選択対象を分離
[] : グループ化

生成規則は次のとおり。

```
stylesheet
  : [CDO|CDC]* [ import [CDO|CDC]* ]* [ ruleset [CDO|CDC]* ]*
  ;
import
  : IMPORT_SYM [STRING|URL] ';'                    /* E.g., @import url(fun.css); */
  ;
unary_operator
  : '-' | '+'
  ;
operator
  : '/' | ',' | /* empty */
  ;
property
  : IDENT
  ;
ruleset
  : selector [ ',' selector ]*
    '{' declaration [ ';' declaration ]* '}'
  ;
selector
  : simple_selector+ [ pseudo_element | solitary_pseudo_element ]?
  | solitary_pseudo_element
```

```
  ;
      /* An "id" is an ID that is attached to an element type
      ** on its left, as in: P#p007
      ** A "solitary_id" is an ID that is not so attached,
      ** as in: #p007
      ** Analogously for classes and pseudo-classes.
      */
simple_selector
  : element_name id? class? pseudo_class?      /* eg: H1.subject */
  | solitary_id class? pseudo_class?           /* eg: #xyz33 */
  | solitary_class pseudo_class?                       /* eg: .author */
  | solitary_pseudo_class                      /* eg: :link */
  ;
element_name
  : IDENT
  ;
pseudo_class                                   /* as in:   A:link */
  : LINK_PSCLASS_AFTER_IDENT
  | VISITED_PSCLASS_AFTER_IDENT
  | ACTIVE_PSCLASS_AFTER_IDENT
  ;
solitary_pseudo_class                          /* as in:   :link */
  : LINK_PSCLASS
  | VISITED_PSCLASS
  | ACTIVE_PSCLASS
  ;
class                                          /* as in:   P.note */
  : CLASS_AFTER_IDENT
  ;
solitary_class                                 /* as in:   .note */
  : CLASS
  ;
pseudo_element                                         /* as in:   P:first-line */
  : FIRST_LETTER_AFTER_IDENT
  | FIRST_LINE_AFTER_IDENT
  ;
solitary_pseudo_element                        /* as in:   :first-line */
  : FIRST_LETTER
  | FIRST_LINE
  ;
```

```
        /* There is a constraint on the id and solitary_id that the
        ** part after the "#" must be a valid HTML ID value;
        ** e.g., "#x77" is OK, but "#77" is not.
        */
    id
      : HASH_AFTER_IDENT
      ;
    solitary_id
      : HASH
      ;
    declaration
      : property ':' expr prio?
      | /* empty */                          /* Prevents syntax errors... */
      ;
    prio
      : IMPORTANT_SYM                        /* !important */
      ;
    expr
      : term [ operator term ]*
      ;
    term
      : unary_operator?
        [ NUMBER | STRING | PERCENTAGE | LENGTH | EMS | EXS
        | IDENT | hexcolor | URL | RGB ]
      ;
        /* There is a constraint on the color that it must
        ** have either 3 or 6 hex-digits (i.e., [0-9a-fA-F])
        ** after the "#"; e.g., "#000" is OK, but "#abcd" is not.
        */
    hexcolor
      : HASH | HASH_AFTER_IDENT
      ;
```

次にトークン化子を示す。flex（**8.**の[16]を参照）記法で記述する。これは，flex の 8 ビット実装を仮定していることに注意。トークン化子は，大文字及び小文字を区別しない（flex コマンドラインオプションでは，-i に相当する。）。

```
    unicode         ¥¥[0-9a-f]{1,4}
    latin1          [!-y]
    escape          {unicode}|¥¥[ -~!-y]
    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 whitespace */}
¥n                              {BEGIN(0); /* ignore whitespace */}
¥<¥!¥-¥-                        {BEGIN(0); return CDO;}
¥-¥-¥>                                  {BEGIN(0); return CDC;}
.                               {fprintf(stderr, "%d: Illegal character (%d)¥n",
                                 lineno, *yytext);}
```

附属書 C（規定）符号化

HTML 文書は，Unicode で定義する約 3 万の文字を含んでよい。多くの文書では，数 100 文字だけを必要とする。多くのフォントも，数 100 のグリフだけを含む。**5.2** と合わせて，この附属書は，文書中の文字及びフォント中のグリフを合致させる方法を示す。

C.1　文字符号化　HTML 文書の内容は，文字の並び及びマーク付けとする。"回線上を"送るために，幾つかの利用可能な 符号化の一つを利用し，バイトの並びとして符号化する。HTML 文書は，文字を見つけるためには復号しなければならない。例えば，西ヨーロッパ語では，a-with-grave-accent(a)にはバイト 224 を，ヘブライ語では，普通，Aleph に 224 を使う。日本語では，一つのバイトの意味は，普通，それに先行するバイト群に依存する。ある符号化では，1 文字は，2（又はそれ以上の）バイトとして符号化される。

UA は，HTTP ヘッダ内の"charset"パラメタを見ることによって，バイトを復号する。典型的な符号化（文字集合値）には，"ASCII"（英語），"ISO-8859-1"（西ヨーロッパ語），"ISO-8859-8"（ヘブライ語），"Shift-JIS"（日本語）などがある。

HTML（**8.**の[2],[4]を参照）は，Unicode で定義した 3 万の文字を許容する。これらの多くの文字を使う文書は多くなく，普通は，適切な符号化を選択すれば，1 文字に 1 バイトだけを必要とする。符号化領域の外の臨時の文字を，数値的文字参照として入れることもできる。'Π'は，どの符号化を使用しても，常にギリシャ文字の大文字 Pi を意味する。このために，UA が少しの符号化だけを扱っている場合でも，あらゆる Unicode 文字を準備しなくてはならないことに注意。

C.2　フォント符号化　フォントは，文字を含まず，グリフとして知られる文字の図を含む。グリフは，アウトライン又はビットマップの形式で，文字の特定の表現を構成する。明示的に又は暗黙的に，各フォントは，それに対応する表，フォント符号化表をもつ。この表は，各グリフに対して，それが表現となる文字を示す。タイプ 1 フォントでは，符号化ベクトルとして表を参照する。

実際には，多くのフォントは同じ文字に対して複数のグリフを含む。これらグリフのどれを使うのがよいかは，言語の規則又は設計者の好みに依存してよい。

例えばアラビア文字では，すべての字は，四つの異なる形状をもつ。これは，字が，語の最初，語の中程，語の最後又は語とは孤立して用いられたかに依存する。これらは，すべての場合において同じ文字であって，HTML 文書の中ではただ一つの文字だけが存在するが，印刷された時には，それぞれに違って見える。

提供された様々な代替の形状の中からの選択が，グラフィック設計者に任されたフォントもある。残念なことに，CSS1 は，これらの代替を選択する手段をまだ提供していない。現在では，これらフォントから選択されるものは，常に，特定のデフォルト形状とする。

C.3　フォント集合　一つのフォントが，一つの文書又は一つの要素の中のすべての文字を表示するには十分でないという問題を処理するために，CSS1 は，フォント集合を使うことができる。

CSS1 におけるフォント集合は，フォントのリストであって，ある文字のグリフを含むかどうかを順番に見ることが可能な，同じスタイル及びサイズすべてとする。数学記号が混在する英文テキストを含む要素は，文字及び数字を含むフォントと数学記号を含むフォントとの，二つのフォント集合を必要とするかもしれない。フォント集合の選択機構の詳細については，**5.2** を参照のこと。

ラテン文字，日本語文字及び数学記号を含むことが期待されるテキストに適したフォント集合の例を，

次に示す。

BODY { font-family: Baskerville, Mincho, Symbol, serif }

Baskerville フォント（ラテン文字だけをもつフォント）で利用可能な文字にはこのフォントを，日本語には明朝を，数学記号には Symbol を使用する。他の文字には，一般的なフォントファミリの'serif'を，（恐らく）使用する。'serif'フォントファミリは，他の複数のフォントが使用可能でない場合にも使用する。

附属書 D（参考）ガンマ補正

この附属書（参考）は，本体及び附属書（規定）に関連する事柄を補足するもので，規定の一部ではない。

ガンマに関することをよく知らない場合には，PNG 規定（**8.**の[12]を参照）の中の ガンマの解説(Gamma Tutorial)を参照すること。

計算上では，CRT 上に表示をする UA は，理想的な CRT を仮定し，ディザによって生じる見かけのガンマの影響を無視してよい。このことは，現在のプラットフォーム上で必要とする最小限の操作が，次のものだけとなることを意味する。

a) MS-Windows を用いた PC なし

b) X11 を用いた UNIX なし

c) QuickDraw を用いた Mac ガンマ 1.39 を適用する（**8.**の[13]を参照）。（ColorSync-savvy 応用は，正しいカラー補正を実行するために，sRGB ICC プロファイル （**8.**の[14]を参照）を，単に ColorSync に渡す。）

d) X を用いた SGI /etc/config/system.glGammaVal からの値を適用する。（デフォルト値は，1.70。Irix 6.2 又はそれ以上の版で動作する応用は，sRGB ICC プロファイルを，単にカラー管理システムに渡す。）

e) NeXTStep を用いた NeXT ガンマ 2.22 を適用する。

"ガンマを適用する"の意味は，三つの R，G 及び B のそれぞれを，OS へと処理を渡す前に，$R'=R^{gamma}$，$G'=G^{gamma}$ 及び $B'=B^{gamma}$ に変換することを意味する。

これは，256 要素のルックアップ表を構成することによって，ブラウザを呼び出す度に 1 回行ってもよい。

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

これによって，1 画素当たりのカラー属性の指数関数計算(pow) の回数を劇的に減らすことができる。

附属書 E（参考）CSS1 の適用可能性及び拡張可能性

この附属書（参考）は，本体及び附属書（規定）に関連する事柄を補足するもので，規定の一部ではない。

CSS1 の作業の目的は，HTML 文書のための単純なスタイルシート機構を作成することにあった。現在の規定は，ウェブ上でスタイルシートを実現するために必要な簡易性とより充実した視覚制御に対する文書作成者からの要求とのバランスしたものとなっている。CSS1 は，次を提供する。

a) 視覚マーク付けの置換。"CENTER"，"FONT"，"SPACER"などの HTML 拡張を，容易に CSS1 スタイルシートに置換できる。

b) もっと良いマーク付け。"FONT"要素を用いて普通のスモールキャップのスタイルとする代わりに，一つのスタイルシートで宣言で十分となる。次の視覚マーク付け

```
<H1>H<FONT SIZE=-1>EADLINE</FONT></H1>
```

を，次のスタイルシートと比較してみるとよい。

```
H1 { font-style: small-caps }
<H1>Headline</H1>
```

c) 様々な統合水準。CSS1 のスタイル規則は，'STYLE'要素に含まれる外部スタイルシートから取ってくるか，又は'STYLE'属性に置くことができる。後者では，HTML 拡張からの変換は容易となる。

d) 新しい効果。いくつかの新たな視覚効果を，利用者の新たな道具を提供するために追加した。印刷の擬似要素及び背景特性の付加的な値は，この分類に入る。

e) スケール化可能性。CSS1 は，テキスト端末から高解像度のカラーワークステーションまでの範囲の機器に有効となる。文書作成者は，一つのスタイルシートを書くことができ，意図したスタイルシートが最も適当な形で実行されると確信できる。

CSS1 は，次を提供しない。

a) 画素単位の制御。CSS1 は，制御のレベルに関する簡潔さに価値を置く。背景画像とスタイル処理した HTML との合成は強力だが，画素レベルの制御は可能ではない。

b) 文書作成者の制御。文書作成者は，特定のスタイルシートの利用の強制はできず，提示だけができる。

c) レイアウト言語。CSS1 は，テキストフロー，重複枠などの多重欄を提供しない。

d) 構文解析木についての豊富な問合わせ言語。他のスタイルシート言語（例えば，DSSSL（**8.**の[6]を参照）は完全な問合わせ言語をもつが，CSS1 は，構文解析木の祖先要素を探すだけとする。

次のいくつかの方向に，CSS が拡張される期待がある。

a) 紙。HTML 文書の印刷のよりよいサポート。

b) 非視覚メディアのサポート。会話及び点字出力をサポートするために，属性及びそれに対応する値のリストを追加する作業が進行中である。

c) カラー名。現在サポートされているリストは，拡張されるかもしれない。

d) フォント。既存の CSS1 フォント特性を補完するために，より正確なフォント規定のシステムが期待される。

e) 値及び属性。ベンダによる，値及び属性の CSS1 集合への拡張提案が期待される。この方向の拡張は，規定としては些細だが，異なる UA 間での相互運用性としては重要となる。

f) レイアウト言語。伝統的なデスクトップ出版における 2 次元レイアウトのためのサポート。

g) 他の DTD。CSS1 は，いくつかの HTML 固有な部分（例えば，'CLASS'属性及び'ID'属性の特別な状態）をもつが，同様に他の DTD へ適用するために容易に拡張されることが望ましい。

CSS が次の方向に発展することは期待しない。

a) プログラミング言語。

JIS X 4168 : 2004

段階スタイルシート 水準 1(CSS1)
解　説

1.　制定の趣旨　段階スタイルシート 水準 1(CSS1)は, World Wide Web Consortium(W3C)による 1996 年 12 月の勧告[1]発表以前から, HTML 文書に対する簡便なスタイル指定の規定として多くの利用者の注目を集めると共に, ブラウザベンダによってその実装が行われてきた。1996 年 12 月の勧告が翻訳され, **TR X 0011**:1998 段階スタイルシート 水準 1(CSS1)[2]として公表されると, 国内では CSS 解説の書籍が発行されると共に, 実装例も増加して CSS1 の普及が加速された。

TR X 0011 の作成に際して明らかになった原勧告の問題点は, W3C にフィードバックされて, 1999 年 1 月の勧告に反映された。1999 年 1 月の勧告の Appendix F には, その旨が記されている。**TR X 0011** の公表後, 原案委員会は, W3C から CSS1 の JIS 化への積極支援の電子メールを受けている。

このような勧告の改訂及び標準情報(TR)の公表を経て, CSS1 は充分に安定し普及した規定内容となったことが確認されたため, 国内での一層の普及を図るために, JIS として制定することとした。

2.　制定の経緯　(社)日本事務機械工業会(当時)の標準化委員会 実装規約小委員会は, 1996 年の CSS1 の動向に着目して調査研究を継続し, "技術標準等の早期公開による JIS 化の前提となるコンセンサスの形成を促進する" という標準情報(TR)による CSS1 の公表の必要性を通商産業省工業技術院(当時)に提言した。

通商産業省工業技術院(当時)は, 1998 年 4 月に(財)日本規格協会 情報技術標準化研究センター(INSTAC)の中に電子出版技術調査研究委員会を設立し, CSS1 の翻訳作業の委託を行った。電子出版技術調査研究委員会では作業グループ WG1 がこの作業を担当し, 設立直後から 1996 年 12 月の CSS1 勧告の翻訳に着手して, 1998 年 8 月に標準情報(TR)の原案を完成した。これは, TR X 0011 として, 1998 年 10 月に公表されている。

その後, 原案委員会は CSS2 の TR 原案作成を行うと共に, 解説の **1.**に示す W3C との連携を保ち, CSS1 の改訂に寄与してきた。CSS1 の 1999 年 1 月の勧告の公表及びその後の訂正票の公表を確認した原案委員会は, 2002 年度から JIS 化の検討を開始し, JIS 原案の作成に着手した。

JIS 原案は, 2002 年度末にはほぼ完成していたが, 関連する用語について **JIS Z 8125** の検討が進められていることが明らかになった。そこで, **JIS Z 8125**[3]の制定の確定を待って, そこで規定している用語への整合を図り, 2003 年度の後半に原案を経済産業省産に提出した。

3.　審議中の主要検討課題

3.1　訳語　訳語選定に際しては, 関連の深い **JIS X 4156**[4], **JIS Z 8125** との整合に配慮すると共に, **JIS X 4175**[5], **JIS X 4159**[6], **JIS X 4173**[7], **JIS X 4151**[8], **JIS X 4153**[9]をも参照して, 訳語の共通化を図った。

この規格で採用した主な訳語(規格本体の **0.2 定義**に示した用語を除く。)を**解説表 1** に示す。

解説表 1　主な訳語

原語	訳語

anchor	アンカ
border	境界
bottom	下部
box	ボックス
cascade	段階
common effects	よく使う組み効果
conflict	競合
content	内容
core feature	コア機能
CSS1 core	CSS1 コア
fictional tag sequence	仮想タグ列
font family	フォントファミリ
font style	フォントスタイル
formatting	フォーマット化
Gamma correction	ガンマ補正
glyph	グリフ
HTML extension	HTML 拡張
inheritance	継承
initial letter	頭文字
initial character	最初の文字
inline	行内
inner bottom	内下部
inner top	内上部
intrinsic dimension	基本寸法
justify	均等割りする
leading	リーディング
left inner edge	左内辺
left outer edge	左外辺
letter	字
ligature	リガチャ
line break	改行
margin	余白
multiple pseudo-element	複数疑似要素
newline	行換え
normal face	本文書体
padding	パディング
property	特性(font property の場合だけ，属性)
pseudo-class	擬似クラス

pseudo-element	擬似要素
render	可視化する
rescale	再スケール化
scaling factor	スケール倍率
selector	選択子
shine through	透けて見える
simple selector	単純選択子
small caps	スモールキャップ
sort	ソートする
specificity	固有性
target anchor	目標アンカ
tokenizer	トークン化子
top	上部
typographical items	表示上の項目
visual effects	可視的効果
weight	重み
whitespace	空白

3.2　用語表記の揺れの統一　原勧告における用語表記の揺れは，この規格の中では**解説表 2** のとおり統一して表記している。

解説表 2　用語表記の揺れの統一

原語	訳文中での表記
UA	UA
User Agent	UA
small caps	スモールキャップ
small-caps	スモールキャップ
face	書体
font face	書体
type face	書体

3.3　章・節構成　W3C の勧告は，必ずしも JIS の様式には整合していないため，整合化の対応が必要である。しかしこの規格の読者が原勧告を参照する際の便を考慮すると，章・節構成はなるべく原勧告のそれを保存することが望まれる。そこで，次に示すだけの修正(章・節番号の変更なし)を施して，この規格を構成した。

a)　原勧告の冒頭部分(Authors, Status of this document など)をまとめて，"**原勧告の標題及びまえがきの翻訳**"として"**まえがき**"に記述する。ただし，"Changes from the original version are listed in Appendix F."の記述は，削除した。

b) "Abstract"を"**0.1 適用範囲**"とし，"Terminology"を"**0.2 定義**"として，それらを"**0. 一般**"の下位構造とする。

c) "**0. 一般**"の前に，"**序文**"を追加する。

d) 附属書の後に，"**解説**"を追加する。

e) "Appendix F: Changes from the 17 December 1996 version"は，削除する。

3.4 その他の翻訳表記上の配慮 原勧告は，HTML 及び CSS1 を用いて記述されている。この規格はできるだけそのレンダリング結果を保存することとし，特に次の点に留意した。

a) HTML(SGML)の共通識別子(要素の名前)及び属性(attribute)の名前，並びに CSS で定義した特性(property)の名前は，引用符を用いた原勧告の表記をそのまま保存する。

b) JIS の様式に合わせて，図に番号を付ける。

c) 原勧告における<pre>タグを使った図は，翻訳すると漢字フォントのメトリクスによって不ぞろ(揃)いが生じるため，ビットマップ画像の図に置き換える。

d) 原勧告では，引用規格 n(n は正整数)へのリンクを[n]で示しているが，この規格では，"**8.**の[n]"として表記している。

4. 原勧告との内容の一致 この規格は，W3C の原勧告に技術的に一致している。この規格を実装する際には，原勧告及び更新された正誤票も参照することが望ましい。

5. 文献

[1] W3C Recommendation, Cascading Style Sheets, level 1, 1996-12
[2] **TR X 0011**:1998 段階スタイルシート 水準 1(CSS1), 1998-10
[3] **JIS Z 8125**:2004, 印刷用語－デジタル印刷, 2004-02
[4] **JIS X 4156**:2000, ハイパテキストマーク付け言語, 2000-10
[5] **JIS X 4175**:2003, マルチメディア対話型文書のための交換規格(ISMID), 2003-02
[6] **JIS X 4159**:2002, 拡張可能なマーク付け言語(XML), 2002-10
[7] **JIS X 4173**:2002, 標準一般化マーク付け言語(SGML)システムの適合性試験, 2002-08
[8] **JIS X 4151**:2001, 文書記述言語 SGML, 2001-01
[9] **JIS X 4153**:2002, 文書スタイル意味指定言語(DSSSL), 2002-10

6. 原案作成委員会 この規格の原案を作成した(財)日本規格協会 情報技術標準化研究センター(INSTAC)の電子出版技術調査研究委員会及び作業グループ WG1 の委員構成を，それぞれ**解説表 3** 及び**解説表 4** に次に示す。

解説表 3 電子出版技術調査研究委員会

	氏名	所属
(委員長)	池田 克夫	大阪工業大学
(幹事)	内山 光一	株式会社東芝
(幹事)	大久保 彰徳	株式会社リコー
(幹事)	小町 祐史	パナソニック コミュニケーションズ株式会社

(幹事)	長村 玄	ネクストソリューション株式会社
	植村 八潮	東京電機大学出版局
	礪波 道夫	読売新聞社
	内藤 求	株式会社シナジー・インキュベート
	中村 幹	株式会社印刷学会出版部
	平山 亮	金沢工業大学
	赤木 孝次	社団法人日本新聞協会
	矢ケ崎 敏明	キヤノン株式会社
	堀坂 和秀	経済産業省産業技術環境局
(事務局)	内藤 昌幸	財団法人日本規格協会

解説表 4 作業グループ WG1

	氏名	**所属**
(主査)	小町 祐史	パナソニック コミュニケーションズ株式会社
(幹事)	内山 光一	株式会社東芝
(幹事)	平山 亮	金沢工業大学
	今門 政記	モスインスティテュート株式会社
	石野 恵一郎	アンテナハウス株式会社
	大久保 彰徳	株式会社リコー
	内藤 求	株式会社シナジー・インキュベート
	長村 玄	ネクストソリューション株式会社
	藤島 雅宏	有限会社イー・エイド
	矢ケ崎 敏明	キヤノン株式会社
	山田 篤	財団法人京都高度技術研究所
(オブザーバ)	浅利 千鶴	浅利会計事務所
(オブザーバ)	赤木 孝次	社団法人日本新聞協会
(オブザーバ)	堀坂 和秀	経済産業省産業技術環境局
(事務局)	内藤 昌幸	財団法人日本規格協会