

CSS 実習マニュアル

第零版 revision02

2008 年 7 月 30 日 (水)

Copyright © 2007–2008 Daikoku Manabu

This tutorial is licensed under a Creative Commons Attribution 2.1 Japan License.

目次

第 1 章	CSS の基礎	3
1.1	CSS の基礎の基礎	3
1.1.1	スタイルシートとは何か	3
1.1.2	CSS とは何か	3
1.1.3	この文章について	3
1.2	スタイルシート	3
1.2.1	ルール	3
1.2.2	セレクタ	4
1.2.3	宣言ブロック	4
1.2.4	宣言	4
1.2.5	ホワイトスペース	4
1.2.6	注釈	5
1.3	スタイルの適用	5
1.3.1	HTML 文書にスタイルを適用する二つの方法	5
1.3.2	スタイルシートファイル	5
1.3.3	link 要素	5
1.3.4	style 要素	6
1.3.5	スタイルシートの部品化	7
1.4	色	7
1.4.1	色の基礎	7
1.4.2	16 進数による色の記述	8
1.4.3	10 進数による色の記述	8
1.4.4	色名	9
1.4.5	Web セーフカラー	9
1.5	長さの単位	10
1.5.1	長さの単位の基礎	10
1.5.2	絶対単位	10
1.5.3	相対単位	11
1.5.4	em によるフォントの大きさの記述	11
1.6	セレクタ	12
1.6.1	セレクタについての復習	12
1.6.2	クラスセレクタ	12
1.6.3	ID セレクタ	13
1.6.4	子孫セレクタ	13
1.6.5	擬似クラスセレクタ	14
1.6.6	擬似要素セレクタ	15
1.6.7	グループ化	16
1.7	継承と詳細度とカスケード	16
1.7.1	継承	16
1.7.2	詳細度	17
1.7.3	カスケード	17

第 2 章	テキスト	18
2.1	フォント	18
2.1.1	この章とこの節について	18
2.1.2	フォントファミリー	18
2.1.3	フォントの太さ	20
2.1.4	フォントのスタイル	20
2.2	テキストのレイアウト	21
2.2.1	この節について	21
2.2.2	ブロックレベル要素とインライン要素	21
2.2.3	水平方向のアラインメント	21
2.2.4	インデント	22
2.2.5	行の高さ	22
2.2.6	テキストの折り返しの抑制	23
2.3	テキストの装飾	24
2.3.1	下線と上線と取り消し線	24
2.3.2	下線のないアンカー	24
第 3 章	ボックス	25
3.1	ボックスの基礎	25
3.1.1	ボックスとは何か	25
3.1.2	ボックスを構成する長方形	25
3.1.3	ボックスを構成する領域	25
3.1.4	背景色	25
3.2	パディング	25
3.2.1	パディングの基礎	25
3.2.2	パディングの上下左右	26
3.3	ボーダー	27
3.3.1	ボーダーの形状	27
3.3.2	ボーダーの幅	28
3.3.3	ボーダーの色	29
3.3.4	ボーダーの一括設定	29
3.3.5	ボーダーの上下左右	30
3.4	マージン	31
3.4.1	マージンの基礎	31
3.4.2	マージンの上下左右	32
3.5	フローティング	32
3.5.1	フローティングの基礎	32
3.5.2	回り込み	33
3.5.3	コンテンツエッジの横の長さ	33
3.5.4	段組み	34
3.6	テーブル	35
3.6.1	ボックスとしてのテーブル	35
3.6.2	垂直方向のアラインメント	36
3.7	リスト	37
3.7.1	マーカー	37
3.7.2	マーカーのタイプ	37
3.7.3	マーカーの位置	38
	参考文献	38
	索引	40

第1章 CSSの基礎

1.1 CSSの基礎の基礎

1.1.1 スタイルシートとは何か

文書というものは、基本的には文字が並んでできているわけですが、それに加えて、構造とスタイルというものを持つことも可能です。

文書の構造というのは、文書の個々の部分がどのように組み合わせられて全体を構成しているのかということです。そして、文書のスタイルというのは、紙や画面などの上に文書をどのように表示するのかということです。

文書の構造は、通常、「マークアップ言語」(markup language) と呼ばれる言語によって記述されます。マークアップ言語としては、troff、HTML¹、L^AT_EX などがあります。

文書のスタイルは、通常、その文書とは別の文書の中に記述されます。文書のスタイルを記述した文書は、「スタイルシート」(style sheet) と呼ばれます。

1.1.2 CSSとは何か

スタイルシートは、「スタイルシート言語」(style sheet language) と呼ばれる言語によって記述されます。スタイルシート言語としては、XSL や CSS などがあります。

XSL と CSS は、どちらも、XML 応用言語を使って書かれた文書のスタイルを記述するためのスタイルシート言語です。XSL という名前は Extensible Stylesheet Language という名前を縮めて作られたもので、CSS という名前は Cascading Style Sheets という名前を縮めて作られたものです。

XSL と CSS の標準規格は、W3C(World Wide Web Consortium)² という組織によって定められています。W3C は、ウェブで使われるさまざまな言語などの標準規格について検討している組織です。W3C によって定められている標準規格としては、XSL と CSS のほかに、XML、HTML、SVG、RDF、OWL、DOM などがあります。

1.1.3 この文章について

この「CSS 実習マニュアル」という文章は、ウェブページ、つまり HTML 文書のスタイルを、CSS を使って記述する方法について解説したものです。

ちなみに、CSS によって記述されたスタイルを適用することのできる文書は、HTML 文書だけではありません。しかし、HTML 文書以外のスタイルは、この文章の対象外です。

この文章は、読者は既に HTML については理解している、ということを想定して書かれています。ですから、HTML というのがどのようなものなのか、まだあまり知らないという読者は、この文章を読む前に、あらかじめそれについて学習しておく必要があります。

1.2 スタイルシート

1.2.1 ルール

CSS で書かれたスタイルシートというのは、基本的には、「ルール」(rule) と呼ばれる記述がいくつか並んだものだと考えることができます。

ルールというのは、「どのような部分をどのようなスタイルにするのか」ということを記述したもののことです。たとえば、

```
p { color: green; }
```

というルールを書くことによって、「すべての p 要素の文字の色を緑色にする」ということを記述することができます。

すべてのルールは、

```
セレクタ 宣言ブロック
```

という構文にしたがって書かれます。たとえば、

```
p { color: green; }
```

¹この文章の中では、HTML という名前を、HTML と XHTML の総称として使うことにします。

²URL は、<http://www.w3.org/> です。

というルールの場合、`p`という部分がセレクタで、

```
{ color: green; }
```

という部分が宣言ブロックです。

1.2.2 セレクタ

「セレクタ」(selector) というのは、スタイルを適用する対象を記述したもののことです。セレクタとして要素型名を書くと、その要素型を持つすべての要素に対して、宣言ブロックで記述されたスタイルが適用されます。たとえば、

```
em { color: red; }
```

というルールを書くことによって、すべての `em` 要素の文字の色を赤色にすることができます。

セレクタについては、第1.6節で、もう少し詳しく説明したいと思います。

1.2.3 宣言ブロック

宣言ブロックは、かならず、左中括弧 (`{`) で始まって、右中括弧 (`}`) で終わります。そして、その中に、「宣言」(declaration) と呼ばれるものを書きます。宣言は、ひとつの宣言ブロックの中に何個でも好きなだけ書くことができます。たとえば、

```
{ color: green; background-color: aqua; }
```

という宣言ブロックの中には、「文字の色を緑色にする」という宣言と「背景の色を水色にする」という宣言とが含まれています。

1.2.4 宣言

宣言というのは、「スタイルの特定の種類に対して、それをどうするのか」ということを記述したもののことです。すべての宣言は、

```
プロパティ : 値 ;
```

という構文にしたがって書かれます。たとえば、

```
color: green;
```

という宣言の場合は、`color` という部分がプロパティで、`green` という部分が値です。

「プロパティ」(property) というのは、スタイルの種類をあらわしている名前のことです。たとえば、`color` というのは前景色 (文字の色) というスタイルの種類をあらわしているプロパティで、`background-color` というのは背景色というスタイルの種類をあらわしているプロパティです。

「値」(value) というのは、プロパティで指定された種類のスタイルをどうするのかという具体的な記述のことです。

1.2.5 ホワイトスペース

空白 (space)、タブ (tab)、改行 (line feed)、復帰 (carriage return)、改ページ (form feed) という5種類の文字は、総称して「ホワイトスペース」(white space) と呼ばれます。

CSS では、単語の前後に任意の個数のホワイトスペースを挿入することができます (ただし、コロン (`:`) の左側にホワイトスペースを挿入することはできません)。そして、ホワイトスペースの有無はスタイルシートの意味を変化させません。たとえば、

```
p { color: green; }
```

というルールと、

```
p {
  color: green;
}
```

というルールとを比較した場合、それらの相違点はホワイトスペースの有無だけですので、それらは同じ意味を持つことになります。

値というのは、通常、何個かの単語から構成される列です。2個以上の単語から値が構成される場合、それらの単語はホワイトスペースで区切られている必要があります。

1.2.6 注釈

スタイルシートの中には、CSS の文法を無視した自由な文字列を書くことも可能です。この機能を利用することによって、スタイルシートを読む人間（書いた本人も含みます）に対する何らかの説明をスタイルシートの中に書き込むことができます。そのような、人間に対する説明のための記述は、「注釈」(comment) と呼ばれます。

CSS では、`/*` で始まって `*/` で終わる文字列は注釈とみなされますので、その中には、CSS の文法を無視した自由な文字列を書くことができます。たとえば、

```
/* I am a comment. */
```

という文字列を、スタイルシートの中に挿入することができます。

注釈は、途中で改行を含んでもかまいません。ですから、

```
/* I am a comment  
which contains a line feed. */
```

というのも正しい注釈です。

注釈を入れ子にすること、つまり注釈の中に注釈を書くことはできません。ですから、

```
/* I am a comment /* comment */ which contains a comment. */
```

というのは正しい注釈ではありません。

スタイルシートを作成したり修正したりしているとき、その一部分を一時的に無効にしたい、ということがしばしばあります。そのような場合、無効にしたい部分を削除してしまうと、復元するのに手間がかかります。そのような場合には、通常、その部分を削除するのではなく、注釈にします。記述の一部分を注釈にすることによって、それを無効にすることを、その部分を「コメントアウトする」(comment out) と言います。

1.3 スタイルの適用

1.3.1 HTML 文書にスタイルを適用する二つの方法

スタイルシートに記述されているスタイルを HTML 文書の表示に適用する方法は、二つあります。

ひとつは、スタイルを適用したい HTML 文書とスタイルを記述したスタイルシートとを別々のファイルに格納しておいて、それらの HTML 文書とスタイルシートとを関連付けるという方法です。そしてもうひとつは、スタイルを適用したい HTML 文書の中にスタイルシートそのものを埋め込むという方法です。

ウェブサイトを構成しているそれぞれのウェブページは、デザインが統一されているというのが普通です。デザインを統一するためには、同一のスタイルを複数の HTML 文書に適用することが必要です。そのため、HTML 文書とスタイルシートは通常、別々のファイルに格納されます。

ですから、HTML 文書にスタイルを適用する二つの方法のうちで、基本となるのは、別々のファイルに格納されている HTML 文書とスタイルシートとを関連付けるという方法です。HTML 文書の中にスタイルシートを埋め込むというもうひとつの方法は、特定の HTML 文書のための特殊なスタイルを記述する場合に使われるものです。

1.3.2 スタイルシートファイル

スタイルシートが格納されているファイルは、「スタイルシートファイル」(stylesheet file) と呼ばれます。CSS で書かれたスタイルシートが格納されているスタイルシートファイルのファイル名には、通常、`.css` という拡張子を付けます。

1.3.3 link 要素

別々のファイルに格納されている HTML 文書とスタイルシートとを関連付けたいときは、その HTML 文書の head の中に、`link` という要素型の要素を書きます。

`link` 要素というのは、HTML 文書を他の文書に関連付けるために使われるもので、次のような属性を持っています。

`rel` 関連付けの種類。

`type` 関連付けるファイルの MIME タイプ。

`href` 関連付けるファイルの URL。

HTML 文書とスタイルシートとを関連付ける場合、link 要素の rel 属性には、stylesheet という単語を設定します。また、スタイルシートが CSS で書かれている場合、type 属性には、text/css という MIME タイプを設定します。

たとえば、namako.css という名前のファイルに、CSS で書かれたスタイルシートが格納されていて、そのファイルと同じディレクトリに HTML 文書のファイルがあるとするならば、その HTML 文書の head 要素の中に、

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

という link 要素を書くことによって、それらの HTML 文書とスタイルシートとを関連付けることができます。

たとえば、次のスタイルシートと HTML 文書のそれぞれを格納したファイルを同じディレクトリに置いて、そして HTML 文書のファイルをブラウザで開くと、スタイルが HTML 文書の表示に適用されますので、その中の p 要素は緑色で表示されます。

スタイルシートの例 link.css

```
p { color: green; }
```

HTML 文書の例 link.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>HTML 文書とスタイルシートとの関連付け</title>
<link rel="stylesheet" type="text/css" href="link.css" />
</head>
<body>
<p>私は段落です。</p>
<p>私も段落です。</p>
</body>
</html>
```

1.3.4 style 要素

スタイルシートそのものを HTML 文書の中に埋め込むという方法で、スタイルを HTML 文書の表示に適用したいときは、HTML 文書の head の中に、style という要素型の要素を書きます。

style 要素というのは、HTML 文書の中にスタイルシートを記述するために使われるもので、その要素の中にスタイルシートを書くと、それによって記述されたスタイルが HTML 文書に適用されます。

style 要素の中には、HTML 文書に適用したいスタイルを記述したスタイルシートを書きます。

style 要素は、type という属性を持っています。この属性には、スタイルシートを書くために使われている言語の MIME タイプを設定します。スタイルシートを CSS で書く場合、この属性には CSS の MIME タイプ（つまり text/css）を設定します。

たとえば、HTML 文書の head 要素の中に、

```
<style type="text/css">
p { color: blue; }
</style>
```

という style 要素を書くことによって、その HTML 文書の p 要素をブラウザに青色で表示させることができます。

HTML 文書の例 style.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>HTML 文書の中へのスタイルシートの埋め込み</title>
<style type="text/css">
p { color: blue; }
</style>
```

```

</head>
<body>
<p>私は段落です。</p>
<p>私も段落です。</p>
</body>
</html>

```

1.3.5 スタイルシートの部品化

スタイルシートは、自分とは別のファイルに格納されているスタイルシートを読み込んで、それを HTML の表示に反映させる、ということもできるようになっています。別のファイルに格納されているスタイルシートを読み込むことを、スタイルシートを「インポートする」(import) と言います。

スタイルシートをインポートするという機能を使うことによって、スタイルシートをいくつかの部品に分割して、それらを別々のファイルに格納しておく、ということができるようになります。

スタイルシートをインポートしたいときは、@import 文と呼ばれるものを書きます。@import 文は、

```
@import url( URL );
```

という構文を持つ記述です。スタイルシートの中に @import 文を書くこと(ただし、@import 文は、ルールよりも上に書く必要があります)、その中の URL で識別されるファイルからスタイルシートが読み込まれます。

スタイルシートの例 module.css

```
p { color: maroon; }
```

スタイルシートの例 import.css

```
@import url(module.css);
body { background: silver; }
```

HTML 文書の例 import.html

```

<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>スタイルシートの部品化</title>
<link rel="stylesheet" type="text/css" href="import.css" />
</head>
<body>
<p>私は段落です。</p>
<p>私も段落です。</p>
</body>
</html>

```

1.4 色

1.4.1 色の基礎

CSS を使うことによって、ウェブページを構成するさまざまなものを表示するための色を設定することができます。

CSS では、色を記述する方法として、

- 16 進数を使う方法
- 10 進数を使う方法
- 色名を使う方法

という 3 種類のものを使うことができます。

1.4.2 16進数による色の記述

16進数で色を記述したいときは、

#赤緑青

という形で、光の三原色のそれぞれの色の強さを書きます。「赤」「緑」「青」というそれぞれの場所には、その原色の光を混ぜ合わせる強さを、2桁の16進数で記述します。たとえば、紫色は、16進数を使うことによって、#800080と記述することができます。

次のスタイルシートは、見出しの色を16進数で記述しています。

スタイルシートの例 colhex.css

```
h1 { color: #ff8080; }
h2 { color: #ff0080; }
h3 { color: #ff8000; }
```

HTML文書の例 colhex.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>16進数による色の記述</title>
<link rel="stylesheet" type="text/css" href="colhex.css" />
</head>
<body>
<h1>#ff8080</h1>
<h2>#ff0080</h2>
<h3>#ff8000</h3>
</body>
</html>
```

16進数で色を記述する場合、それぞれの原色の強さを基本的には2桁の16進数で書くわけですが、2桁ではなくて1桁で書くことも可能です。1桁で書いた場合、それは、それと同じ数字を二つ並べたものと同じだとみなされます。たとえば、fはffと同じです。ですから、#3cfと#33ccffは同じ色を意味することになります。

1.4.3 10進数による色の記述

10進数で色を記述したいときは、

rgb(赤, 緑, 青)

という形で、光の三原色のそれぞれの色の強さを書きます。「赤」「緑」「青」というそれぞれの場所には、その原色の光を混ぜ合わせる強さを、0から255までのあいだの10進数で書きあらわします。たとえば、紫色は、10進数を使うことによって、rgb(128, 0, 128)と記述することができます。

次のスタイルシートは、見出しの色を10進数で記述しています。

スタイルシートの例 coldec.css

```
h1 { color: rgb(0, 128, 0); }
h2 { color: rgb(0, 192, 64); }
h3 { color: rgb(64, 192, 0); }
```

HTML文書の例 coldec.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>10進数による色の記述</title>
<link rel="stylesheet" type="text/css" href="coldec.css" />
</head>
<body>
```

```
<h1>rgb(0, 128, 0)</h1>
<h2>rgb(0, 192, 64)</h2>
<h3>rgb(64, 192, 0)</h3>
</body>
</html>
```

1.4.4 色名

「色名」(color keyword) というのは、色を識別するための英語の単語のことです。たとえば、紫色は、purple という色名で記述することができます。

CSS では、色名として、HTML 4.01 で定義されている 16 の単語に orange を加えた、次の 17 の単語を使うことができます。

色名	16 進数	色名	16 進数	色名	16 進数
red	#ff0000	maroon	#800000	white	#ffffff
lime	#00ff00	green	#008000	silver	#c0c0c0
blue	#0000ff	navy	#000080	gray	#808080
yellow	#ffff00	olive	#808000	black	#000000
fuchsia	#ff00ff	purple	#800080	orange	#ffa500
aqua	#00ffff	teal	#008080		

次のスタイルシートは、見出しの色を色名で記述しています。

スタイルシートの例 colword.css

```
h1 { color: blue; }
h2 { color: aqua; }
h3 { color: teal; }
```

HTML 文書の例 colword.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>色名</title>
<link rel="stylesheet" type="text/css" href="colword.css" />
</head>
<body>
<h1>blue</h1>
<h2>aqua</h2>
<h3>teal</h3>
</body>
</html>
```

1.4.5 Web セーフカラー

光の三原色のそれぞれを 256 段階で混ぜ合わせると、 $256^3 = 16777216$ の異なる色を作ることができるわけですが、いかなる環境でも、それらの色が正しく表示されるとは限りません。しかし、「Web セーフカラー」(Web safe colors) と呼ばれる 216 の色だけは、いかなる環境でも正しく表示されることが保障されています。

Web セーフカラーは、原色のそれぞれを、16 進数の 00、33、66、99、cc、ff という 6 段階で混ぜ合わせることによってできる、 $6^3 = 216$ の色です (これらの原色の強さは、1 桁の 16 進数で、0、3、6、9、c、f と書くことも可能です)。たとえば、#ff0、#396、#0fcなどは Web セーフカラーです。

次のスタイルシートは、見出しの色として Web セーフカラーを設定しています。

スタイルシートの例 websafe.css

```
h1 { color: #cf0; }
h2 { color: #69f; }
h3 { color: #c03; }
```

HTML 文書の例 websafe.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>Web セーフカラー</title>
<link rel="stylesheet" type="text/css" href="websafe.css" />
</head>
<body>
<h1>#cf0</h1>
<h2>#69f</h2>
<h3>#c03</h3>
</body>
</html>
```

1.5 長さの単位

1.5.1 長さの単位の基礎

CSS を使って記述することのできるスタイルの中には、さまざまなものの長さというものが含まれています。また、位置や大きさも、長さによって記述されます。

スタイルシートの中で長さを記述するためには、それをあらかず数値だけではなくて、その数値が依存している単位を記述することも必要です。ただし、ゼロという長さを記述する場合は、単位なしで、0 と書くだけでかまいません。

CSS では、長さの単位は、2 文字の略語によって記述されます。たとえば、センチメートルという単位をあらわす略語は cm です。

長さは、まず 10 進数を書いて、その右側に単位をあらわす略語を書くことによって記述されます。たとえば、7 センチメートルという長さは、7cm と書きます。

長さの単位は、絶対単位と相対単位の 2 種類に分類することができます。

1.5.2 絶対単位

絶対的な長さを基準とする単位は、「絶対単位」(absolute length unit) と呼ばれます。CSS では、次の 5 種類の絶対単位を使うことができます。

cm センチメートル。

mm ミリメートル。1 ミリメートルは 0.1 センチメートル。

in インチ。1 インチは 2.54 センチメートル。

pt ポイント。主として印刷業界で使われる単位。1 ポイントは、72 分の 1 インチ、0.3514 ミリメートル。

pc パイカ。主として印刷業界で使われる単位。1 パイカは、6 分の 1 インチ、12 ポイント。

font-size というプロパティは、フォントの大きさ (縦方向の長さ) をあらわします。次のスタイルシートは、見出しを表示するために使われるフォントの大きさを、絶対単位を使って記述したものです。

スタイルシートの例 absolute.css

```
h1 { font-size: 1in; }
h2 { font-size: 2cm; }
h3 { font-size: 3pc; }
h4 { font-size: 10mm; }
h5 { font-size: 20pt; }
```

HTML 文書の例 absolute.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>絶対単位</title>
```

```
<link rel="stylesheet" type="text/css" href="absolute.css" />
</head>
<body>
<h1>1in</h1>
<h2>2cm</h2>
<h3>3pc</h3>
<h4>10mm</h4>
<h5>20pt</h5>
</body>
</html>
```

1.5.3 相対単位

何らかの長さに対する比率によって長さを指定する単位は、「相対単位」(relative length unit)と呼ばれます。CSS では、次の 3 種類の相対単位を使うことができます。

em 使用中のフォントの大きさ(縦方向の長さ)を 1 とする単位。

ex 使用中のフォントで表示される小文字の x の高さを 1 とする単位。

px 使用中のモニターの画面を構成しているピクセルの大きさを 1 とする単位。

letter-spacing というプロパティは、文字の間隔をあらわします。次のスタイルシートは、見出しの文字の間隔を、相対単位を使って記述したものです。

スタイルシートの例 relative.css

```
h1 { letter-spacing: 0em; }
h2 { letter-spacing: 1em; }
h3 { letter-spacing: 2em; }
```

HTML 文書の例 relative.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>相対単位</title>
<link rel="stylesheet" type="text/css" href="relative.css" />
</head>
<body>
<h1>第一レベルの見出し</h1>
<h2>第二レベルの見出し</h2>
<h3>第三レベルの見出し</h3>
</body>
</html>
```

1.5.4 em によるフォントの大きさの記述

フォントの大きさを記述するときに単位として em を使った場合、それは、親要素に適用されたフォントの大きさに対する相対的な大きさだと解釈されます。

次のスタイルシートは、見出しを表示するために使われるフォントの大きさを、em を使って記述したものです。

スタイルシートの例 fontsize.css

```
body { font-size: 20px; }
h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h3 { font-size: 1em; }
```

HTML 文書の例 fontsize.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>em によるフォントの大きさの記述</title>
```

```
<link rel="stylesheet" type="text/css" href="fontsize.css" />
</head>
<body>
<h1>第一レベルの見出し</h1>
<h2>第二レベルの見出し</h2>
<h3>第三レベルの見出し</h3>
</body>
</html>
```

1.6 セレクタ

1.6.1 セレクタについての復習

第1.2節で説明したように、ルールの先頭にかかれる、スタイルを適用する対象の記述は、「セレクタ」(selector)と呼ばれます。

セレクタとして要素型名を書くことによって、その要素型を持つすべての要素に対して適用されるルールを記述することができます。たとえば、

```
em { color: red; }
```

というルールを書くことによって、すべての em 要素の文字の色を赤色にすることができます。この節では、セレクタの書き方について、もう少し詳細な説明を加えたいと思います。

1.6.2 クラスセレクタ

class に設定された属性値、つまりクラス名によってスタイルを適用する対象を指定するセレクタは、「クラスセレクタ」(class selector)と呼ばれます。

クラスセレクタは、ドット (dot) の右側にクラス名を書いたもの、つまり、

```
. クラス名
```

という形の記述です。このようなセレクタを書くことによって、ドットの右側に書かれたクラス名が class 属性に設定されているものに対してスタイルを適用することができます。たとえば、

```
.special
```

というセレクタを書いたとすると、special というクラス名が class 属性に設定されている要素に対してスタイルが適用されることになります。

クラスセレクタとしては、要素型名の右側にドットを書いて、そのさらに右側にクラス名を書いたもの、つまり、

```
要素型名 . クラス名
```

という形のものを書くこともできます。この形のクラスセレクタを書くことによって、ドットの左側に書かれた要素型名で指定された要素のうちで、ドットの右側に書かれたクラス名が class 属性に設定されているものに対してスタイルを適用することができます。たとえば、

```
blockquote.deluxe
```

というセレクタを書いたとすると、blockquote 要素のうちで、special というクラス名が class 属性に設定されているものに対してスタイルが適用されることになります。

スタイルシートの例 class.css

```
.normal { color: blue; }
.special { color: red; }
blockquote.deluxe { color: green; }
```

HTML 文書の例 class.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>クラスセレクタ</title>
```

```
<link rel="stylesheet" type="text/css" href="class.css" />
</head>
<body>
<p class="normal">normal クラスの段落</p>
<blockquote class="normal">normal クラスの引用文</blockquote>
<p class="special">special クラスの段落</p>
<blockquote class="special">special クラスの引用文</blockquote>
<p class="deluxe">deluxe クラスの段落</p>
<blockquote class="deluxe">deluxe クラスの引用文</blockquote>
</body>
</html>
```

1.6.3 ID セレクタ

要素の id 属性に設定された属性値、つまり ID によってスタイルを適用する対象を指定するセレクタは、「ID セレクタ」(ID selector) と呼ばれます。

ID セレクタは、シャープ (sharp) の右側に ID を書いたもの、つまり、

#ID

という形の記述です。このようなセレクタを書くことによって、シャープの右側に書かれた ID が id 属性に設定されているものだけに対してスタイルを適用することができます。たとえば、

#id000

というセレクタを書くことによって、id000 という ID が id 属性に設定されている要素だけに対してスタイルを適用することができます。

スタイルシートの例 id.css

```
#id000 { color: teal; }
#id001 { color: orange; }
#id002 { color: purple; }
```

HTML 文書の例 id.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>ID セレクタ</title>
<link rel="stylesheet" type="text/css" href="id.css" />
</head>
<body>
<p id="id000">ID が id000 の段落</p>
<p id="id001">ID が id001 の段落</p>
<blockquote id="id002">ID が id002 の引用文</blockquote>
</body>
</html>
```

1.6.4 子孫セレクタ

HTML 文書は、さまざまな要素を組み合わせることによって、木構造を構成しています。CSS では、指定された要素の子孫になっている要素だけに対してスタイルを適用する、ということができるようになっています。要素の子孫にスタイルを適用するセレクタは、「子孫セレクタ」(descendant selector) と呼ばれます。

子孫セレクタは、セレクタの右側に空白を書いて、その右側にさらにセレクタを書いたもの、つまり、

セレクタ セレクタ

という形の記述です。このようなセレクタを書くことによって、左に書かれたセレクタによって指定された要素の子孫になっている要素のうちで、右に書かれたセレクタによって指定される要素に対して、スタイルが適用されることとなります。たとえば、

#id000 em.normal

というセレクタを書くことによって、#id000というセレクタによって指定される要素の子孫になっている要素のうちで、em.normalというセレクタによって指定される要素だけに対してスタイルを適用することができます。

子孫セレクタの全体もひとつのセレクタになりますので、セレクタは、

```
セレクタ セレクタ … セレクタ
```

というように、いくつでも並べることができます。そうすることによって、の子孫のうちの子孫のうち……というように、スタイルを適用する要素を何重にも限定することができます。

スタイルシートの例 descend.css

```
p em { color: red; }
blockquote em { color: green; }
```

HTML 文書の例 descend.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>子孫セレクタ</title>
<link rel="stylesheet" type="text/css" href="descend.css" />
</head>
<body>
<p>吾輩は<em>段落</em>である。</p>
<blockquote>吾輩は<em>引用文</em>である。</blockquote>
</body>
</html>
```

1.6.5 擬似クラスセレクタ

aによって表示されるもの、つまりアンカーは、ブラウザ上での状態によってスタイルを変化させることができます。そのような、ブラウザ上での要素の状態は、「擬似クラス」(pseudo-class)と呼ばれます。擬似クラスは、「擬似クラス名」(pseudo-class name)と呼ばれる名前によって識別されます。

アンカーに関連する擬似クラスとしては、次のようなものがあります。

```
link      まだ訪問していないリンク先を持つアンカー。
visited   すでに訪問したリンク先を持つアンカー。
hover     マウスポインタが重なっている要素。
active    アクティブ化された要素(クリックされたアンカーなど)。
```

擬似クラスに対してスタイルを適用したいときは、「擬似クラスセレクタ」(pseudo-class selector)と呼ばれるセレクタを使ってルールを記述します。

擬似クラスセレクタは、セレクタの右側にコロン(colon)を書いて、その右側に擬似クラス名を書いたもの、つまり、

```
セレクタ : 擬似クラス名
```

という形の記述です。このようなセレクタを書くことによって、コロンの左に書かれたセレクタで指定された要素のうちで、コロンの右に書かれた擬似クラス名で指定された状態を持つものだけに対してスタイルを適用することができます。たとえば、

```
a.mysite:visited
```

というセレクタを書くことによって、a.mysiteというセレクタで指定されたアンカーのうちで、すでに訪問したリンク先を持つものだけに対してスタイルを適用することができます。

擬似クラスセレクタを使ったルールは、LVHAという順序、つまり、link、visited、hover、activeという順序で書く必要があります。それ以外の順序で書くと、意図した結果が得られないことがあります。

スタイルシートの例 pseudo.css

```
a:link { color: green; }
a:visited { color: navy; }
a:hover { color: orange; }
a:active { color: fuchsia; }
```

HTML 文書の例 pseudo.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>擬似クラスセレクタ</title>
<link rel="stylesheet" type="text/css" href="pseudo.css" />
</head>
<body>
<p><a href="http://www.example.com/">www.example.com</a></p>
<p><a href="http://www.example.org/">www.example.org</a></p>
</body>
</html>
```

1.6.6 擬似要素セレクタ

CSS は、HTML の要素ではないものを仮想的な要素とみなして、それに対してスタイルを適用することができる、という機能を持っています。そのような仮想的な要素は、「擬似要素」(pseudo-element) と呼ばれます。

擬似要素としては、たとえば次のようなものがあります。

first-letter 要素の最初の文字。
 first-line ブロックレベル要素の最初の行。

擬似要素に対してスタイルを適用したいときは、「擬似要素セレクタ」(pseudo-element selector) と呼ばれるセレクタを使ってルールを記述します。

擬似要素セレクタは、セレクタの右側にコロン (colon) を書いて、その右側に擬似要素名を書いたもの、つまり、

セレクタ : 擬似要素名

という形の記述です。このようなセレクタを書くことによって、コロンの左に書かれたセレクタで指定された要素が持っている、コロンの右に書かれた擬似要素名で指定された擬似要素に対して、スタイルを適用することができます。たとえば、

```
p.special:first-letter
```

というセレクタを書くことによって、p.special というセレクタで指定された段落の先頭の文字に対してスタイルを適用することができます。

次のスタイルシートは、要素の先頭の文字に対するスタイルと、要素の先頭の行に対するスタイルを記述しています。

スタイルシートの例 first.css

```
#id000:first-letter { font-size: 3em; color: green; }
#id001:first-line { font-size: 3em; color: blue; }
```

HTML 文書の例 first.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>擬似要素セレクタ</title>
<link rel="stylesheet" type="text/css" href="first.css" />
</head>
<body>
<p id="id000">最初の文字は「最」です。</p>
<p id="id001">この段落は、first-line という擬似要素を使って、
```

最初の行だけが、3倍の大きさのフォントを使って、青色で表示されるように設定されています。</p>
</body>
</html>

1.6.7 グループ化

同一のスタイルを異なる対象に適用する複数のルールは、それらの対象を指定するひとつのセレクタを書くことによって、それらのルールをひとつにまとめることができます。同一のスタイルを対象に適用する複数のルールをひとつにまとめることを、「グループ化」(grouping)と呼びます。

複数の対象を指定するセレクタというのは、セレクタをコンマ (comma) で区切って並べたもののことです。たとえば、

```
h1, h2, h3 { color: maroon; }
```

というルールを書くことによって、h1要素、h2要素、h3要素の色を栗色にすることができます。

スタイルシートの例 grouping.css

```
h1, .deluxe, #id000 { color: olive; }
```

HTML 文書の例 grouping.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>グループ化</title>
<link rel="stylesheet" type="text/css" href="grouping.css" />
</head>
<body>
<h1>第一レベルの見出し</h1>
<p class="deluxe">deluxe クラスの段落</p>
<p id="id000">ID が id000 の段落</p>
</body>
</html>
```

1.7 継承と詳細度とカスケード

1.7.1 継承

HTML 文書は、さまざまな要素を組み合わせることによって、木構造を構成しています。HTML 文書を構成している何らかの要素に対して何らかのスタイルを適用すると、そのスタイルは、その要素だけではなくて、その子供や孫、つまりすべての子孫にも適用されます。そのように、要素に適用されたスタイルが子孫の要素にも適用されることは、「継承」(inheritance)と呼ばれます。

次のスタイルシートと HTML 文書は、body 要素に適用されたスタイルが、その子供の p 要素と、その孫の em 要素にも継承されるということを示しています。

スタイルシートの例 inherit.css

```
body { color: blue; }
```

HTML 文書の例 inherit.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>継承</title>
<link rel="stylesheet" type="text/css" href="inherit.css" />
</head>
<body>
<p>吾輩は<em>段落</em>である。</p>
</body>
```

</html>

スタイルは、基本的には子孫に継承されると考えていいのですが、スタイルの種類によっては、子孫に継承されないものもあります。

1.7.2 詳細度

特定の要素に適用することのできるルールは、かならずしもひとつだけとは限りません。たとえば、

```
<h2 class="appendix">付録 A</h2>
```

という要素に適用することのできるルールとして、

```
h2 { color: maroon; }
.appendix { color: navy; }
```

という二つのものが存在する、という場合があります。

このように、競合する複数のルールが存在する場合、要素に適用されるのは、それらの中で詳細度がもっとも高いものです。

「詳細度」(specificity) というのは、ルールのそれぞれが持っている優先順位のことです。詳細度が高いルールほど、優先的に要素に適用されます。

詳細度は、そのルールを適用することのできる範囲が狭いほど高くなります。たとえば、要素型を指定しただけのルールよりもクラスを指定したルールのほうが詳細度が高くなりますし、IDを指定したルールはそれらよりも詳細度が高くなります。

次のスタイルシートと HTML 文書は、競合するルールのうちのどれを要素に適用するかということが、それぞれのルールの詳細度によって決定されるということを示しています。

スタイルシートの例 specific.css

```
p { color: green; }
.special { color: blue; }
#id000 { color: red; }
```

HTML 文書の例 specific.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>詳細度</title>
<link rel="stylesheet" type="text/css" href="specific.css" />
</head>
<body>
<p>普通の段落</p>
<p class="special">special クラスの段落</p>
<p class="special" id="id000">special クラスで、
かつ ID が id000 の段落</p>
</body>
</html>
```

1.7.3 カスケード

特定の要素に適用することのできる複数のルールが存在していて、しかもそれらの詳細度が等しい場合(つまり、競合する複数のルールが存在する場合)、実際に要素に適用されるのは、それらの中のどれなのでしょう。

CSS では、競合する複数のルールが存在する場合に、それらの中のどれを要素に適用するかということを決定するために、「カスケード」(cascade) と呼ばれる規則が定められています。

カスケードというのは、次のような規則です。

- 競合する複数のルールがひとつのスタイルシートに書かれている場合は、それらの中でもっとも下に書かれているものを要素に適用する。
- HTML 文書の中のスタイルシートと、その HTML 文書と関連付けられているスタイルシートの双方に、競合するルールが書かれている場合は、HTML 文書の中のスタイルシートに

書かれているもののほうを優先する。

- @import 文によってインポートされたスタイルシート (A) と、それをインポートしたスタイルシート (B) の双方に、競合するルールが書かれている場合は、インポートしたスタイルシート (B) に書かれているもののほうを優先する。

ちなみに、CSS(Cascading Style Sheets) という名前の中に cascading という単語が含まれているという事実は、カスケードという規則が CSS においてとても重要な意味を持っているということを示しています。

次のスタイルシートと HTML 文書は、競合する複数のルールのうちで、どれが要素に適用されるのかということを示しています。

スタイルシートの例 cascade1.css

```
h1 { color: blue; }
h2 { color: blue; }
h3 { color: blue; }
h4 { color: blue; }
```

スタイルシートの例 cascade2.css

```
@import url(cascade1.css);
h2 { color: red; }
h3 { color: red; }
h4 { color: red; }
```

HTML 文書の例 cascade.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>カスケード</title>
<link rel="stylesheet" type="text/css" href="cascade2.css" />
<style type="text/css">
h3 { color: maroon; }
h4 { color: green; }
h4 { color: teal; }
</style>
</head>
<body>
<h1>第一レベルの見出し</h1>
<h2>第二レベルの見出し</h2>
<h3>第三レベルの見出し</h3>
<h4>第四レベルの見出し</h4>
</body>
</html>
```

第2章 テキスト

2.1 フォント

2.1.1 この章とこの節について

この章では、テキストに関連するさまざまなプロパティについて説明していきたいと思います。テキストは文字から構成されます。そして、文字はフォントによって表示されます。ですから、CSS では、フォントに関連するさまざまなプロパティが定義されています。

というわけで、この節では、フォントに関連するさまざまなプロパティについて説明していきたいと思います。

2.1.2 フォントファミリー

共通の視覚的特徴を持つフォントの集合は、「フォントファミリー」(font family) と呼ばれます。フォントファミリーは、「フォントファミリー名」(font family name) と呼ばれる名前によって識別さ

れます。

font-family というプロパティは、使用するフォントファミリーをあらわします。このプロパティの値としてフォントファミリー名を書くと、そのフォントファミリー名で指定されたフォントファミリーによって要素が表示されることになります。たとえば、

```
h1 { font-family: Helvetica; }
```

というルールを書くことによって、Helvetica という名前のフォントファミリーを使って第一レベルの見出しを表示することができます。

font-family プロパティの値として、フォントファミリー名をコンマで区切って並べたものを書いてかまいません。たとえば、

```
h2 { font-family: Georgia, Times, Chicago; }
```

というルールを書くことができます。このように複数のフォントファミリー名を並べて書いた場合、ブラウザは、それらのフォントファミリーが使用可能かどうかを、左から右に向かって順番に調べていきます。そして、最初に見つかった使用可能なフォントファミリーを使って要素を表示します。

CSS は、「汎用フォントファミリー名」(generic font family name) と呼ばれるフォントファミリー名を定義しています。汎用フォントファミリー名を使うことによって、希望したフォントがすべて使用不可能だった場合に、適切なフォントファミリーを選択する上でのヒントをブラウザに与えることができます。

汎用フォントファミリー名は、次の五つです。

serif	プロポーショナルで、かつセリフを持つフォントファミリー。「プロポーショナル」(proportional) というのは、文字の横幅が一定ではないということ。「セリフ」(serif) というのは、線の端にあるひげ飾りのこと。
sans-serif	プロポーショナルで、かつセリフを持たないフォントファミリー。
monospace	プロポーショナルではない、つまり文字の横幅が一定のフォントファミリー。
cursive	筆記体などの、主として曲線で構成されるフォントファミリー。
fantasy	ほかのフォントファミリーには分類されないフォントファミリー。

次のスタイルシートと HTML 文書は、汎用フォントファミリー名を使ってフォントファミリーを指定した場合に、どのようなフォントファミリーが選択されるか、ということを示しています。ただし、その結果は、ブラウザなどの環境によって変化します。

スタイルシートの例 family.css

```
p { font-size: 30px; }
p.serif { font-family: serif; }
p.sans-serif { font-family: sans-serif; }
p.monospace { font-family: monospace; }
p.cursive { font-family: cursive; }
p.fantasy { font-family: fantasy; }
```

HTML 文書の例 family.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>フォントファミリー</title>
<link rel="stylesheet" type="text/css" href="family.css" />
</head>
<body>
<p class="serif">serif</p>
<p class="sans-serif">sans-serif</p>
<p class="monospace">monospace</p>
<p class="cursive">cursive</p>
<p class="fantasy">fantasy</p>
</body>
</html>
```

2.1.3 フォントの太さ

ひとつのフォントファミリーは、通常、太さの異なるいくつかのフォントを含んでいます。

`font-weight` というプロパティは、使用するフォントの太さをあらわします。このプロパティの値としては、100、200、300、.....というような数値を書くこともできますが、通常は、`normal` または `bold` と書きます。`normal` というのは標準的な太さ、`bold` というのは標準よりも太いという意味です。たとえば、

```
blockquote { font-weight: bold; }
```

というルールを書くことによって、引用文を標準よりも太いフォントで表示することができます。

次のスタイルシートと HTML 文書は、`normal` と `bold` がどのように異なるかということを示しています。

スタイルシートの例 `weight.css`

```
p { font-family: sans-serif; font-size: 30pt; }
p.normal { font-weight: normal; }
p.bold { font-weight: bold; }
```

HTML 文書の例 `weight.html`

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>フォントの太さ</title>
<link rel="stylesheet" type="text/css" href="weight.css" />
</head>
<body>
<p class="normal">normal</p>
<p class="bold">bold</p>
</body>
</html>
```

2.1.4 フォントのスタイル

ひとつのフォントファミリーは、通常、スタイルの異なるいくつかのフォントを含んでいます。フォントのスタイルとしては、「ローマン体」(roman)、「オブリーク体」(oblique)、「イタリック体」(italic) などがあります。ローマン体というのは直立したスタイルで、それに対してオブリーク体とイタリック体というのは斜めになったスタイルです。

オブリーク体とイタリック体の相違点は、前者はローマン体を機械的に斜めにしたデザインなのに対して、後者は、斜めにするだけではなくて、少し曲線的にデザインされているというところにあります。とは言っても、オブリーク体とイタリック体の両方を含んでいるフォントファミリーというのはほとんどないというのが実情です。

`font-style` というプロパティは、使用するフォントのスタイルをあらわします。このプロパティの値としては、`normal`、`oblique`、`italic` のいずれかを書きます。`normal` はローマン体、`oblique` はオブリーク体、`italic` はイタリック体です。たとえば、

```
blockquote { font-style: oblique; }
```

というルールを書くことによって、引用文をオブリーク体のフォントで表示することができます。

オブリーク体を含んでいないフォントファミリーでオブリーク体を指定した場合は、通常、計算によってローマン体から生成されたオブリーク体が使われます。また、イタリック体を含んでいないフォントファミリーでイタリック体を指定した場合は、オブリーク体で代用されることとなります。

次のスタイルシートと HTML 文書は、ローマン体とオブリーク体とイタリック体がどのように異なるかということ（同じかもしれませんが）を示しています。

スタイルシートの例 `style.css`

```
p { font-family: sans-serif; font-size: 30pt; }
p.normal { font-style: normal; }
p.oblique { font-style: oblique; }
p.italic { font-style: italic; }
```

HTML 文書の例 style.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>フォントのスタイル</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<p class="normal">roman</p>
<p class="oblique">oblique</p>
<p class="italic">italic</p>
</body>
</html>
```

2.2 テキストのレイアウト

2.2.1 この節について

テキストに関連するスタイルのうちで、もっとも重要なのは、テキストのレイアウト、つまりテキストを表示する位置に関連するスタイルです。

この節では、テキストのレイアウトに関連するプロパティについて説明したいと思います。

2.2.2 ブロックレベル要素とインライン要素

テキストのレイアウトに関連するプロパティという本題に入る前に、ブロックレベル要素とインライン要素について説明しておくことにしましょう。

HTMLの要素の多くは、ブロックレベル要素またはインライン要素のどちらかに分類されます。

「ブロックレベル要素」(block-level element) というのは、その前後にかならず改行が表示される要素のことです。ブロックレベル要素を作る要素型としては、div、p、blockquote、address、h1 ~ h6 などがあります。

「インライン要素」(inline element) というのは、その前後に改行が表示されない要素のことです。インライン要素を作る要素型としては、span、a、em、img などがあります。

2.2.3 水平方向のアラインメント

CSSでは、ブロックレベル要素に対して、アラインメントに関連するスタイルを適用することができます。

「アラインメント」(alignment) というのは、要素が表示される長方形の領域に対して、テキストをどのような位置に配置するか、ということです。

text-align というプロパティは、水平方向のアラインメントをあらわします。このプロパティの値としては、left、right、center などを書くことができます。left は左寄せ、right は右寄せ、center はセンタリングという意味です。たとえば、

```
blockquote { text-align: right; }
```

というルールを書くことによって、引用文を右寄せで表示することができます。

次のスタイルシートと HTML 文書は、段落に対して左寄せ、右寄せ、センタリングのそれぞれを適用すると、どのように表示されるか、ということを示しています。

スタイルシートの例 align.css

```
p { font-size: 30pt; }
p.left { text-align: left; }
p.right { text-align: right; }
p.center { text-align: center; }
```

HTML 文書の例 align.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>水平方向のラインメント</title>
<link rel="stylesheet" type="text/css" href="align.css" />
</head>
<body>
<p class="left">左寄せ</p>
<p class="right">右寄せ</p>
<p class="center">センタリング</p>
</body>
</html>
```

2.2.4 インデント

特定の行について、その先頭の位置を、それ以外の行の先頭をつなぐ直線から離れた位置に置くことを、その行を「インデントする」(indent)と言います。CSSでは、ブロックレベル要素に対して、1行目をどれだけインデントするかというスタイルを適用することができます。

text-indent というプロパティは、インデントの距離をあらわします。このプロパティの値としては、1行目を右方向へインデントする距離を書きます(マイナスの距離を書くことによって、左方向へインデントすることも可能です)。距離は、絶対単位を使って書いてもかまいませんし、相対単位を使って書いてもかまいません。相対単位を使った場合は、使用されているフォントの縦の長さを基準とする相対的な距離とみなされます。たとえば、

```
p { text-indent: 3em; }
```

というルールを書くことによって、段落の1行目を、フォントの縦の長さの3倍だけインデントすることができます。

次のスタイルシートは、段落に対して、フォントの縦の長さと同じ距離のインデントを設定しています。

スタイルシートの例 indent.css

```
p {
  font-size: 30pt;
  text-indent: 1em;
}
```

HTML 文書の例 indent.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>インデント</title>
<link rel="stylesheet" type="text/css" href="indent.css" />
</head>
<body>
<p>日本語には、段落の1行目は漢字1文字分だけインデントする、
という慣習があります。
日本語のフォントでは、
漢字の大きさは縦の長さと横の長さが同じですので、
text-indent プロパティの値として1emと書いておけば、
日本語の自然なインデントになります。</p>
</body>
</html>
```

2.2.5 行の高さ

CSSでは、ブロックレベル要素に対して、行の高さ、つまり行送りの距離というスタイルを適用することができます。

line-height というプロパティは、行の高さをあらわします。このプロパティの値は、絶対単位を使って書いてもかまいませんし、相対単位を使って書いてもかまいません。相対単位を使った場合は、使用されているフォントの縦の長さを基準とする相対的な距離とみなされます。たとえば、

```
blockquote { line-height: 1.5em; }
```

というルールを書くことによって、引用文の行の高さを、フォントの縦の長さを 1.5 倍した距離に設定することができます。

次のスタイルシートは、段落の行の高さとして、フォントの縦の長さを 3 倍した距離を設定しています。

スタイルシートの例 height.css

```
p {
  font-size: 30pt;
  line-height: 3em;
}
```

HTML 文書の例 height.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>行の高さ</title>
<link rel="stylesheet" type="text/css" href="height.css" />
</head>
<body>
<p>この段落には、行の高さとして、
フォントの縦の長さを 3 倍した距離が設定されています。
ということは、行と行とのあいだに、
フォントの縦の長さを 2 倍した距離の空白ができる
ということになります。</p>
</body>
</html>
```

2.2.6 テキストの折り返しの抑制

ブラウザは、通常、テキストを任意の位置で折り返して、それをいくつかの行に分割することができます。

しかし、white-space というプロパティに対して、nowrap という値を設定することによって、ブラウザによるテキストの折り返しを抑制することができます。

表のセルに対して、折り返しを抑制するスタイルを適用しておくこと、そのセルはかならず 1 行で表示されることとなります。

次のスタイルシートと HTML 文書は、折り返しが抑制されたセルを持つ表を表示します。

スタイルシートの例 nowrap.css

```
td { font-size: 30px; }
td.nowrap { white-space: nowrap; }
```

HTML 文書の例 nowrap.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>テキストの折り返しの抑制</title>
<link rel="stylesheet" type="text/css" href="nowrap.css" />
</head>
<body>
<table><tr>
<td>このセルは、
テキストの折り返しに関するスタイルがデフォルトのままですので、
ブラウザは、
このセルの中のテキストを自由に折り返すことができます。</td>
<td class="nowrap">このセルは 1 行で表示されます。</td>
</tr></table>
</body>
</html>
```

2.3 テキストの装飾

2.3.1 下線と上線と取り消し線

`text-decoration` というプロパティは、テキストの下線、上線、取り消し線というスタイルをあらわします。このプロパティの値としては、次のようなものを書くことができます。

```
underline    下線。
overline    上線。
line-through 取り消し線。
none        下線も上線も取り消し線も引かない。
```

たとえば、

```
blockquote { text-decoration: underline; }
```

というルールを書くことによって、引用文に下線を引くことができます。

次のスタイルシートは、`span` 要素に対して下線、上線、取り消し線のそれぞれを適用すると、どのように表示されるか、ということを示しています。

スタイルシートの例 `decora.css`

```
span.underline { text-decoration: underline; }
span.overline  { text-decoration: overline;  }
span.line-through { text-decoration: line-through; }
```

HTML 文書の例 `decora.html`

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>下線と上線と取り消し線</title>
<link rel="stylesheet" type="text/css" href="decora.css" />
</head>
<body>
<p>テキストには、
<span class="underline">下線</span>を引くことも、
<span class="overline">上線</span>を引くことも、
<span class="line-through">取り消し線</span>を
引くことも可能です。</p>
</body>
</html>
```

2.3.2 下線のないアンカー

たいていのブラウザは、下線を引くという装飾をアンカーに適用します。しかし、`a` 要素の `text-decoration` プロパティに対して、`none` という値を設定することによって、下線のないアンカーを作ることも可能です。

次のスタイルシートは、下線のないアンカーを作るためのルールを記述しています。

スタイルシートの例 `noline.css`

```
a.noline { text-decoration: none; }
```

HTML 文書の例 `noline.html`

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>下線のないアンカー</title>
<link rel="stylesheet" type="text/css" href="noline.css" />
</head>
```

```

<body>
<p>アンカーには、普通、
<a href="http://www.example.com/">下線があります</a>。
しかし、
<a class="noline" href="http://www.example.org/">
下線のないアンカー</a>を作ることにも可能です。</p>
</body>
</html>

```

第3章 ボックス

3.1 ボックスの基礎

3.1.1 ボックスとは何か

CSS では、ブラウザーは、HTML のひとつの要素に対してひとつの長方形の領域を割り当てると考えられています。ブラウザーが要素に対して割り当てる長方形の領域は、「ボックス」(box) と呼ばれます。

3.1.2 ボックスを構成する長方形

ひとつのボックスは、入れ子になった四つの長方形から構成されています。それらの長方形は、内側から順番に、「コンテンツエッジ」(content edge)、「パディングエッジ」(padding edge)、「ボーダーエッジ」(border edge)、「マージンエッジ」(margin edge) と呼ばれます。

3.1.3 ボックスを構成する領域

コンテンツエッジの内部の領域は、「コンテンツエリア」(content area) と呼ばれます。これは、要素の内容、つまり、子供の要素やテキストや画像が表示される領域です。

コンテンツエッジとパディングエッジに挟まれた領域は、「パディング」(padding) と呼ばれます。

パディングエッジとボーダーエッジに挟まれた領域は、「ボーダー」(border) と呼ばれます。ブラウザーは、ボーダーをさまざまなスタイルの線として表示することができます。

ボーダーエッジとマージンエッジに挟まれた領域は、「マージン」(margin) と呼ばれます。

3.1.4 背景色

第 1.2 節で説明したように、background-color というプロパティは、背景色というスタイルをあらわします。

background-color に設定された色が適用される領域は、コンテンツエリア、パディング、そしてボーダーまでで、マージンには適用されません。マージンは、常に無色透明ですので、親の要素に適用されている背景色がそのまま反映されます。

background-color に対して transparent という単語を設定すると、マージンの内側の領域も無色透明になって、親の要素に適用されている背景色がそのまま反映されるようになります。

3.2 パディング

3.2.1 パディングの基礎

padding というプロパティは、コンテンツエッジからパディングエッジまでの距離をあらわします。ですから、このプロパティを使うことによって、パディングの広さを設定することができます。たとえば、

```
blockquote { padding: 20px; }
```

というルールを書くことによって、引用文のコンテンツエッジからパディングエッジまでの距離として 20 ピクセルを設定することができます。

次のスタイルシートと HTML 文書は、段落のパディングをさまざまな広さに設定しています。

スタイルシートの例 padding.css

```
p { font-size: 40px; color: #00c; background-color: #9ff; }
```

```
#id000 { padding: 20px; }
#id001 { padding: 40px; }
#id002 { padding: 80px; }
```

HTML 文書の例 padding.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>パディング</title>
<link rel="stylesheet" type="text/css" href="padding.css" />
</head>
<body>
<p>この段落のパディングはデフォルトです。</p>
<p id="id000">この段落のパディングは 20 ピクセルです。</p>
<p id="id001">この段落のパディングは 40 ピクセルです。</p>
<p id="id002">この段落のパディングは 80 ピクセルです。</p>
</body>
</html>
```

3.2.2 パディングの上下左右

パディングも含めて、ボックスを構成するそれぞれの領域に関するスタイルは、上下左右のそれぞれを個別に設定することが可能です。

領域の上下左右を個別に設定するための方法は、二つあります。ひとつは、プロパティの値として、単語を空白で区切って並べたものを書くという方法で、もうひとつは、上下左右の個別のプロパティを使うという方法です。

領域のスタイルをあらわすプロパティの値として、空白で区切って4個の単語を書くと、それらは、上、右、下、左、という順番（つまり時計回りの順番）で、領域のスタイルを個別に設定していると見なされます。たとえば、

```
p { padding: 10px 20px 30px 40px; }
```

というルールは、段落のパディングについて、上を 10 ピクセル、右を 20 ピクセル、下を 30 ピクセル、左を 40 ピクセルに設定します。ちなみに、長さは、2 個だけ書くことも 3 個だけ書くことも可能です。2 個の場合は、上下、左右、とみなされ、3 個の場合は、上、左右、下、とみなされます。

パディングの広さの場合は、次の 4 個のプロパティを使うことによって、上下左右を個別に設定することができます。

```
padding-top    上のパディング。
padding-bottom 下のパディング。
padding-left   左のパディング。
padding-right  右のパディング。
```

次のスタイルシートは、段落のパディングに対して、上下左右で異なった広さを設定しています。

スタイルシートの例 padtrbl.css

```
p { font-size: 40px; color: #090; background-color: #ff9; }
#id000 { padding: 10px 20px 30px 40px; }
#id001 {
    padding-top: 40px;
    padding-right: 30px;
    padding-bottom: 20px;
    padding-left: 10px;
}
```

HTML 文書の例 padtrbl.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
```

```

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>パディングの上下左右</title>
<link rel="stylesheet" type="text/css" href="padtrbl.css" />
</head>
<body>
<p id="id000">この段落のパディングは、上が 10 ピクセル、
右が 20 ピクセル、下が 30 ピクセル、左が 40 ピクセルです。</p>
<p id="id001">この段落のパディングは、上が 40 ピクセル、
右が 30 ピクセル、下が 20 ピクセル、左が 10 ピクセルです。</p>
</body>
</html>

```

3.3 ボーダー

3.3.1 ボーダーの形状

ブラウザは、実線や破線や点線など、さまざまな形状でボーダーを表示することができます。

`border-style` というプロパティは、ボーダーの形状をあらわします。このプロパティに設定することのできる値としては、次のようなものがあります。

<code>none</code>	非存在。デフォルト。
<code>hidden</code>	<code>none</code> と同じ意味。
<code>solid</code>	実線。
<code>dashed</code>	破線。
<code>dotted</code>	点線。
<code>double</code>	二重線。
<code>groove</code>	陥没した線。
<code>ridge</code>	隆起した線。
<code>inset</code>	押されたボタンのように要素を見せる線。
<code>outset</code>	押されていないボタンのように要素を見せる線。

たとえば、

```
blockquote { border-style: solid; }
```

というルールを書くことによって、引用文のボーダーの形状として実線を設定することができます。

ボーダーの形状は、デフォルトでは `none` です。これは、ボーダーが存在しないという形状です。

次のスタイルシートと HTML 文書は、段落のボーダーに対してさまざまな形状を設定しています。

スタイルシートの例 `bostyle.css`

```

p {
  text-align: center;
  font-size: 30px;
  color: #009;
  background-color: #cfc;
  padding: 10px;
}
p.none { border-style: none; }
p.solid { border-style: solid; }
p.dashed { border-style: dashed; }
p.dotted { border-style: dotted; }
p.double { border-style: double; }
p.groove { border-style: groove; }
p.ridge { border-style: ridge; }
p.inset { border-style: inset; }
p.outset { border-style: outset; }

```

HTML 文書の例 `bostyle.html`

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>ボーダーの形状</title>
<link rel="stylesheet" type="text/css" href="bostyle.css" />
</head>
<body>
<p class="none">none</p>
<p class="solid">solid</p>
<p class="dashed">dashed</p>
<p class="dotted">dotted</p>
<p class="double">double</p>
<p class="groove">groove</p>
<p class="ridge">ridge</p>
<p class="inset">inset</p>
<p class="outset">outset</p>
</body>
</html>
```

3.3.2 ボーダーの幅

`border-width`というプロパティは、パディングエッジからボーダーエッジまでの距離をあらわします。ですから、このプロパティを使うことによって、ボーダーの幅を設定することができます。たとえば、

```
blockquote { border-width: 20px; }
```

というルールを書くことによって、引用文のパディングエッジからボーダーエッジまでの距離として20ピクセルを設定することができます。

`border-style`プロパティに`none`が設定されている場合、ボーダーは存在しないとみなされますので、たとえボーダーの幅が設定されていたとしても、その幅の無色透明なボーダーが存在するとみなされるわけではありません。

次のスタイルシートとHTML文書は、それぞれの段落のボーダーを異なる幅で表示します。

スタイルシートの例 `bowidth.css`

```
p {
    font-size: 30px;
    color: #090;
    background-color: #ffc;
    padding: 10px;
    border-style: solid;
}
#id000 { border-width: 10px; }
#id001 { border-width: 20px; }
#id002 { border-width: 30px; }
```

HTML文書の例 `bowidth.html`

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>ボーダーの幅</title>
<link rel="stylesheet" type="text/css" href="bowidth.css" />
</head>
<body>
<p id="id000">この段落のボーダーの幅は 10 ピクセルです。 </p>
<p id="id001">この段落のボーダーの幅は 20 ピクセルです。 </p>
<p id="id002">この段落のボーダーの幅は 30 ピクセルです。 </p>
</body>
</html>
```

3.3.3 ボーダーの色

ボーダーは、border-style プロパティに none が設定されている場合はまったく表示されませんが、それ以外の形状が設定されている場合は、何らかの色を使って、設定された幅を持つボーダーが表示されることになります。

border-color というプロパティは、ボーダーの色をあらわします。たとえば、

```
blockquote { border-color: orange; }
```

というルールを書くことによって、引用文のボーダーをオレンジ色で表示することができます。

border-color プロパティに色が設定されていない場合、ボーダーは、color プロパティに設定されている色を使って表示されます。

次のスタイルシートと HTML 文書は、それぞれの段落のボーダーを異なる色で表示します。

スタイルシートの例 bcolor.css

```
p {
  font-size: 30px;
  color: #009;
  background-color: white;
  padding: 10px;
  border-style: solid;
  border-width: 10px;
}
#id000 { border-color: #966; }
#id001 { border-color: #696; }
#id002 { border-color: #669; }
```

HTML 文書の例 bcolor.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>ボーダーの色</title>
<link rel="stylesheet" type="text/css" href="bcolor.css" />
</head>
<body>
<p id="id000">この段落のボーダーの色は#966 です。 </p>
<p id="id001">この段落のボーダーの色は#696 です。 </p>
<p id="id002">この段落のボーダーの色は#669 です。 </p>
</body>
</html>
```

3.3.4 ボーダーの一括設定

ボーダーの幅と色と形状は、それぞれのプロパティを使って別々に設定することも可能ですが、それらを一括して設定することも可能です。

border というプロパティは、ボーダーの幅と色と形状をあらわします。このプロパティに設定する値は、ボーダーの幅と形状と色を、空白で区切って、その順番で並べたものです。たとえば、

```
blockquote { border: 20px solid blue; }
```

というルールを書くことによって、引用文の周囲に、幅が 20 ピクセル、形状が実線、色が青のボーダーを表示することができます。

次のスタイルシートと HTML 文書は、それぞれの段落のボーダーを、異なる幅、異なる色、異なる形状で表示します。

スタイルシートの例 border.css

```
p {
  font-size: 30px;
  color: #900;
  background-color: white;
  padding: 20px;
}
```

	幅	形状	色
上	border-top-width	border-top-style	border-top-color
下	border-bottom-width	border-bottom-style	border-bottom-color
左	border-left-width	border-left-style	border-left-color
右	border-right-width	border-right-style	border-right-color

表 3.1: ボーダーの上下左右を個別に設定するプロパティ

```
#id000 { border: 2px solid #06f; }
#id001 { border: 4px dashed #6f0; }
#id002 { border: 8px dotted #f06; }
```

HTML 文書の例 border.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>ボーダーの一括設定</title>
<link rel="stylesheet" type="text/css" href="border.css" />
</head>
<body>
<p id="id000">幅が 2、形状が実線、色が#06f。</p>
<p id="id001">幅が 4、形状が破線、色が#6f0。</p>
<p id="id002">幅が 8、形状が点線、色が#f06。</p>
</body>
</html>
```

3.3.5 ボーダーの上下左右

ボーダーの幅や形状や色を上下左右で個別に設定したいときは、プロパティの値として、2個以上の幅、形状、色を、空白で区切って並べたものを書くか、または表 3.1 のプロパティを使います。

ちなみに、border というプロパティを使ってボーダーの上下左右を個別に設定する、ということではできません。

次のスタイルシートと HTML 文書は、上下左右で異なった形状のボーダーを持つ段落と、上下左右で異なった色のボーダーを持つ段落を表示します。

スタイルシートの例 bortrbl.css

```
p {
  font-size: 30px;
  color: #069;
  background-color: #9cf;
  padding: 20px;
}
#id000 {
  border-width: 6px;
  border-style: solid dashed dotted double;
}
#id001 {
  border-width: 10px;
  border-style: solid;
  border-top-color: red;
  border-right-color: green;
  border-bottom-color: blue;
  border-left-color: yellow;
}
```

HTML 文書の例 bortrbl.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>ボーダーの上下左右</title>
<link rel="stylesheet" type="text/css" href="bortrbl.css" />
</head>
<body>
<p id="id000">上が実線、右が破線、下が点線、左が二重線。 </p>
<p id="id001">上が赤、右が緑、下が青、左が黄色。 </p>
</body>
</html>
```

3.4 マージン

3.4.1 マージンの基礎

margin というプロパティは、ボーダーエッジからマージンエッジまでの距離をあらわします。ですから、このプロパティを使うことによって、マージンの広さを設定することができます。たとえば、

```
blockquote { margin: 20px; }
```

というルールを書くことによって、引用文のボーダーエッジからマージンエッジまでの距離として 20 ピクセルを設定することができます。

次のスタイルシートと HTML 文書は、段落のマージンをさまざまな広さに設定しています。

スタイルシートの例 margin.css

```
div {
background-color: #ccc;
padding: 0px;
border: 2px solid #f00;
margin: 10px;
}
p {
font-size: 40px;
color: white;
background-color: #00c;
padding: 4px;
}
#id000 { margin: 0px; }
#id001 { margin: 20px; }
#id002 { margin: 40px; }
```

HTML 文書の例 margin.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>マージン</title>
<link rel="stylesheet" type="text/css" href="margin.css" />
</head>
<body>
<div><p>マージンはデフォルトです。 </p></div>
<div><p id="id000">マージンは 0 ピクセルです。 </p></div>
<div><p id="id001">マージンは 20 ピクセルです。 </p></div>
<div><p id="id002">マージンは 40 ピクセルです。 </p></div>
</body>
</html>
```

3.4.2 マージンの上下左右

マージンの広さを上下左右で個別に設定したいときは、margin プロパティの値として、空白で区切って2個以上の長さを並べたものを書くか、または次の4個のプロパティを使います。

```
margin-top    上のマージン。
margin-bottom 下のマージン。
margin-left   左のマージン。
margin-right  右のマージン。
```

次のスタイルシートは、段落のマージンに対して、上下左右で異なった広さを設定しています。

スタイルシートの例 martrbl.css

```
div {
    background-color: #ccc;
    padding: 0px;
    border: 2px solid #00f;
    margin: 10px;
}
p {
    font-size: 40px;
    color: white;
    background-color: #090;
    padding: 4px;
}
#id000 { margin: 10px 20px 30px 40px; }
#id001 {
    margin-top: 40px;
    margin-right: 30px;
    margin-bottom: 20px;
    margin-left: 10px;
}
```

HTML 文書の例 martrbl.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>マージンの上下左右</title>
<link rel="stylesheet" type="text/css" href="martrbl.css" />
</head>
<body>
<div><p id="id000">上が 10、右が 20、下が 30、左が 40。</p></div>
<div><p id="id001">上が 40、右が 30、下が 20、左が 10。</p></div>
</body>
</html>
```

3.5 フローティング

3.5.1 フローティングの基礎

HTML の要素は、float というプロパティを持っていて、デフォルトでは、none という値がこのプロパティに設定されています。

値として left または right が設定されている float プロパティが適用された要素は、「フローティング要素」(floating element) と呼ばれます。

通常の要素とフローティング要素との相違点は、ブラウザがそれを表示する位置を決定する規則にあります。

通常の要素の場合、ブロックレベル要素は上から下に向かって並べられ、インライン要素は左から右に向かって並べられます。それに対して、フローティング要素の場合は、それがあかかも浮遊しているかのような考え方にもとづいて表示する位置が決定されます。

float プロパティに設定する left と right という値は、どちらも要素をフローティング要素にするという意味ですが、どちらの値が設定されているかということは、要素が表示される位置

に影響を与えます。leftの場合は要素が左寄りに表示され、rightの場合は要素が右寄りに表示されます。

3.5.2 回り込み

フローティング要素を作る目的のひとつは、テキストを要素の周囲に回り込ませることです。

フローティング要素が表示されている画面の上にテキストを表示すると、そのテキストは、フローティング要素の周囲に回り込んで表示されます。

次のスタイルシートとHTML文書は、フローティング要素が表示されている画面の上にテキストを表示するとどうなるかということを示しています。

スタイルシートの例 mawari.css

```
p {
  font-size: 40px;
  color: white;
  background-color: maroon;
  padding: 4px;
}
p.floatleft {
  font-size: 44px;
  color: blue;
  background-color: white;
  padding: 10px;
  margin: 10px;
  float: left;
}
```

HTML文書の例 mawari.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>回り込み</title>
<link rel="stylesheet" type="text/css" href="mawari.css" />
</head>
<body>
<p class="floatleft">フローティング要素</p>
<p>この段落の文字は、
フローティング要素の周囲に回り込んで表示されます。</p>
</body>
</html>
```

3.5.3 コンテントエッジの横の長さ

フローティング要素の周囲にテキストを回り込ませる場合には、そのフローティング要素に含まれているテキストの長さに注意する必要があります。テキストが長くなると、それにとまってフローティング要素の横の長さが伸びていきますので、その周囲にテキストが回り込む余地がなくなってしまうこともあるからです。

そのような事態を避けるための最善の方法は、要素のコンテントエッジに対して横の長さを設定しておくことです。そうしておけば、要素に含まれるテキストがいくら長くなったとしても、その要素の横の長さは常に一定に保たれます。

widthというプロパティは、コンテントエッジの横の長さをあらわします。たとえば、

```
blockquote { width: 500px; }
```

というルールを書くことによって、コンテントエッジの横の長さを常に500ピクセルにする、というスタイルを引用文に適用することができます。

次のスタイルシートとHTML文書は、コンテントエッジの横の長さを一定に保つフローティング要素を作っています。

スタイルシートの例 width.css

```
p {
  font-size: 40px;
```

```

    color: white;
    background-color: navy;
    padding: 4px;
}
p.floatright {
    font-size: 24px;
    color: green;
    background-color: white;
    width: 400px;
    padding: 10px;
    margin: 10px;
    float: right;
}

```

HTML 文書の例 width.html

```

<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>コンテンツエッジの横の長さ</title>
<link rel="stylesheet" type="text/css" href="width.css" />
</head>
<body>
<p class="floatright">コンテンツエッジの横の長さが
400 ピクセルのフローティング要素。</p>
<p>長いテキストを含む要素をフローティング要素にして、
その周囲にテキストを回り込ませたいときは、この例のように、
width プロパティにコンテンツエッジの横の長さを設定した
スタイルを、そのフローティング要素に適用する
必要があります。</p>
</body>
</html>

```

3.5.4 段組み

フローティング要素を作るもうひとつの目的は、段組みです。

「段組み」(multicolumn) というのは、ひとつのページを、水平方向に並んだ 2 個以上の長方形の領域に分割して、それぞれの領域にテキストを流し込むことです。段組みに使われるそれぞれの領域は、「段」(column) と呼ばれます。

フローティング要素を 2 個以上作ると、それらの要素は、文書の中に書かれている順番のとおり左から右へ向かって並びます。ですから、段組みは、それぞれの段をフローティング要素にすることによって実現することができます。

次のスタイルシートと HTML 文書は、テキストを 2 段組みで表示します。

スタイルシートの例 column.css

```

div {
    font-size: 40px;
    color: maroon;
    background-color: white;
    padding: 4px;
    border-style: solid;
    border-width: 3px;
    margin: 0px;
}
#column00 {
    width: 200px;
    border-color: blue;
    margin-right: 10px;
    float: left;
}
#column01 {
    width: 300px;
    border-color: green;
}

```

```
float: left;
}
```

HTML 文書の例 column.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>段組み</title>
<link rel="stylesheet" type="text/css" href="column.css" />
</head>
<body>
<div id="column00">
<p>ここは 1 段目です。</p>
<p>1 段目は、コンテンツエッジの横の長さが 200 ピクセルで、
ボダーの色が青色です。</p>
</div>
<div id="column01">
<p>ここは 2 段目です。</p>
<p>2 段目は、コンテンツエッジの横の長さが 300 ピクセルで、
ボダーの色が緑色です。</p>
</div>
</body>
</html>
```

3.6 テーブル

3.6.1 ボックスとしてのテーブル

CSS の観点から見ると、テーブルというのはひとつのボックスとして表示される要素です。そしてさらに、表を構成するそれぞれのセルもまたボックスとして表示されます。ですから、ボックスに関するスタイルは、テーブルとセルに対しても有効です。ただし、margin プロパティを使ってセルとセルとの間隔を設定することはできません。

次のスタイルシートは、table 要素、th 要素、td 要素に対してスタイルを適用しています。

スタイルシートの例 table.css

```
table {
  font-size: 30px;
  border: 5px solid #00f;
}
th {
  color: white;
  background-color: #00c;
  padding: 10px;
}
td {
  color: maroon;
  background-color: #cfc;
  border: 3px solid #0c0;
  padding: 10px;
}
```

HTML 文書の例 table.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>ボックスとしてのテーブル</title>
<link rel="stylesheet" type="text/css" href="table.css" />
</head>
<body>
```

```

<table>
<tr>
<th>谷山由紀夫</th><td>富山県</td><td>双子座</td>
</tr>
<tr>
<th>杉浦真里菜</th><td>宮崎県</td><td>牡牛座</td>
</tr>
</table>
</body>
</html>

```

3.6.2 垂直方向のアラインメント

テーブルのひとつの行を構成するセルの高さは、それらのセルの中で、内容の高さがもっとも高いものに統一されます。ですから、内容の高さがセルの高さよりも低い場合、その内容を上に寄せるか、下に寄せるか、それとも中央に配置するかということ、つまり垂直方向のアラインメントが重要な問題になります。

vertical-align というプロパティは、垂直方向のアラインメントをあらわします。このプロパティの値としては、top、bottom、middle などを書くことができます。top は上寄せ、bottom は下寄せ、middle は中央に配置するという意味です。たとえば、

```
td { vertical-align: bottom; }
```

というルールを書くことによって、セルの内容を下寄せで表示することができます。

次のスタイルシートと HTML 文書は、セルに対して上寄せ、下寄せ、中央配置のそれぞれを適用すると、どのように表示されるか、ということを示しています。

スタイルシートの例 valign.css

```

table { font-size: 30px; }
td {
  color: #066;
  background-color: #cff;
  border: 3px solid #0cc;
  padding: 10px;
}
td.top { vertical-align: top; }
td.bottom { vertical-align: bottom; }
td.middle { vertical-align: middle; }

```

HTML 文書の例 valign.html

```

<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>垂直方向のアラインメント</title>
<link rel="stylesheet" type="text/css" href="valign.css" />
</head>
<body>
<table>
<tr>
<td>高さが<br />とても<br />高い<br />セル</td>
<td class="top">上寄せ</td>
<td class="bottom">下寄せ</td>
<td class="middle">中央配置</td>
</tr>
</table>
</body>
</html>

```

3.7 リスト

3.7.1 マーカー

HTML には、リスト（箇条書き）を作るための要素型として、ul、ol、dl という三つの要素型があります。

ブラウザは、ul または ol を使って作られたリストを構成するそれぞれの項目（li という要素型の要素によって作られます）の左側に、「マーカー」（marker）と呼ばれる記号を表示します。

3.7.2 マーカーのタイプ

マーカーにはいくつかのタイプがあって、どのタイプを使うかということは、スタイルシートによって指定することができます。

list-style-type というプロパティは、リストのマーカーのタイプをあらわします。このプロパティには、マーカーのタイプをあらわす、次のような単語を設定します。

disc	塗りつぶされた円。
circle	線だけの円。
square	正方形。
decimal	1、2、3、4、5、...
upper-alpha	A、B、C、D、E、...
lower-alpha	a、b、c、d、e、...
upper-roman	I、II、III、IV、V、...
lower-roman	i、ii、iii、iv、v、...
none	マーカーを表示しない。

スタイルシートの例 marker.css

```
ul { list-style-type: circle; }
ul ul { list-style-type: square; }
ol { list-style-type: upper-roman; }
ol ol { list-style-type: lower-alpha; }
```

HTML 文書の例 marker.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>マーカーのタイプ</title>
<link rel="stylesheet" type="text/css" href="marker.css" />
</head>
<body>
<ul>
<li>お好み焼き</li>
<li>うどん
  <ul>
    <li>きつねうどん</li>
    <li>カレーうどん</li>
    <li>月見うどん</li>
  </ul>
</li>
<li>オムライス</li>
</ul>
<ol>
<li>序奏</li>
<li>ソナタ形式
  <ol>
    <li>提示部</li>
    <li>展開部</li>
    <li>再現部</li>
  </ol>
</li>
</ol>
```

```
<li>コーダ</li>
</ol>
</body>
</html>
```

3.7.3 マーカーの位置

マーカーは、デフォルトでは項目の外側に表示されますが、項目の内側に表示することも可能です。

`list-style-position` というプロパティは、マーカーの位置というスタイルをあらわします。このプロパティに対しては、マーカーの位置をあらわす、次のような単語を設定します。

`outside` 項目の外側。デフォルト。

`inside` 項目の内側。

スタイルシートの例 markpos.css

```
li {
  color: #060;
  background-color: #cfc;
  list-style-type: circle;
  margin: 10px;
  padding: 10px;
}
li.outside { list-style-position: outside; }
li.inside { list-style-position: inside; }
```

HTML 文書の例 markpos.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
<head>
<title>マーカーの位置</title>
<link rel="stylesheet" type="text/css" href="markpos.css" />
</head>
<body>
<ul>
<li class="outside">マーカーが外側にある項目</li>
<li class="inside">マーカーが内側にある項目</li>
</ul>
</body>
</html>
```

参考文献

[CSS,2006] Bert Bos, Tantek Çelik, Ian Hickson and Håkon Wium Lie (eds.), “Cascading Style Sheets, level 2 revision 1”, World Wide Web Consortium, 2006.

[Meyer,2004] Eric A. Meyer, *Cascading Style Sheets: The Definitive Guide, Second Edition*, O’Reilly Media, 2004, ISBN 0-596-00525-3. 邦訳 (株式会社クイープ)、『CSS 完全ガイド・第2版』、オライリー・ジャパン、2005、ISBN 4-87311-232-X。

[相原,2003] 相原哲哉、『一週間でマスターする CSS for Windows』、毎日コミュニケーションズ、2003、ISBN 4-8399-1268-8。

[エ・ビスコム,2003] エ・ビスコム・テック・ラボ、『スタイルシート・ステップアップアレンジブック——基本とそのバリエーションでマスターする CSS 活用術——』、毎日コミュニケーションズ、2003、ISBN 4-8399-1252-1。

[エ・ビスコム,2004] エ・ビスコム・テック・ラボ、『スタイルシート・スタンダード・デザインガイド——SEO / ユーザビリティ / アクセシビリティを考慮した実践的 HTML&CSS デザイン術——』、毎日コミュニケーションズ、2004、ISBN 4-8399-1501-6。

- [大原則,2006] 『プロとして恥ずかしくないスタイルシートの大原則』、エムディエヌ・ムック、エムディエヌコーポレーション、2006、ISBN 4-8443-5838-3。

索引

- .css (拡張子), 5
- @import 文, 7, 18
- 10 進数
 - による色の記述, 8
- 16 進数
 - による色の記述, 8
- a 要素, 14
- active 擬似クラス, 14
- background-color プロパティ, 4, 25
- bold (フォントの太さ), 20
- border プロパティ, 29
- border-bottom-color プロパティ, 30
- border-bottom-style プロパティ, 30
- border-bottom-width プロパティ, 30
- border-color プロパティ, 29
- border-left-color プロパティ, 30
- border-left-style プロパティ, 30
- border-left-width プロパティ, 30
- border-right-color プロパティ, 30
- border-right-style プロパティ, 30
- border-right-width プロパティ, 30
- border-style プロパティ, 27
- border-top-color プロパティ, 30
- border-top-style プロパティ, 30
- border-top-width プロパティ, 30
- border-width プロパティ, 28
- bottom (アラインメント), 36
- center (アラインメント), 21
- circle (マーカーのタイプ), 37
- class 属性, 12
- cm (単位), 10
- color プロパティ, 4, 29
- CSS, 3, 18
 - の MIME タイプ, 6
- cursive (フォント), 19
- dashed (ボーダーの形状), 27
- decimal (マーカーのタイプ), 37
- disc (マーカーのタイプ), 37
- d1 要素, 37
- DOM, 3
- dotted (ボーダーの形状), 27
- double (ボーダーの形状), 27
- em (単位), 11
 - によるフォントの大きさの記述, 11
- ex (単位), 11
- fantasy (フォント), 19
- first-letter 擬似要素, 15
- first-line 擬似要素, 15
- float プロパティ, 32
- font-family プロパティ, 19
- font-size プロパティ, 10
- font-style プロパティ, 20
- font-weight プロパティ, 20
- groove (ボーダーの形状), 27
- head 要素, 5, 6
- hidden (ボーダーの形状), 27
- hover 擬似クラス, 14
- href 属性, 5
- HTML, 3
- HTML 文書, 3
 - とスタイルシートとの関連付け, 5
- ID, 13
- id 属性, 13
- ID セレクタ, 13
- in (単位), 10
- inset (ボーダーの形状), 27
- inside (マーカーの位置), 38
- italic (フォントのスタイル), 20
- LaTeX, 3
- left (アラインメント), 21
- letter-spacing プロパティ, 11
- li 要素, 37
- line-height プロパティ, 22
- line-through (テキストの装飾), 24
- link 擬似クラス, 14
- link 要素, 5
- list-style-position プロパティ, 38
- list-style-type プロパティ, 37
- lower-alpha (マーカーのタイプ), 37
- lower-roman (マーカーのタイプ), 37
- LVHA, 14
- margin プロパティ, 31
- margin-bottom プロパティ, 32
- margin-left プロパティ, 32
- margin-right プロパティ, 32

- margin-top プロパティ, 32
- middle (アラインメント), 36
- MIME タイプ, 5, 6
 - CSS の——, 6
- mm (単位), 10
- monospace (フォント), 19

- left (フローティング), 32
- none (フローティング), 32
- right (フローティング), 32
- none (テキストの装飾), 24
- none (ボーダーの形状), 27
- none (マーカーのタイプ), 37
- normal (フォントのスタイル), 20
- normal (フォントの太さ), 20
- nowrap (折り返しの制御), 23

- oblique (フォントのスタイル), 20
- ol 要素, 37
- outset (ボーダーの形状), 27
- outside (マーカーの位置), 38
- overline (テキストの装飾), 24
- OWL, 3

- padding プロパティ, 25
- padding-bottom プロパティ, 26
- padding-left プロパティ, 26
- padding-right プロパティ, 26
- padding-top プロパティ, 26
- pc (単位), 10
- pt (単位), 10
- px (単位), 11

- RDF, 3
- rel 属性, 5
- ridge (ボーダーの形状), 27
- right (アラインメント), 21

- sans-serif (フォント), 19
- serif (フォント), 19
- solid (ボーダーの形状), 27
- square (マーカーのタイプ), 37
- style 要素, 6
- stylesheet, 6
- SVG, 3

- text-align プロパティ, 21
- text-decoration プロパティ, 24
- text-indent プロパティ, 22
- text/css, 6
- top (アラインメント), 36

- transparent (背景色), 25
- troff, 3
- type 属性, 5, 6

- ul 要素, 37
- underline (テキストの装飾), 24
- upper-alpha (マーカーのタイプ), 37
- upper-roman (マーカーのタイプ), 37

- vertical-align プロパティ, 36
- visited 擬似クラス, 14

- W3C, 3
- Web セーフカラー, 9
- white-space プロパティ, 23
- width プロパティ, 33

- x
 - の高さ, 11
- XHTML, 3
- XML, 3
- XML 応用言語, 3
- XSL, 3

- 値, 4
- アラインメント, 21
 - 垂直方向の——, 36
- アンカー, 14
 - 下線のない——, 24

- イタリック体, 20
- 位置
 - マーカーの——, 38
- 一括設定
 - ボーダーの——, 29
- 色, 7
 - の 10 進数による記述, 8
 - の 16 進数による記述, 8
 - ボーダーの——, 29
- 色名, 9
- インチ, 10
- インデント, 22
- インポート, 7
- インライン要素, 21

- ウェブページ, 3

- 大きさ
 - ピクセルの——, 11
 - フォントの——, 10, 11
- オブリーク体, 20

- 改ページ, 4
- 箇条書き, 37
- カスケード, 17
- 下線, 24

- の無いアンカー, 24
- 間隔
 - 文字の—, 11
- 関連付け
 - HTML 文書とスタイルシートとの—, 5
- 擬似クラス, 14
- 擬似クラスセレクタ, 14
- 擬似クラス名, 14
- 擬似要素, 15
- 擬似要素セレクタ, 15
- 行
 - の高さ, 22
 - 最初の—, 15
- 空白, 4
- クラスセレクタ, 12
- クラス名, 12
- グループ化, 16
- 形状
 - ボーダーの—, 27
- 継承, 16
- 構造
 - 文書の—, 3
 - コメントアウト, 5
 - コロン, 14, 15
 - コンテンツエッジ, 25
 - の横の長さ, 33
 - コンテンツエリア, 25
 - コンマ, 16
- 最初
 - の行, 15
 - の文字, 15
- 子孫セレクタ, 13
- シャープ, 13
- 上下左右
 - パディングの—, 26
 - ボーダーの—, 30
 - マージンの—, 32
 - 領域の—, 26
- 詳細度, 17
- 上線, 24
- 垂直方向
 - のアラインメント, 36
- スタイル
 - の適用, 5
 - フォントの—, 20
 - 文書の—, 3
 - スタイルシート, 3
 - と HTML 文書との関連付け, 5
 - の部品化, 7
 - スタイルシート言語, 3
 - スタイルシートファイル, 5
- 絶対単位, 10
- セリフ, 19
- セレクタ, 4, 12
- 前景色, 4
- 宣言, 4
- センチメートル, 10
- 装飾
 - テキストの—, 24
- 相対単位, 10, 11
- タイプ
 - マーカーの—, 37
- 高さ
 - x の—, 11
 - 行の—, 22
- タブ, 4
- 段, 34
- 段組み, 34
- 注釈, 5
- 長方形
 - ボックスを構成する—, 25
- テーブル
 - ボックスとしての—, 35
- テキスト
 - の装飾, 24
- 適用
 - スタイルの—, 5
- ドット, 12
- 取り消し線, 24
- パイカ, 10
- 背景色, 4, 25
- パディング, 25
 - の上下左右, 26
- パディングエッジ, 25
- 幅
 - ボーダーの—, 28
- 汎用フォントファミリ名, 19
- 光の三原色, 8
- ピクセル
 - の大きさ, 11
- 筆記体, 19
- フォント, 18
 - の大きさ, 10, 11
 - のスタイル, 20
 - の太さ, 20
- フォントファミリ, 18

フォントファミリ名, 18

復帰, 4

太さ

 フォントの——, 20

部品化

 スタイルシートの——, 7

フローティング要素, 32

ブロックレベル要素, 21

プロパティ, 4

プロポーショナル, 19

文書

 ——の構造, 3

 ——のスタイル, 3

ポイント, 10

ボーダー, 25

 ——の一括設定, 29

 ——の色, 29

 ——の上下左右, 30

 ——の形状, 27

 ——の幅, 28

ボーダーエッジ, 25

ボックス, 25

 ——としてのテーブル, 35

 ——を構成する長方形, 25

 ——を構成する領域, 25

ホワイトスペース, 4

マーカー, 37

 ——の位置, 38

 ——のタイプ, 37

マークアップ言語, 3

マージン, 25, 31

 ——の上下左右, 32

マージンエッジ, 25

回り込み, 33

ミリメートル, 10

文字

 ——の間隔, 11

 最初の——, 15

横の長さ

 コンテンツエッジの——, 33

リスト, 37

領域

 ——の上下左右, 26

 ボックスを構成する——, 25

ルール, 3

ローマン体, 20