

# CSS 実習マニュアル

第一版

2011 年 12 月 15 日 (木)

Copyright © 2007–2011 Daikoku Manabu

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

## 目次

<b>第 1 章</b>	<b>CSS の基礎</b>	<b>4</b>
1.1	CSS の基礎の基礎	4
1.1.1	文書の構造とスタイル	4
1.1.2	HTML とは何か	4
1.1.3	CSS とは何か	4
1.1.4	この文章について	4
1.2	スタイルシート	4
1.2.1	ルール	4
1.2.2	セレクタ	5
1.2.3	宣言ブロック	5
1.2.4	宣言	5
1.2.5	ホワイトスペース	5
1.2.6	注釈	6
1.3	スタイルの適用	6
1.3.1	スタイルシートの分類	6
1.3.2	外部スタイルシート	6
1.3.3	内部スタイルシート	7
1.3.4	インラインスタイルシート	8
1.3.5	スタイルシートの部品化	8
1.4	色	9
1.4.1	色の基礎	9
1.4.2	16 進数による色の記述	9
1.4.3	10 進数による色の記述	9
1.4.4	色名	10
1.4.5	ウェブセーフカラー	11
1.5	長さの単位	11
1.5.1	長さの単位の基礎	11
1.5.2	絶対単位	11
1.5.3	相対単位	12
1.5.4	em によるフォントの大きさの記述	12
1.6	継承	13
1.6.1	継承の基礎	13
1.6.2	相対単位による長さの継承	13
<b>第 2 章</b>	<b>セレクタ</b>	<b>14</b>
2.1	セレクタの基礎	14
2.1.1	セレクタについての復習	14
2.1.2	セレクタの種類	14
2.2	基本的なセレクタ	15
2.2.1	全称セレクタ	15
2.2.2	型セレクタ	15
2.2.3	クラスセレクタ	15
2.2.4	ID セレクタ	16
2.2.5	属性セレクタ	16

2.3	擬似クラスセレクタと擬似要素セレクタ	17
2.3.1	擬似クラスの基礎	17
2.3.2	擬似クラスセレクタ	17
2.3.3	ユーザーアクション擬似クラス	17
2.3.4	リンク擬似クラス	18
2.3.5	擬似要素セレクタ	18
2.4	複合セレクタとセレクタリスト	19
2.4.1	複合セレクタの基礎	19
2.4.2	子孫結合子	20
2.4.3	子供結合子	20
2.4.4	隣接兄弟結合子	21
2.4.5	一般兄弟結合子	21
2.4.6	セレクタリスト	22
2.5	詳細度	22
2.5.1	詳細度とは何か	22
2.5.2	詳細度の計算方法	22
<b>第3章</b>	<b>テキスト</b>	<b>23</b>
3.1	フォント	23
3.1.1	この章とこの節について	23
3.1.2	フォントファミリー	23
3.1.3	フォントの太さ	24
3.1.4	フォントのスタイル	25
3.2	テキストのレイアウト	26
3.2.1	この節について	26
3.2.2	テキストの水平方向のアラインメント	26
3.2.3	インデント	26
3.2.4	行の高さ	27
3.2.5	テキストの折り返しの抑制	27
3.3	テキストの装飾	28
3.3.1	下線と上線と取り消し線	28
3.3.2	テキストの影	29
<b>第4章</b>	<b>ボックス</b>	<b>29</b>
4.1	ボックスの基礎	29
4.1.1	ボックスとは何か	29
4.1.2	ボックスを構成する長方形	29
4.1.3	ボックスを構成する領域	29
4.1.4	背景色	30
4.2	ボックスの種類	30
4.2.1	ボックスの種類の基本	30
4.2.2	ブロックボックス	30
4.2.3	インラインボックス	30
4.3	パディング	31
4.3.1	パディングの基本	31
4.3.2	パディングの上下左右	32
4.4	ボーダー	32
4.4.1	ボーダーの基本	32
4.4.2	ボーダーの形状	33
4.4.3	ボーダーの幅	34
4.4.4	ボーダーの色	35
4.4.5	ボーダーの上下左右	35
4.4.6	角丸	36
4.5	マージン	37
4.5.1	マージンの基礎	37

目次	3
4.5.2 マージンの上下左右	38
4.5.3 マージンの相殺	39
4.6 ボックスの大きさ	39
4.6.1 ボックスの横の長さ	39
4.6.2 ボックスの縦の長さ	40
4.6.3 オーバーフロー	40
4.6.4 ボックスの水平方向のアラインメント	41
4.7 フローティング	42
4.7.1 フローティングの基礎	42
4.7.2 回り込み	42
4.7.3 回り込みの解除	43
4.7.4 段組	43
4.8 テーブル	44
4.8.1 ボックスとしてのテーブル	44
4.8.2 セルの間隔	45
4.8.3 結合ボーダーモデル	45
4.8.4 表示されない空セル	46
4.8.5 垂直方向のアラインメント	47
4.9 リスト	47
4.9.1 リストの基礎	47
4.9.2 マーカーのタイプ	47
4.9.3 マーカーの位置	48
4.9.4 リストアイテムボックス	49
参考文献	49
索引	51

## 第1章 CSSの基礎

### 1.1 CSSの基礎の基礎

#### 1.1.1 文書の構造とスタイル

文字を並べることによって何かを記述したものは、「文書」(document)と呼ばれます。

文書は、「構造」(structure)と呼ばれるものと「スタイル」(style)と呼ばれるものを持つことができます。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がどのように表示されるのかということです。

#### 1.1.2 HTMLとは何か

ウェブを構成している個々の文書は、「ウェブページ」(web page)と呼ばれます。

ウェブページの構造は、「タグ」(tag)と呼ばれる文字列をウェブページの中に埋め込むことによって記述されます。ウェブページの構造を記述するタグを作るための言語は、「HTML」(Hypertext Markup Language)と呼ばれます。ウェブページは、HTMLによってその構造が記述されていますので、「HTML文書」(HTML document)と呼ばれることもあります。

HTMLのような、文書の構造を記述するために文書の中に埋め込む文字列を定義している言語は、「マークアップ言語」(markup language)と呼ばれます。

HTMLは、W3C(World Wide Web Consortium)<sup>1</sup>という組織によって策定されている標準規格です。

#### 1.1.3 CSSとは何か

ウェブページのスタイルは、通常、「CSS」(Cascading Style Sheets)と呼ばれる言語によって記述されます。CSSを使って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。

CSSのような、文書のスタイルを記述するための言語は、「スタイルシート言語」(style sheet language)と呼ばれます。

CSSも、HTMLと同じように、W3Cによって策定されている標準規格です。

#### 1.1.4 この文章について

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

この文章は、読者は既にHTMLについて理解している、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

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; }
```

---

<sup>1</sup>URL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2章で、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」(declaration block)というのは、「宣言」(declaration)と呼ばれるものをひとつにまとめたもののことで、

```
{ 宣言 … }
```

というように、宣言を何個でも好きなだけ並べて、それらを左中括弧 (`{`) と右中括弧 (`}`) で囲むと、その全体がひとつの宣言ブロックになります。たとえば、

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

という宣言ブロックは、「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」という宣言をひとつにまとめたものです。

### 1.2.4 宣言

「宣言」(declaration)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、

```
body { background-color: orange; }
```

というルールと、

```
body {  
  background-color: orange;  
}
```

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（書いた本人も含まれます）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、注釈にすることによって無効にするという手段が、しばしば使われます。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。(comment out)と言います。逆に、コメントアウトされている部分を復活させることを、その部分を「コメントインする」(comment in)と言います。

## 1.3 スタイルの適用

### 1.3.1 スタイルシートの分類

スタイルシートは、「それがどこに書かれているか」という観点から、次の3種類に分類することができます。

- 外部スタイルシート (external style sheet)
- 内部スタイルシート (internal style sheet)
- インラインスタイルシート (inline style sheet)

### 1.3.2 外部スタイルシート

HTML文書とは別のファイルに格納されているスタイルシートは、「外部スタイルシート」(external style sheet)と呼ばれます。

ひとつの外部スタイルシートは、複数のHTML文書に適用することができます。ですから、ウェブサイトを構成しているそれぞれのHTML文書のスタイルを統一したいときは、外部スタイルシートが使われます。

外部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」(style sheet 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。

外部スタイルシートをHTML文書に適用したいときは、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。

`href` 関連付けるファイルのURL。

スタイルシートを HTML 文書に適用する場合、link 要素の rel 属性には、stylesheet という単語を設定します。

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" href="namako.css">
```

という link 要素を書くことによって、namako.css をそれらの HTML 文書に適用することができます。

それでは、実際に試してみましょう。次のスタイルシートと HTML 文書のそれぞれを、同じディレクトリにある別々のファイルに格納して、そして HTML 文書のファイルをブラウザで開いてみてください。そうすると、背景がオレンジ色のページが表示されるはずです。

スタイルシートの例 external.css

---

```
body { background-color: orange; }
```

---

HTML 文書の例 external.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>外部スタイルシート</title>
    <link rel="stylesheet" href="link.css">
  </head>
  <body></body>
</html>
```

---

### 1.3.3 内部スタイルシート

HTML の head 要素の中には、style という要素型の要素を書くことができます。style 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 要素の中に書かれたスタイルシートは、「内部スタイルシート」(internal style sheet) と呼ばれます。

たとえば、HTML 文書の head 要素の中に、

```
<style>
  p { color: blue; }
</style>
```

という style 要素を書くことによって、その HTML 文書の中にあるすべての p 要素の文字をブラウザに青色で表示させることができます。

内部スタイルシートが適用される範囲は、それが書かれている HTML 文書の中だけに限定されます。ですから、内部スタイルシートは、特定の HTML 文書だけに適用したいスタイルを記述するために使われます。

次の HTML 文書の中の p 要素の文字は、青色で表示されます。

HTML 文書の例 internal.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>内部スタイルシート</title>
    <style>
      p { color: blue; }
    </style>
  </head>
  <body>
    <p>私は段落です。</p>
    <p>私も段落です。</p>
  </body>
</html>
```

---

### 1.3.4 インラインスタイルシート

HTMLの要素は、`style`という属性を持っています。`style`属性に対して、CSSの宣言の列を属性値として設定すると、それによって記述されたスタイルが、その要素に適用されます。`style`属性に設定された宣言の列は、「インラインスタイルシート」(inline style sheet)と呼ばれます。

たとえば、HTML文書の中に書かれた、

```
<p style="color: maroon;">私は段落です。</p>
```

というp要素の文字は、栗色で表示されます。

インラインスタイルシートが適用される範囲は、その属性を持っている要素だけに限定されません。ですから、インラインスタイルシートは、特定のHTML文書の中の特定の要素だけに適用したいスタイルを記述するために使われます。

次のHTML文書の中のp要素は、それぞれの要素の`style`属性で指定された前景色と背景色で表示されます。

HTML文書の例 `inline.html`

```
<!DOCTYPE html>
<html>
  <head><title>インラインスタイルシート</title></head>
  <body>
    <p style="color: maroon; background-color: silver;">
      私は段落です。
    </p>
    <p style="color: yellow; background-color: green;">
      私も段落です。
    </p>
  </body>
</html>
```

### 1.3.5 スタイルシートの部品化

外部スタイルシートと内部スタイルシートは、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、それを自分の一部分にする、という機能を持っています。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」(@import statement)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。

次のスタイルシートとHTML文書は、body要素の背景を銀色で表示して、p要素の文字を栗色で表示します。p要素の文字を栗色にするスタイルは、`module.css`から`import.css`へインポートされたものです。

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-color: silver; }
```

HTML文書の例 `import.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>スタイルシートの部品化</title>
```

```

<link rel="stylesheet" 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

```

<!DOCTYPE html>
<html>
  <head>
    <title>16 進数による色の記述</title>
    <link rel="stylesheet" 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`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>10進数による色の記述</title>
    <link rel="stylesheet" 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147個の単語を使うことができます。次の表は、それらの色名のうちの主要なものについて、その意味を16進数で記述したものです。

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`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>色名</title>
    <link rel="stylesheet" href="colword.css">
  </head>
  <body>
    <h1>blue</h1>
    <h2>aqua</h2>
    <h3>teal</h3>
  </body>
</html>
```

---

### 1.4.5 ウェブセーフカラー

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、それらの色は、どのような環境でも正しく表示されるとは限りません。しかし、「ウェブセーフカラー」(web safe color) と呼ばれる 216 の色だけは、どのような環境でも正しく表示されることが保障されています。

ウェブセーフカラーは、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。たとえば、#ffff00、#339966、#00ffcc などはウェブセーフカラーです。

すべてのウェブセーフカラーは、それぞれの原色の強さを 1 桁の 16 進数で書くことができます。たとえば、#99ccff というウェブセーフカラーは、#9cf と書くことができます。

次のスタイルシートは、見出しの色としてウェブ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

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ウェブセーフカラー</title>
    <link rel="stylesheet" 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、それをあらわす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、次のような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`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>絶対単位</title>
    <link rel="stylesheet" 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、次のような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。

`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、`em`という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`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>相対単位</title>
    <link rel="stylesheet" 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: 4em; }
h2 { font-size: 2em; }
h3 { font-size: 1em; }
```

---

HTML 文書の例 `fontsize.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>em によるフォントの大きさの記述</title>
    <link rel="stylesheet" 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 継承

### 1.6.1 継承の基礎

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、その子供や孫など、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。ただし、すべてのプロパティが子孫に継承されるわけではありません。継承されないプロパティというも存在します。

次のスタイルシートは、フォントの大きさを 40px にするというスタイルを `body` 要素に適用しています。このスタイルは、`body` 要素の子孫にも継承されますので、`body` 要素の子供の `p` 要素も、そのさらに子供の `em` 要素も、フォントの大きさは 40px になります。

スタイルシートの例 `inherit.css`

---

```
body { font-size: 40px; }
```

---

HTML 文書の例 `inherit.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>継承</title>
    <link rel="stylesheet" href="inherit.css">
  </head>
  <body>
    <p>吾輩は<em>段落</em>である。</p>
  </body>
</html>
```

---

### 1.6.2 相対単位による長さの継承

スタイルが相対単位による長さで記述されている場合、継承されるのは、相対単位による長さではなくて、その長さから算出された絶対単位による長さです。

次のスタイルシートでは、`p` 要素のフォントの大きさが 2em という相対単位による長さで記述されています。

スタイルシートの例 `relinhe.css`

---

```
body { font-size: 10px; }
p { font-size: 2em; }
```

---

HTML 文書の例 `relinhe.html`

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>相対単位の継承</title>
    <link rel="stylesheet" href="relinhe.css">
  </head>
  <body>
    <p>吾輩は<em>段落</em>である。</p>
  </body>
</html>

```

このHTML文書をブラウザに表示させると、`em`要素の文字は、`p`要素の文字と同じ大きさ(20px)で表示されます。もしも、`p`要素から`em`要素へ継承されるのが`2em`という相対単位による長さだとするならば、`em`要素の文字の大きさは、10pxの2倍のさらに2倍で、40pxになるはずですが、そのようにはなりません。なぜなら、この場合に継承されるのは、`2em`という相対単位による長さではなくて、20pxという絶対単位による長さだからです。

## 第2章 セレクタ

### 2.1 セレクタの基礎

#### 2.1.1 セレクタについての復習

この章では、セレクタについて説明したいと思います。

「セレクタ」(selector)というのは、第1.2.2項で簡単に説明したように、スタイルを適用する対象を記述したもののことです。

スタイルシートは、「ルール」(rule)と呼ばれる、

```

セレクタ 宣言ブロック

```

という形のものから構成されます。「宣言ブロック」(declaration block)は、スタイルを記述するところです。そして、宣言ブロックで記述されたスタイルをどこに適用するのかということも記述しているのがセレクタです。

要素型名は、その要素型を持つすべての要素に対してスタイルを適用するセレクタです。たとえば、

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

というルールは、セレクタが`em`という要素型名ですので、文字の色を赤色にするというスタイルをすべての`em`要素に適用します。

#### 2.1.2 セレクタの種類

セレクタには、次のような種類があります。

- 全称セレクタ (universal selector)
- タイプセレクタ (type selector)
- クラスセレクタ (class selector)
- IDセレクタ (ID selector)
- 属性セレクタ (attribute selector)
- 擬似クラスセレクタ (pseudo-class selector)
- 擬似要素セレクタ (pseudo-element selector)
- 複合セレクタ (complex selector)
- セレクタリスト (selector list)

全称セレクタ、タイプセレクタ、クラスセレクタ、IDセレクタ、属性セレクタについては第2.2節で、擬似クラスセレクタと擬似要素セレクタについては第2.3節で、複合セレクタとセレクタリストについては第2.4節で説明することにしたと思います。

## 2.2 基本的なセレクタ

### 2.2.1 全称セレクタ

すべての要素に対してスタイルを適用するセレクタは、「全称セレクタ」(universal selector)と呼ばれます。

全称セレクタは、1文字のアスタリスク (asterisk) です。たとえば、

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

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

スタイルシートの例 `universal.css`

---

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

---

HTML 文書の例 `universal.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>全称セレクタ</title>
    <link rel="stylesheet" href="universal.css">
  </head>
  <body>
    <h1>私は見出しです。</h1>
    <p>私は段落です。</p>
    <pre>私は整形済みテキストです。</pre>
  </body>
</html>
```

---

### 2.2.2 型セレクタ

第 1.2.2 項で説明したように、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スタイルを適用することができます。このようなセレクタは、「型セレクタ」(type selector)と呼ばれます。

### 2.2.3 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、

```
.tokumori
```

というセレクタを書くことによって、`tokumori` というクラス名が `class` 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 `class.css`

---

```
.udon { color: red; }
.tokumori { background-color: #9ff; }
```

---

HTML 文書の例 `class.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>クラスセレクタ</title>
    <link rel="stylesheet" href="class.css">
  </head>
  <body>
    <p class="udon">きつねうどん</p>
```

```

    <p class="tokumori">特盛天ぶらそば</p>
    <p class="tokumori udon">特盛カレーうどん</p>
  </body>
</html>

```

---

### 2.2.4 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

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>ID セレクタ</title>
    <link rel="stylesheet" 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>
    <p id="id002">ID が id002 の段落</p>
  </body>
</html>

```

---

### 2.2.5 属性セレクタ

特定の属性を持つ要素に対してスタイルを適用するセレクタは、「属性セレクタ」(attribute selector) と呼ばれます。

属性値が何であるにかかわらず、特定の属性を持つ要素に対してスタイルを適用する属性セレクタは、属性名を角括弧 (square bracket) で囲んだもの、つまり、

```
[属性名]
```

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、角括弧の中に書かれた名前の属性を持つ要素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、

```
[onclick]
```

というセレクタを書くことによって、onclick という名前の属性を持っ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。

特定の属性値が設定された属性を持っている要素だけに対してスタイルを適用することも可能です。そのような場合は、

```
[属性名]="属性値"]
```

という形の記述を書きます。たとえば、

```
[onclick="alert('hoge')"]
```

というセレクタを書くことによって、`alert('hoge')`という属性値が設定された `onclick` という属性を持っ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 `attribute.css`

---

```
[onclick] { background-color: #9ff; }
[onclick="alert('hoge')"] { color: red; }
```

---

HTML 文書の例 `attribute.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>属性セレクタ</title>
    <link rel="stylesheet" href="attribute.css">
  </head>
  <body>
    <p>onclick 属性を持たない段落</p>
    <p onclick="null">onclick 属性を持つ段落</p>
    <p onclick="alert('hoge')">
      alert('hoge') が設定された onclick 属性を持つ段落
    </p>
  </body>
</html>
```

---

## 2.3 擬似クラスセレクタと擬似要素セレクタ

### 2.3.1 擬似クラスの基礎

CSS は、要素の状態や特性などを擬似的なクラスとみなして、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。そのような擬似的なクラスは、「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。たとえば、`hover` という名前は、マウスポインタが重なっているという状態を示す擬似クラスを識別する擬似クラス名です。

### 2.3.2 擬似クラスセレクタ

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、

```
p:hover
```

というセレクタを書くことによって、マウスポインタが重なっている段落に対してスタイルを適用することができます。

### 2.3.3 ユーザーアクション擬似クラス

ユーザーによって操作されている状態を示す擬似クラスは、「ユーザーアクション擬似クラス」(user action pseudo-class) と呼ばれます。

ユーザーアクション擬似クラスとしては、次の三つのものがあります。

`hover` マウスポインタが重なっている状態。

`active` アクティブ化された状態。

`focus` フォーカスが当たっている状態。

スタイルシートの例 `action.css`

---

```
#id000:hover { background-color: #99f; }
#id001:active { background-color: #f99; }
#id002:focus { background-color: #ff9; }
#id003:focus { background-color: #9ff; }
```

---

#### HTML 文書の例 action.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ユーザーアクション擬似クラスセレクタ</title>
    <link rel="stylesheet" href="action.css">
  </head>
  <body>
    <p id="id000">私の上にマウスポインタを重ねてください。</p>
    <p id="id001">私をクリックしてください。</p>
    <form>
      <p><input id="id002" type="text"></p>
      <p><input id="id003" type="text"></p>
    </form>
  </body>
</html>
```

---

### 2.3.4 リンク擬似クラス

a 要素、つまりアンカー (anchor) は、リンク先をまだ訪問していない状態と、すでに訪問した状態のそれぞれを示す擬似クラスを持っています。アンカーの状態を示すこれらの擬似クラスは、「リンク擬似クラス」(link pseudo-class) と呼ばれます。

リンク擬似クラスとしては、次の二つのものがあります。

**link**      リンク先をまだ訪問していない状態。

**visited**    リンク先をすでに訪問した状態。

アンカーの擬似クラスに対してスタイルを適用するルールは、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 anchor.css

---

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

---

#### HTML 文書の例 anchor.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>リンク擬似クラスセレクタ</title>
    <link rel="stylesheet" href="anchor.css">
  </head>
  <body>
    <p><a href="http://www.kantei.go.jp/">www.kantei.go.jp</a></p>
    <p><a href="http://www.jaxa.jp/">www.jaxa.jp</a></p>
  </body>
</html>
```

---

### 2.3.5 擬似要素セレクタ

CSS は、HTML の要素ではないものを擬似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。そのような擬似的な要素は、「擬似要素」(pseudo-element) と呼ばれます。擬似要素は、「擬似要素名」(pseudo-element name) と呼ばれる名前によって識別されます。

擬似要素としては、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、セレクタの右側に二つのコロン (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、二つの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、二つの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::first-letter
```

というセレクタを書くことによって、すべての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。

スタイルシートの例 `first.css`

```
#id000::first-letter {
    font-size: 80px;
    color: white;
    background-color: green;
}
#id001::first-line {
    font-size: 80px;
    color: white;
    background-color: maroon;
}
```

HTML 文書の例 `first.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>擬似要素セレクタ</title>
    <link rel="stylesheet" href="first.css">
  </head>
  <body>
    <p id="id000">
      この段落は、最初の文字だけが、
      それ以外の文字とは異なるスタイルで表示されます。
    </p>
    <p id="id001">
      この段落は、最初の行だけが、
      それ以外の行とは異なるスタイルで表示されます。
    </p>
  </body>
</html>
```

## 2.4 複合セレクタとセレクタリスト

### 2.4.1 複合セレクタの基礎

結合子 (combinator) と呼ばれる文字によって結合されたセレクタは、「複合セレクタ」(complex selector) と呼ばれます。

結合子としては、次のようなものがあります。

- 子孫結合子 (descendant combinator)
- 子供結合子 (child combinator)
- 隣接兄弟結合子 (adjacent sibling combinator)
- 一般兄弟結合子 (general sibling combinator)

### 2.4.2 子孫結合子

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したいときは、「子孫結合子」(descendant combinator) と呼ばれる結合子を使います。

ホワイトスペースを挟んで二つのセレクタを並べると、そのホワイトスペースは子孫結合子になります。子孫結合子によって作られた複合セレクタは、その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、その右に書かれた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します。たとえば、

```
#id000 p
```

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 `p` 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 `descendant.css`

```
#id000 p { background-color: yellow; }
#id001 p { background-color: aqua; }
```

HTML 文書の例 `descendant.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>子孫結合子</title>
    <link rel="stylesheet" href="descendant.css">
  </head>
  <body>
    <div id="id000">
      <p>私は子供です。</p>
      <div>
        <p>私は孫です。</p>
      </div>
    </div>
    <div id="id001">
      <p>私は子供です。</p>
      <div>
        <p>私は孫です。</p>
      </div>
    </div>
  </body>
</html>
```

### 2.4.3 子供結合子

指定された要素の子供になっている要素だけに対してスタイルを適用したいときは、「子供結合子」(child combinator) と呼ばれる結合子を使います。

子供結合子は、大なり (greater than) という文字です。子供結合子を挟んで二つのセレクタを並べることによって作られた複合セレクタは、その左に書かれたセレクタによって指定された要素の子供になっている要素のうちで、その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、スタイルを適用します。たとえば、

```
body > p
```

というセレクタを書くことによって、`body` 要素の子供になっている `p` 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 `child.css`

```
body > p { background-color: #9f9; }
```

HTML 文書の例 `child.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>子供結合子</title>
    <link rel="stylesheet" href="child.css">
  </head>
```

```

<body>
  <p>私は子供です。</p>
  <div>
    <p>私は孫です。</p>
  </div>
</body>
</html>

```

---

#### 2.4.4 隣接兄弟結合子

指定された要素の直後にある要素だけに対してスタイルを適用したいときは、「隣接兄弟結合子」(adjacent sibling combinator) と呼ばれる結合子を使います。

隣接兄弟結合子は、プラス (plus) という文字です。隣接兄弟結合子を挟んで二つのセレクタを並べることによって作られた複合セレクタは、その左に書かれたセレクタによって指定された要素の直後に、その右に書かれたセレクタによって指定される要素があって、それらの親が共通の場合、後者の要素だけに対してスタイルを適用します。たとえば、

```
h1 + p
```

というセレクタを書くことによって、h1 要素の直後にある、それと共通の親を持つ p 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 adjacent.css

```
h1 + p { background-color: #f99; }
```

---

HTML 文書の例 adjacent.html

```

<!DOCTYPE html>
<html>
  <head>
    <title>隣接兄弟結合子</title>
    <link rel="stylesheet" href="adjacent.css">
  </head>
  <body>
    <h1>第一レベルの見出し</h1>
    <p>私は第一レベルの見出しの直後にある段落です。</p>
    <p>私はその次にある段落です。</p>
  </body>
</html>

```

---

#### 2.4.5 一般兄弟結合子

指定された要素の後ろにある要素だけに対してスタイルを適用したいときは、「一般兄弟結合子」(general sibling combinator) と呼ばれる結合子を使います。

一般兄弟結合子は、チルダ (tilde) という文字です。一般兄弟結合子を挟んで二つのセレクタを並べることによって作られた複合セレクタは、その左に書かれたセレクタによって指定された要素の後ろに、その右に書かれたセレクタによって指定される要素があって、それらの親が共通の場合、後者の要素だけに対してスタイルを適用します。たとえば、

```
#id000 ~ p
```

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後ろにある、それと共通の親を持つ p 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 general.css

```
#id000 ~ p { background-color: #99f; }
```

---

HTML 文書の例 general.html

```

<!DOCTYPE html>
<html>
  <head>
    <title>一般兄弟結合子</title>
    <link rel="stylesheet" href="general.css">
  </head>
  <body>

```

```

<div>
  <h2 id="id000">ID が id000 の見出し</h2>
  <p>私は段落です。</p>
  <p>私も段落です。</p>
</div>
<div>
  <p>私も段落ですが、親が違います。</p>
</div>
</body>
</html>

```

---

### 2.4.6 セレクタリスト

複数のセレクタによって記述されるそれぞれの対象に対して、同一のスタイルを適用したいときは、「セレクタリスト」(selector list) と呼ばれるセレクタを使います。

セレクタリストは、セレクタをコンマ (comma) で区切って並べたものです。

セレクタリストは、それを構成しているそれぞれのセレクタによって記述された対象に対して同一のスタイルを適用します。たとえば、

```
h1, h2, h3
```

というセレクタを書くことによって、h1 要素と h2 要素と h3 要素に対して同一の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 list.css

```
h1, h2, h3 { background-color: lime; }
```

---

HTML 文書の例 list.html

```

<!DOCTYPE html>
<html>
  <head>
    <title>セレクタリスト</title>
    <link rel="stylesheet" href="list.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>
    <h4>第四レベルの見出し</h4>
  </body>
</html>

```

---

## 2.5 詳細度

### 2.5.1 詳細度とは何か

特定の要素を指定するセレクタは、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" id="answer">付録 A : 練習問題の解答</h2>
```

という要素は、h2 という型セレクタによって指定することもできますし、.appendix というクラスセレクタによって指定することもできますし、#answer という ID セレクタによって指定することもできます。競合する複数のセレクタが存在する場合、スタイルが適用されるのは、それらのうちで詳細度がもっとも高いものです。

「詳細度」(specificity) というのは、セレクタのそれぞれが持っている優先順位のことです。詳細度が高いセレクタほど、優先的にスタイルが適用されます。

詳細度は、そのセレクタによって指定される範囲が狭いほど高くなります。たとえば、型セレクタよりもクラスセレクタのほうが詳細度が高くなりますし、ID セレクタはそれらよりもさらに詳細度が高くなります。

### 2.5.2 詳細度の計算方法

詳細度は、4 桁の数によってあらわされます (ただし、それは 10 進数ではなくて、基数が無限大の数です)。その数があらわしている数値が大きいほど、詳細度が高くなります。

詳細度をあらかず数の桁を、左から順番に a、b、c、d とすると、それぞれの桁は、次のように求めることができます。

- a 宣言がインラインスタイルシートに書かれているならば 1、そうでなければ 0。
- b セレクタに含まれる ID セレクタの個数。
- c セレクタに含まれるクラスセレクタと擬似クラス名の個数。
- d セレクタに含まれる要素型名と擬似要素名の個数。

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 `specificity.css`

---

```
p { background-color: lime; }
.deluxe { background-color: aqua; }
#id000 { background-color: yellow; }
#id001 { background-color: orange; }
```

---

HTML 文書の例 `specificity.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>詳細度</title>
    <link rel="stylesheet" href="specificity.css">
  </head>
  <body>
    <p>普通の段落</p>
    <p class="deluxe">deluxe クラスの段落</p>
    <p class="deluxe" id="id000">
      deluxe クラスで、ID が id000 の段落
    </p>
    <p class="deluxe" id="id001" style="background-color: silver;">
      deluxe クラスで、ID が id001 で、
      インラインスタイルシートを持つ段落
    </p>
  </body>
</html>
```

---

## 第3章 テキスト

### 3.1 フォント

#### 3.1.1 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。

#### 3.1.2 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となります。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を使って要素を表示します。

CSS は、「汎用フォントファミリ名」(generic font family name) と呼ばれるフォントファミリ名を定義しています。汎用フォントファミリ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名は、次の五つです。

<code>serif</code>	プロポーションで、かつセリフを持つフォントファミリ。「プロポーション」(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。
<code>sans-serif</code>	プロポーションで、かつセリフを持たないフォントファミリ。
<code>monospace</code>	プロポーションではない、つまり文字の横幅が一定のフォントファミリ。
<code>cursive</code>	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。
<code>fantasy</code>	ほかのフォントファミリには分類されないフォントファミリ。

次のスタイルシートと HTML 文書は、汎用フォントファミリ名を使ってフォントファミリを指定した場合に、どのようなフォントファミリが選択されるか、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; }
.serif { font-family: serif; }
.sans-serif { font-family: sans-serif; }
.monospace { font-family: monospace; }
.cursive { font-family: cursive; }
.fantasy { font-family: fantasy; }
```

---

HTML 文書の例 `family.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>フォントファミリ</title>
    <link rel="stylesheet" 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>
```

---

### 3.1.3 フォントの太さ

ひとつのフォントファミリは、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; }
.normal { font-weight: normal; }
.bold { font-weight: bold; }
```

---

HTML 文書の例 `weight.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>フォントの太さ</title>
    <link rel="stylesheet" href="weight.css">
  </head>
  <body>
    <p class="normal">normal</p>
    <p class="bold">bold</p>
  </body>
</html>
```

---

### 3.1.4 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になります。

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; }
.normal { font-style: normal; }
.oblique { font-style: oblique; }
.italic { font-style: italic; }
```

---

HTML 文書の例 `style.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>フォントのスタイル</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <p class="normal">roman</p>
    <p class="oblique">oblique</p>
    <p class="italic">italic</p>
```

---

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

---

## 3.2 テキストのレイアウト

### 3.2.1 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。

### 3.2.2 テキストの水平方向のアラインメント

要素が表示される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は、テキストの「アラインメント」(alignment)と呼ばれます。

`text-align` というプロパティは、テキストの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; }
.left { text-align: left; }
.right { text-align: right; }
.center { text-align: center; }
```

---

HTML 文書の例 `align.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>テキストの水平方向のアラインメント</title>
    <link rel="stylesheet" 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>
```

---

### 3.2.3 インデント

テキストが複数の行に分割されているとき、その中の特定の行について、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、要素に対して、テキストの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、テキストの1行目を右方向へインデントする長さをあらわします (値としてマイナスの長さ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)。長さは、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大きさを基準とする相対的な長さともみなされます。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、フォントの大きさの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、フォントの大きさと同じ長さ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: 40pt; text-indent: 1em; }
```

---

HTML 文書の例 `indent.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>インデント</title>
    <link rel="stylesheet" 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。ですから、日本語のウェブサイトを
      作る場合は、text-indent プロパティを 1em にするスタイルを
      p 要素に適用するといいでしょう。
    </p>
  </body>
</html>
```

---

### 3.2.4 行の高さ

CSS では、ブロックレベル要素に対して、行の高さ、つまり行送りの長さ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大きさを基準とする相対的な長さともみなされます。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、フォントの大きさを 1.5 倍した長さ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、フォントの大きさを 3 倍した長さを設定しています。

スタイルシートの例 `height.css`

---

```
p { font-size: 40pt; line-height: 3em; }
```

---

HTML 文書の例 `height.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>行の高さ</title>
    <link rel="stylesheet" href="height.css">
  </head>
  <body>
    <p>
      この段落には、行の高さとして、フォントの大きさを 3 倍した
      長さが設定されています。ですから、行と行とのあいだには、
      フォントの大きさを 2 倍した長さの空白ができるはずです。
    </p>
  </body>
</html>
```

---

### 3.2.5 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: 40px; }
.nowrap { white-space: nowrap; }
```

HTML文書の例 nowrap.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>テキストの折り返しの抑制</title>
    <link rel="stylesheet" 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。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>
```

### 3.3 テキストの装飾

#### 3.3.1 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とHTML文書は、下線、上線、取り消し線のそれぞれが引かれた段落を表示します。

スタイルシートの例 decoration.css

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

HTML文書の例 decoration.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>下線と上線と取り消し線</title>
    <link rel="stylesheet" href="decoration.css">
  </head>
  <body>
    <p class="underline">私には下線が引かれています。</p>
    <p class="overline">私には上線が引かれています。</p>
```

```

    <p class="line-through">私には取り消し線が引かれています。</p>
  </body>
</html>

```

---

### 3.3.2 テキストの影

`text-shadow` というプロパティは、テキストの影というスタイルをあらわします。このプロパティには、値として、次の四つのもをホワイトスペースで区切って並べたものを書きます。

右方向の位置 テキストの位置から右方向へどれだけ移動した位置に影を表示するか。

下方向の位置 テキストの位置から下方向へどれだけ移動した位置に影を表示するか。

ぼかし幅 影のぼかし幅。

影の色 影を表示する色。

たとえば、

```
p { text-shadow: 10px 8px 6px gray; }
```

というルールを書くことによって、段落のテキストに対して、右へ 10 ピクセル、下へ 8 ピクセルだけ移動した位置に、ぼかし幅が 6 ピクセルで色が灰色の影を付けることができます。

次のスタイルシートと HTML 文書は、影の付いた段落を表示します。

スタイルシートの例 `textshadow.css`

```
p { font-size: 40px; text-shadow: 10px 8px 6px gray; }
```

---

HTML 文書の例 `textshadow.html`

```

<!DOCTYPE html>
<html>
  <head>
    <title>テキストの影</title>
    <link rel="stylesheet" href="textshadow.css">
  </head>
  <body>
    <p>私には影があります。</p>
  </body>
</html>

```

---

## 第4章 ボックス

### 4.1 ボックスの基礎

#### 4.1.1 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。

#### 4.1.2 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。

#### 4.1.3 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。

#### 4.1.4 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。

## 4.2 ボックスの種類

### 4.2.1 ボックスの種類の基本

ボックスには、さまざまな種類があります。HTMLの要素は、それぞれの要素型ごとにデフォルトのボックスの種類が決まっていますが、CSSを使うことによって、ボックスの種類を変更することも可能です。

`display`というプロパティは、ボックスの種類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、たとえば次のようなものを書くことができます。

`block`          ブロックボックス。  
`inline`        インラインボックス。  
`list-item`    リストアイテムボックス。

この節では、ブロックボックスとインラインボックスについて説明します。リストアイテムボックスについては、第4.9節で説明することにしたと思います。

### 4.2.2 ブロックボックス

`div`、`p`、`blockquote`、`pre`などの要素型の要素は、デフォルトでは「ブロックボックス」(block box)と呼ばれる種類のボックスを作ります。

ブロックボックスの特徴は、その前後に改行が加わるという点にあります。したがって、連続するブロックボックスは、画面上で縦方向に並ぶことになります。

デフォルトのボックスがブロックボックスではない要素型の要素をブロックボックスにしたいときは、`display`プロパティに対して `block` という値を設定します。たとえば、

```
a { display: block; }
```

というルールを書くことによって、`a`要素をブロックボックスにすることができます。

次のスタイルシートとHTML文書は、`a`要素をブロックボックスで表示します。

スタイルシートの例 `block.css`

---

```
a { display: block; }
```

---

HTML文書の例 `block.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ブロックボックス</title>
    <link rel="stylesheet" href="block.css">
  </head>
  <body>
    <a href="http://www.w3.org/">W3C</a>
    <a href="http://www.ietf.org/">IETF</a>
    <a href="http://www.ecma.org/">ECMA</a>
  </body>
</html>
```

---

### 4.2.3 インラインボックス

`span`、`a`、`em`などの要素型の要素は、デフォルトでは「インラインボックス」(inline box)と呼ばれる種類のボックスを作ります。

ブロックボックスとは対照的に、インラインボックスの前後に改行は加わりません。したがって、連続するインラインボックスは、画面上で横方向に並ぶことになります。

デフォルトのボックスがインラインボックスではない要素型の要素をインラインボックスにしたいときは、`display` プロパティに対して `inline` という値を設定します。たとえば、

```
p { display: inline; }
```

というルールを書くことによって、`p` 要素をインラインボックスにすることができます

次のスタイルシートと HTML 文書は、`p` 要素をインラインボックスで表示します。

スタイルシートの例 `inline.css`

---

```
p { display: inline; }
```

---

HTML 文書の例 `inline.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>インラインボックス</title>
    <link rel="stylesheet" href="inline.css">
  </head>
  <body>
    <p>私は段落です。</p>
    <p>私も段落です。</p>
    <p>そして私も段落です。</p>
  </body>
</html>
```

---

## 4.3 パディング

### 4.3.1 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、さまざまな広さのパディングを持つ段落を表示します。

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`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>パディング</title>
    <link rel="stylesheet" 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>
```

---

### 4.3.2 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。

個別のプロパティを使うことによっても、領域の上下左右を個別に設定することができます。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とHTML文書は、上下左右で異なる広さのパディングを持つ段落を表示します。

スタイルシートの例 `padtrbl.css`

---

```
p { font-size: 40px; color: #090; background-color: #ff9; }
#id000 { padding: 10px 20px 40px 80px; }
#id001 {
  padding-top: 80px;
  padding-right: 40px;
  padding-bottom: 20px;
  padding-left: 10px;
}
```

---

HTML文書の例 `padtrbl.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>パディングの上下左右</title>
    <link rel="stylesheet" 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40ピクセル、左が80ピクセルです。</p>
    <p id="id001">この段落のパディングは、上が80ピクセル、
    右が40ピクセル、下が20ピクセル、左が10ピクセルです。</p>
  </body>
</html>
```

---

## 4.4 ボーダー

### 4.4.1 ボーダーの基礎

`border`というプロパティは、ボーダーの幅と形状と色をあらわします（ボーダーの幅というのは、パディングエッジからボーダーエッジまでの距離のことです）。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 red; }
```

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、色が赤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、ボーダーを持つ段落を表示します。

スタイルシートの例 border.css

---

```
p {
  font-size: 30px;
  color: maroon;
  background-color: silver;
  padding: 20px;
  border: 10px solid blue;
}
```

---

HTML 文書の例 border.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ボーダー</title>
    <link rel="stylesheet" href="border.css">
  </head>
  <body>
    <p>段落のボーダーは、幅が 10 ピクセル、形状が実線、
    色が青色です。</p>
  </body>
</html>
```

---

#### 4.4.2 ボーダーの形状

`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。このプロパティを使うことによって、ボーダーの形状だけを指定する宣言を書くことができます。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。

ボーダーの形状としては、次のようなものがあります。

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

次のスタイルシートと HTML 文書は、さまざまな形状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 borderstyle.css

---

```
p {
  text-align: center;
  font-size: 30px;
  color: #900;
  background-color: #6ff;
  padding: 30px;
  border: 10px solid #690;
}
.none { border-style: none; }
.solid { border-style: solid; }
.dashed { border-style: dashed; }
.dotted { border-style: dotted; }
```

```
.double { border-style: double; }
.groove { border-style: groove; }
.ridge { border-style: ridge; }
.inset { border-style: inset; }
.outset { border-style: outset; }
```

---

#### HTML 文書の例 borderstyle.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ボーダーの形状</title>
    <link rel="stylesheet" href="borderstyle.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>
```

---

#### 4.4.3 ボーダーの幅

`border-width` というプロパティは、ボーダーの幅をあらわします。このプロパティを使うことによって、ボーダーの幅だけを指定する宣言を書くことができます。たとえば、

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

というルールを書くことによって、引用文のボーダーの幅として 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、幅は 0 になります。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、異なる幅のボーダーを持つ三つの段落を表示します。

#### スタイルシートの例 borderwidth.css

---

```
p {
  font-size: 30px;
  color: #090;
  background-color: #ff9;
  padding: 20px;
  border: 0px solid #00c;
}
#id000 { border-width: 10px; }
#id001 { border-width: 20px; }
#id002 { border-width: 40px; }
```

---

#### HTML 文書の例 borderwidth.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ボーダーの幅</title>
    <link rel="stylesheet" href="borderwidth.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 40 ピクセルです。</p>
  </body>
```

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

表 4.1: ボーダーの上下左右を個別に設定するプロパティ

---

</html>

#### 4.4.4 ボーダーの色

`border-color`というプロパティは、ボーダーの色をあらわします。このプロパティを使うことによって、ボーダーの幅だけを指定する宣言を書くことができます。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、異なる色のボーダーを持つ三つの段落を表示します。

スタイルシートの例 `bordercolor.css`

```
p {
  font-size: 30px;
  color: #009;
  background-color: white;
  padding: 20px;
  border: 20px solid black;
}
#id000 { border-color: #966; }
#id001 { border-color: #696; }
#id002 { border-color: #669; }
```

HTML 文書の例 `bordercolor.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>ボーダーの色</title>
    <link rel="stylesheet" href="bordercolor.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>
```

#### 4.4.5 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 4.1 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、ということはありません。

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: #009;
    background-color: #9ff;
    padding: 30px;
    border: 30px solid #090;
}
#id000 {
    border-style: dotted solid dashed double;
}
#id001 {
    border-top-color: red;
    border-right-color: green;
    border-bottom-color: blue;
    border-left-color: yellow;
}

```

---

#### HTML 文書の例 bortrbl.html

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>ボーダーの上下左右</title>
    <link rel="stylesheet" href="bortrbl.css">
  </head>
  <body>
    <p id="id000">上が点線、右が実線、下が破線、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>

```

---

#### 4.4.6 角丸

ボーダーの角は、丸くすることもできます。丸くなった角は、「角丸 (かどまる)」(rounded corner) と呼ばれます。

`border-radius` というプロパティは、角丸の半径をあらわします。このプロパティを使うことによって、ボーダーの角を丸くすることができます。たとえば、

```
border-radius: 30px;
```

という宣言は、半径が 30 ピクセルの円に沿ってボーダーの角を丸くする、ということを意味しています。

`border-radius` に設定する値は、

水平方向の半径 / 垂直方向の半径

というように書くこともできます。このように書くことによって、ボーダーの角を楕円に沿って丸くすることができます。たとえば、

```
border-radius: 60px / 30px;
```

という宣言は、水平方向の半径が 60 ピクセルで垂直方向の半径が 30 ピクセルの楕円に沿ってボーダーの角を丸くする、ということを意味しています。

四箇所の角丸の半径を個別に設定したい場合は、`border-radius` の値として、

左上、右上、右下、左下

という順序でそれぞれの半径を空白で区切って並べたものを書くか、または次のプロパティを使います。

```

border-top-left-radius    左上の角丸の半径。
border-top-right-radius   右上の角丸の半径。
border-bottom-right-radius 右下の角丸の半径。
border-bottom-left-radius 左下の角丸の半径。

```

これらのプロパティに設定する値は、1 個の長さか、または 2 個の長さを空白で区切って並べたものです。長さが 1 個の場合はそれが半径になって、2 個の場合は、1 個目が水平方向の長さ、2 個目が垂直方向の長さになります。

次のスタイルシートと HTML 文書は、さまざまな半径の角丸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 `borderradius.css`

---

```
p {
  font-size: 30px;
  color: #900;
  background-color: #9f9;
  padding: 30px;
  border: 6px solid #090;
}
#id000 { border-radius: 30px; }
#id001 { border-radius: 60px / 30px; }
#id002 { border-radius: 80px 20px 80px 40px / 80px 20px 40px 80px; }
#id003 {
  border-top-left-radius: 80px 80px;
  border-top-right-radius: 20px 20px;
  border-bottom-right-radius: 80px 40px;
  border-bottom-left-radius: 40px 80px;
}
```

---

HTML 文書の例 `borderradius.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>角丸</title>
    <link rel="stylesheet" href="borderradius.css">
  </head>
  <body>
    <p id="id000">半径が 30 ピクセルの角丸。</p>
    <p id="id001">水平方向が半径 60 ピクセル、
    垂直方向が半径 30 ピクセルの角丸。</p>
    <p id="id002">左上、右上、右下、左下のそれぞれで、
    半径が異なる角丸。</p>
    <p id="id003">左上、右上、右下、左下のそれぞれで、
    半径が異なる角丸。</p>
  </body>
</html>
```

---

## 4.5 マージン

### 4.5.1 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、さまざまな広さのマージンを持つ段落を表示します。

スタイルシートの例 `margin.css`

---

```
div {
  background-color: silver;
  padding: 0px;
  border: 2px solid red;
  margin: 10px;
}
p {
  font-size: 40px;
  color: white;
  background-color: navy;
  padding: 4px;
```

```

}
#id000 { margin: 0px; }
#id001 { margin: 20px; }
#id002 { margin: 40px; }

```

---

#### HTML 文書の例 margin.html

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>マージン</title>
    <link rel="stylesheet" href="margin.css">
  </head>
  <body>
    <div><p>この段落のマージンはデフォルトです。</p></div>
    <div><p id="id000">この段落のマージンは 0 ピクセルです。</p></div>
    <div><p id="id001">この段落のマージンは 20 ピクセルです。</p></div>
    <div><p id="id002">この段落のマージンは 40 ピクセルです。</p></div>
  </body>
</html>

```

---

#### 4.5.2 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と HTML 文書は、上下左右で異なる広さのマージンを持つ段落を表示します。

#### スタイルシートの例 martrbl.css

---

```

div {
  background-color: silver;
  padding: 0px;
  border: 2px solid blue;
  margin: 10px;
}
p {
  font-size: 40px;
  color: white;
  background-color: green;
  padding: 4px;
}
#id000 { margin: 10px 20px 40px 80px; }
#id001 {
  margin-top: 80px;
  margin-right: 40px;
  margin-bottom: 20px;
  margin-left: 10px;
}

```

---

#### HTML 文書の例 martrbl.html

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>マージンの上下左右</title>
    <link rel="stylesheet" href="martrbl.css">
  </head>
  <body>
    <div><p id="id000">この段落のマージンは、上が 10 ピクセル、
    右が 20 ピクセル、下が 40 ピクセル、左が 80 ピクセルです。</p></div>

```

```
<div><p id="id001">この段落のマージンは、上が 80 ピクセル、
  右が 40 ピクセル、下が 20 ピクセル、左が 10 ピクセルです。</p></div>
</body>
</html>
```

---

### 4.5.3 マージンの相殺

ブロックボックスが上下に隣接している場合、それらのボックスのマージンは、重ねて表示されます。ですから、上のボックスのボーダーエッジから下のボックスのボーダーエッジまでの距離は、上下のボックスのマージンのうちで、広いほうの距離に一致することになります。CSS が持っているこの規則は、「マージンの相殺」(collapsing margins) と呼ばれます。

次のスタイルシートと HTML 文書は、マージンの相殺を確認するためのものです。

スタイルシートの例 `collapsing.css`

---

```
div {
  background-color: silver;
  padding: 0px;
  border: 2px solid green;
  margin: 10px;
}
p {
  font-size: 40px;
  color: white;
  background-color: maroon;
  padding: 4px;
}
#id000 { margin: 20px; }
#id001 { margin: 40px; }
```

---

HTML 文書の例 `collapsing.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>マージンの相殺</title>
    <link rel="stylesheet" href="collapsing.css">
  </head>
  <body>
    <div>
      <p id="id000">この段落のマージンは 20 ピクセルです。</p>
      <p id="id001">この段落のマージンは 40 ピクセルです。</p>
    </div>
  </body>
</html>
```

---

## 4.6 ボックスの大きさ

### 4.6.1 ボックスの横の長さ

ブロックボックスの横の長さは、デフォルトでは、自分を囲んでいるボックスの中に納まる最大の長さになっていますが、特定の長さにも限定することも可能です。

`width` というプロパティは、ボックスの横の長さ（厳密に言えば、コンテンツエッジの横の長さ）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、コンテンツ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、さまざまな横の長さを持つ段落を表示します。

スタイルシートの例 `width.css`

---

```
p {
  font-size: 40px;
  padding: 40px;
  border: 40px solid navy;
}
```

```

}
#id000 { width: 200px; }
#id001 { width: 400px; }
#id002 { width: 800px; }

```

---

#### HTML 文書の例 width.html

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>ボックスの横の長さ</title>
    <link rel="stylesheet" href="width.css">
  </head>
  <body>
    <p id="id000">この段落の横の長さは 200 ピクセルです。 </p>
    <p id="id001">この段落の横の長さは 400 ピクセルです。 </p>
    <p id="id002">この段落の横の長さは 800 ピクセルです。 </p>
  </body>
</html>

```

---

#### 4.6.2 ボックスの縦の長さ

ブロックボックスの縦の長さは、デフォルトでは、自分の内容を表示することのできる必要最小限の長さになっていますが、特定の長さに限定することも可能です。

`height` というプロパティは、ボックスの縦の長さ（厳密に言えば、コンテンツエッジの縦の長さ）をあらわします。たとえば、

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

というルールを書くことによって、コンテンツエッジの縦の長さを常に 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、さまざまな縦の長さを持つ段落を表示します。

#### スタイルシートの例 height.css

---

```

p {
  font-size: 40px;
  padding: 40px;
  border: 40px solid maroon;
}
#id000 { height: 80px; }
#id001 { height: 120px; }
#id002 { height: 160px; }

```

---

#### HTML 文書の例 height.html

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>ボックスの縦の長さ</title>
    <link rel="stylesheet" href="height.css">
  </head>
  <body>
    <p id="id000">この段落の縦の長さは 80 ピクセルです。 </p>
    <p id="id001">この段落の縦の長さは 120 ピクセルです。 </p>
    <p id="id002">この段落の縦の長さは 160 ピクセルです。 </p>
  </body>
</html>

```

---

#### 4.6.3 オーバーフロー

`height` プロパティによってボックスの縦の長さが限定されている場合、そのボックスのコンテンツエリアは、必ずしもボックスの内容を表示することのできる十分な広さを持っているとは限りません。ボックスの内容がコンテンツエリアからはみ出す可能性、すなわち、「オーバーフロー」(overflow) が発生する可能性があります。

`overflow` というプロパティは、コンテンツエリアからはみ出したボックスの内容をどのよう

に表示するかということをお知らせします。このプロパティに対しては、次の値を設定することができます。

`visible` はみ出した形のままで表示する。  
`hidden` はみ出した部分は表示しない。  
`scroll` 縦と横のスクロールバーを常に表示して、スクロールで表示する。  
`auto` 必要に応じてスクロールバーを表示して、スクロールで表示する。

次のスタイルシートと HTML 文書は、`overflow` プロパティに設定するそれぞれの値がどのような意味かということを示しています。

スタイルシートの例 `overflow.css`

---

```
p {
  font-size: 40px;
  border: 2px solid red;
  margin-bottom: 100px;
  width: 200px;
  height: 100px;
}
.visible { overflow: visible; }
.hidden { overflow: hidden; }
.scroll { overflow: scroll; }
.auto { overflow: auto; }
```

---

HTML 文書の例 `overflow.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>オーバーフロー</title>
    <link rel="stylesheet" href="overflow.css">
  </head>
  <body>
    <p class="visible">visible は、
    はみ出した形のままで表示します。</p>
    <p class="hidden">hidden は、はみ出した部分は表示しません。</p>
    <p class="scroll">scroll は、縦と横のスクロールバーを常に
    表示して、スクロールで表示します。</p>
    <p class="auto">auto は、必要に応じてスクロールバーを表示して、
    スクロールで表示します。</p>
    <p class="auto">私も auto です。</p>
  </body>
</html>
```

---

#### 4.6.4 ボックスの水平方向のアラインメント

ボックスの内部にボックスがあるとき、内側のボックスをどのような位置に配置するかということは、ボックスの「アラインメント」(alignment) と呼ばれます。

`width` プロパティで横の長さが限定されているボックスの水平方向のアラインメントは、デフォルトでは左寄せになります。

横の長さが限定されているボックスを右寄せしたいときは、その左側のマージンに対して `auto` という値を設定して、右側のマージンをゼロにします。

横の長さが限定されているボックスをセンタリングしたいときは、その左側と右側の両方のマージンに対して `auto` を設定します。

次のスタイルシートと HTML 文書は、一つ目の段落を左寄せ、二つ目の段落を右寄せ、三つ目の段落をセンタリングします。

スタイルシートの例 `boxalign.css`

---

```
p {
  font-size: 40px;
  border: 2px solid green;
  width: 400px;
}
.right { margin-left: auto; margin-right: 0px; }
```

```
.center { margin-left: auto; margin-right: auto; }
```

---

#### HTML 文書の例 boxalign.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ボックスの水平方向のアラインメント</title>
    <link rel="stylesheet" href="boxalign.css">
  </head>
  <body>
    <p>左寄せ (デフォルト) </p>
    <p class="right">右寄せ</p>
    <p class="center">センタリング</p>
  </body>
</html>
```

---

## 4.7 フローティング

### 4.7.1 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。

通常の要素の場合は、それがブロックボックスならば上から下に向かって並べられ、インラインボックスならば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、それがあた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。

### 4.7.2 回り込み

フローティング要素に後続する要素のテキストは、最初の部分がフローティング要素の横のスペースに表示されて、そのスペースからあふれた部分はフローティング要素の下に表示されます。そのように、フローティング要素の周囲にテキストが表示されることは、テキストの「回り込み」(wrap) と呼ばれます。

テキストの回り込みは、フローティング要素を作る目的のひとつです。

次のスタイルシートと HTML 文書は、フローティング要素に後続する要素のテキストがフローティング要素の周囲に回り込むことを示しています。

#### スタイルシートの例 wrap.css

---

```
p {
  font-size: 40px;
  border: 2px solid blue;
  padding: 10px;
  margin: 10px;
}
.floatleft { float: left; height: 400px; }
```

---

#### HTML 文書の例 wrap.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>回り込み</title>
    <link rel="stylesheet" href="wrap.css">
  </head>
```

```

<body>
  <p class="floatleft">フローティング要素</p>
  <p>私は第一の段落です。この段落のテキストは、
  フローティング要素の周囲に回り込んで表示されます。</p>
  <p>私は第二の段落です。この段落のテキストも、
  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>

```

---

### 4.7.3 回り込みの解除

ところで、フローティング要素に後続する要素のテキストを、フローティング要素の周囲に回り込ませたくない、というときはどうすればいいのでしょうか。

`clear` というプロパティは、回り込みの解除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、次のような値を設定することができます。

`left` 左寄りのフローティング要素に対する回り込みを解除する。  
`right` 右寄りのフローティング要素に対する回り込みを解除する。  
`both` フローティング要素に対する回り込みを解除する。  
`none` 回り込みを解除しない。

次のスタイルシートと HTML 文書は、`clear` プロパティを使うことによって回り込みを解除することができるということを示しています。

スタイルシートの例 `clear.css`

---

```

p {
  font-size: 40px;
  border: 2px solid blue;
  padding: 10px;
  margin: 10px;
}
.floatleft { float: left; height: 400px; }
.clearleft { clear: left; }

```

---

HTML 文書の例 `clear.html`

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>回り込みの解除</title>
    <link rel="stylesheet" href="clear.css">
  </head>
  <body>
    <p class="floatleft">フローティング要素</p>
    <p>私は第一の段落です。この段落のテキストは、
    フローティング要素の周囲に回り込んで表示されます。</p>
    <p class="clearleft">私は第二の段落です。この段落は、
    左寄りのフローティング要素に対する回り込みが
    解除されていますので、左寄りのフローティング要素の下に
    表示されます。</p>
  </body>
</html>

```

---

### 4.7.4 段組

フローティング要素を作るもうひとつの目的は、段組です。

「段組」(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。段組に使われるそれぞれの領域は、「段」(column) と呼ばれます。 $n$  個の段から構成される段組は、「 $n$  段組」(n-column) と呼ばれます。

左寄りの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段組で表示します。

スタイルシートの例 column.css

---

```
div { font-size: 40px; color: white; margin: 0px; }
#container { width: 300px; margin: 0px auto; }
#column00 {
  width: 100px;
  float: left;
  background-color: green;
}
#column01 {
  width: 200px;
  float: left;
  background-color: blue;
}
```

---

HTML 文書の例 column.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>段組</title>
    <link rel="stylesheet" href="column.css">
  </head>
  <body>
    <div id="container">
      <div id="column00">私は第一の段です。</div>
      <div id="column01">私は第二の段です。</div>
    </div>
  </body>
</html>
```

---

## 4.8 テーブル

### 4.8.1 ボックスとしてのテーブル

table 要素、つまりテーブルは、ひとつのボックスとして表示されます。そしてさらに、th 要素と td 要素、つまりテーブルを構成するそれぞれのセルもまた、ひとつの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。

次のスタイルシートと HTML 文書は、table 要素、th 要素、td 要素に対してもボックスに関するスタイルを適用することができる、ということを示しています。

スタイルシートの例 table.css

---

```
table {
  font-size: 30px;
  background-color: #ff6;
  border: 5px solid #f66;
  padding: 20px;
}
th, td {
  color: #090;
  background-color: #cfc;
  border: 8px double #0c0;
  padding: 10px;
}
```

---

HTML 文書の例 table.html

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>ボックスとしてのテーブル</title>
    <link rel="stylesheet" href="table.css">
  </head>
  <body>
```

```

<table>
  <tr>
    <th>奈良の大仏</th><td>14.98 メートル</td>
  </tr>
  <tr>
    <th>鎌倉の大仏</th><td>11.39 メートル</td>
  </tr>
</table>
</body>
</html>

```

---

#### 4.8.2 セルの間隔

セルとセルとのあいだの間隔を指定したいときは、`margin` プロパティではなくて、専用のプロパティを使う必要があります。それは、`border-spacing` というプロパティです。このプロパティに長さを設定したスタイルを `table` 要素に適用すると、そのテーブルを構成しているそれぞれのセルのあいだの間隔は、その長さになります。

横方向の間隔と縦方向との間隔とで異なる長さを設定したいときは、2個の長さを空白で区切って書きます。そうすると、1個目が横方向の間隔、2個目が縦方向の間隔になります。

次のスタイルシートと HTML 文書は、`border-spacing` プロパティによってセルの間隔が指定されたテーブルを表示します。

スタイルシートの例 `spacing.css`

---

```

table { font-size: 30px; border: 2px solid blue; }
th, td { border: 2px solid orange; padding: 10px; }
#id000 { border-spacing: 20px; }
#id001 { border-spacing: 30px 10px; }

```

---

HTML 文書の例 `spacing.html`

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>セルの間隔</title>
    <link rel="stylesheet" href="spacing.css">
  </head>
  <body>
    <table id="id000">
      <tr>
        <th>富士山</th><td>3776 メートル</td>
      </tr>
      <tr>
        <th>天保山</th><td>4.53 メートル</td>
      </tr>
    </table>
    <table id="id001">
      <tr>
        <th>宗谷岬</th><td>北緯 45 度 31 分 22 秒</td>
      </tr>
      <tr>
        <th>佐多岬</th><td>北緯 30 度 59 分 10 秒</td>
      </tr>
    </table>
  </body>
</html>

```

---

#### 4.8.3 結合ボーダーモデル

テーブルとセルのボーダーには、「分離ボーダーモデル」(separate border model) と「結合ボーダーモデル」(collapsing border model) という二つのボーダーモデルがあって、デフォルトでは分離ボーダーモデルになっています。

分離ボーダーモデルでは、どのボーダーも重ならずに表示されます。それに対して結合ボーダーモデルでは、隣接しているテーブルとセルのボーダーは、重ねて表示されます (`border-spacing` の値は無視されます)。

結合ボーダーモデルでテーブルを表示したいときは、`border-collapse`というプロパティに対して `collapse` という値を設定して、そのスタイルを `table` 要素に適用します。

次のスタイルシートと HTML 文書は、結合ボーダーモデルでテーブルを表示します。

スタイルシートの例 `collapse.css`

```
table { font-size: 30px; border-collapse: collapse; }
th, td { border: 5px solid green; padding: 10px; }
```

HTML 文書の例 `collapse.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>結合ボーダーモデル</title>
    <link rel="stylesheet" href="collapse.css">
  </head>
  <body>
    <table>
      <tr>
        <th>フォボス</th><td>7 時間 39 分</td>
      </tr>
      <tr>
        <th>ダイモス</th><td>30 時間 18 分</td>
      </tr>
    </table>
  </body>
</html>
```

#### 4.8.4 表示されない空セル

内容のないセルは、「空セル」(empty cell)と呼ばれます。デフォルトでは空セルもセルとして表示されますが、空セルを表示しないことも可能です。

表示されない空セルを作りたいときは、`empty-cells`というプロパティに `hide` という値を設定して、そのスタイルをセルの要素に適用します。

次のスタイルシートと HTML 文書は、`empty-cells` プロパティを使うことによって、表示されない空セルを作っています。

スタイルシートの例 `empty.css`

```
th, td {
  font-size: 30px;
  color: #00c;
  background-color: #ccf;
  border: 5px solid #00c;
  padding: 10px;
}
.hide { empty-cells: hide; }
```

HTML 文書の例 `empty.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>表示されない空セル</title>
    <link rel="stylesheet" href="empty.css">
  </head>
  <body>
    <table>
      <tr>
        <th>空ではないセル</th><td>私は空ではありません。</td>
      </tr>
      <tr>
        <th>デフォルトの空セル</th><td></td>
      </tr>
      <tr>
        <th>表示されない空セル</th><td class="hide"></td>
      </tr>
    </table>
  </body>
</html>
```

```

    </table>
  </body>
</html>

```

---

#### 4.8.5 垂直方向のアラインメント

テーブルのひとり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大きな問題になります。

`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。このプロパティによって記述されたスタイルをセルの要素に適用することによって、そのセルの垂直方向のアラインメントを指定することができます。

`vertical-align`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`は中央配置という意味です。

次のスタイルシートとHTML文書は、`td`要素に対して、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 `vertical.css`

---

```

td { font-size: 30px; border: 5px solid teal; padding: 10px; }
.top { vertical-align: top; }
.bottom { vertical-align: bottom; }
.middle { vertical-align: middle; }

```

---

HTML文書の例 `vertical.html`

---

```

<!DOCTYPE html>
<html>
  <head>
    <title>垂直方向のアラインメント</title>
    <link rel="stylesheet" href="vertical.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>

```

---

## 4.9 リスト

### 4.9.1 リストの基礎

`ul`要素または`ol`要素は、「リスト」(list)または「箇条書き」と呼ばれるものを作ります。

リストを構成するそれぞれの項目は、`li`という要素型の要素によって作られます。`li`要素は、デフォルトでは「リストアイテムボックス」(list item box)と呼ばれる種類のボックスを作ります。

リストアイテムボックスは、ブロックボックスと同様に、その前後に改行が加わります。それに加えて、リストアイテムボックスは、「マーカー」(marker)と呼ばれる記号が左側に表示されます。

### 4.9.2 マーカーのタイプ

`list-style-type`というプロパティは、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	線のみの円。
square	正方形。
decimal	アラビア数字。1、2、3、4、5、・・・
upper-alpha	英字の大文字。A、B、C、D、E、・・・
lower-alpha	英字の小文字。a、b、c、d、e、・・・
upper-roman	ローマ数字の大文字。I、II、III、IV、V、・・・
lower-roman	ローマ数字の小文字。i、ii、iii、iv、v、・・・
none	マーカーを表示しない。

次のスタイルシートとHTML文書は、`list-style-type` プロパティを使うことによって、マーカーのタイプを指定することができることを示しています。

スタイルシートの例 `marker.css`

---

```
li { list-style-type: upper-roman; }
li li { list-style-type: lower-alpha; }
```

---

HTML 文書の例 `marker.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>マーカーのタイプ</title>
    <link rel="stylesheet" href="marker.css">
  </head>
  <body>
    <ol>
      <li>序論</li>
      <li>本論
        <ol>
          <li>磯野家の秘密</li>
          <li>イクラ語の文法と語彙</li>
          <li>アナゴ vs ムスカ大佐</li>
        </ol>
      </li>
      <li>結論</li>
    </ol>
  </body>
</html>
```

---

#### 4.9.3 マーカーの位置

マーカーは、デフォルトでは項目のマージンに表示されますが、項目のコンテンツエリア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	マーカーをマージンに表示する。デフォルト。
inside	マーカーをコンテンツエリアに表示する。

次のスタイルシートとHTML文書は、`list-style-position` プロパティを使うことによって、項目のコンテンツエリアにマーカーを表示することができることを示しています。

スタイルシートの例 `position.css`

---

```
li { border: 2px solid blue; margin: 10px; padding: 10px; }
li.inside { list-style-position: inside; }
```

---

HTML 文書の例 `position.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>マーカーの位置</title>
    <link rel="stylesheet" href="position.css">
  </head>
```

```
<body>
  <ul>
    <li>マーカーがマージンにある項目 (デフォルト) </li>
    <li class="inside">マーカーがコンテンツエリアにある項目</li>
  </ul>
</body>
</html>
```

---

#### 4.9.4 リストアイテムボックス

第 4.2 節で説明したように、ボックスの種類というのは、`display` というプロパティを使うことによって自由に変更することができます。ですから、デフォルトがリストアイテムボックスではない要素型の要素も、`display` プロパティに対して `list-item` という値を設定することによって、リストアイテムボックスにすることができます。たとえば、

```
a { display: list-item; }
```

というルールを書くことによって、`a` 要素をリストアイテムボックスにすることができます。

次のスタイルシートと HTML 文書は、`a` 要素をリストアイテムボックスで表示します。

スタイルシートの例 `listitem.css`

---

```
a { display: list-item; margin-left: 30px; }
```

---

HTML 文書の例 `listitem.html`

---

```
<!DOCTYPE html>
<html>
  <head>
    <title>リストアイテムボックス</title>
    <link rel="stylesheet" href="listitem.css">
  </head>
  <body>
    <a href="http://www.nasa.gov/">NASA</a>
    <a href="http://www.jaxa.jp/">JAXA</a>
    <a href="http://www.nao.ac.jp/">国立天文台</a>
  </body>
</html>
```

---

## 参考文献

- [Backgrounds,2011] Bert Bos, Erika J. Etemad and Brad Kemper (eds.), “CSS Backgrounds and Borders Module Level 3”, World Wide Web Consortium, 2011.
- [Color,2011] Tantek Çelik, Chris Lilley and L. David Baron (eds.), “CSS Color Module Level 3”, World Wide Web Consortium, 2011.
- [CSS,2011] Bert Bos, Tantek Çelik, Ian Hickson and Håkon Wium Lie (eds.), “Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification”, World Wide Web Consortium, 2011.
- [Selectors,2011] Erika J. Etemad, Tantek Çelik, Daniel Glazman, Ian Hickson, Peter Linss and John Williams (eds.), “Selectors Level 4”, World Wide Web Consortium, 2011.
- [Text,2011] Erika J. Etemad, Koji Ishii and Shinyu Murakami (eds.), “CSS Text Level 3”, World Wide Web Consortium, 2011.
- [Meyer,2004] Eric A. Meyer, *Cascading Style Sheets: The Definitive Guide, Second Edition*, O'Reilly Media, 2004, ISBN 978-0-596-00525-2. 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 978-4-87311-232-9.
- [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 978-4-8399-1268-0.
- [秋葉,2011] 秋葉秀樹、秋葉ちひろ、小山田晃浩、外村和仁、蒲生トシヒロ、宮澤了祐、『CSS3 デザインプロフェッショナルガイド』、毎日コミュニケーションズ、2011、ISBN 978-4-8399-3546-7.
- [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 978-4-8399-1252-9。
- [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 978-4-8399-1501-8。
- [エ・ビスコム,2011] エ・ビスコム・テック・ラボ、『CSS3 スタンダード・デザインガイド：サンプルでマスターする、CSS3 のデザインテクニック』、毎日コミュニケーションズ、2011、ISBN 978-4-8399-3829-1。
- [大藤,2011] 大藤幹、『10 日でおぼえる CSS/CSS3 入門教室』、翔泳社、2011、ISBN 978-4-7981-2403-2。
- [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 978-4-8443-5838-1。

## 索引

- .css (拡張子), 6
- @import 文, 8
- 10 進数
  - による色の記述, 9
- 16 進数
  - による色の記述, 9
- a 要素, 18, 30
- active 擬似クラス, 17
- auto (オーバーフロー), 41
- auto (マージン), 41
- background-color プロパティ, 5, 30
- bisible (オーバーフロー), 41
- block (ボックスの種類), 30
- blockquote 要素, 30
- bold (フォントの太さ), 24
- border プロパティ, 32
- border-bottom-color プロパティ, 35
- border-bottom-left-radius プロパティ, 36
- border-bottom-right-radius プロパティ, 36
- border-bottom-style プロパティ, 35
- border-bottom-width プロパティ, 35
- border-collapse プロパティ, 46
- border-color プロパティ, 35
- border-left-color プロパティ, 35
- border-left-style プロパティ, 35
- border-left-width プロパティ, 35
- border-radius プロパティ, 36
- border-right-color プロパティ, 35
- border-right-style プロパティ, 35
- border-right-width プロパティ, 35
- border-spacing プロパティ, 45
- border-style プロパティ, 33
- border-top-color プロパティ, 35
- border-top-left-radius プロパティ, 36
- border-top-right-radius プロパティ, 36
- border-top-style プロパティ, 35
- border-top-width プロパティ, 35
- border-width プロパティ, 34
- both (回り込みの解除), 43
- bottom (アラインメント), 47
- center (アラインメント), 26
- circle (マーカーのタイプ), 48
- class 属性, 15
- clear プロパティ, 43
- cm (単位), 11
- collapse (ボーダーモデル), 46
- color プロパティ, 5, 35
- CSS, 4
- cursive (フォント), 24
- dashed (ボーダーの形状), 33
- decimal (マーカーのタイプ), 48
- disc (マーカーのタイプ), 47
- display プロパティ, 30, 31, 49
- div 要素, 30
- dotted (ボーダーの形状), 33
- double (ボーダーの形状), 33
- em (単位), 12
  - によるフォントの大きさの記述, 12
- em 要素, 30
- empty-cells プロパティ, 46
- fantasy (フォント), 24
- first-letter 擬似要素, 18
- first-line 擬似要素, 19
- float プロパティ, 42
- focus 擬似クラス, 17
- font-family プロパティ, 23
- font-size プロパティ, 12
- font-style プロパティ, 25
- font-weight プロパティ, 24
- groove (ボーダーの形状), 33
- head 要素, 6, 7
- height プロパティ, 40
- hidden (オーバーフロー), 41
- hidden (ボーダーの形状), 33
- hide (セルの表示), 46
- hover 擬似クラス, 17
- href 属性, 6
- HTML, 4
- HTML 文書, 4
  - への外部スタイルシートの適用, 6
- ID, 16
- id 属性, 16
- ID セレクタ, 14, 16
- in (単位), 11
- inline (ボックスの種類), 30, 31

- inset (ボーダーの形状), 33
- inside (マーカーの位置), 48
- italic (フォントのスタイル), 25
- left (アラインメント), 26
- left (回り込みの解除), 43
- letter-spacing プロパティ, 12
- li 要素, 47
- line-height プロパティ, 27
- line-through (テキストの装飾), 28
- link 擬似クラス, 18
- link 要素, 6
- list-item (ボックスの種類), 30, 49
- list-style-position プロパティ, 48
- list-style-type プロパティ, 47
- lower-alpha (マーカーのタイプ), 48
- lower-roman (マーカーのタイプ), 48
- LVHA, 18
- margin プロパティ, 37, 45
- margin-bottom プロパティ, 38
- margin-left プロパティ, 38
- margin-right プロパティ, 38
- margin-top プロパティ, 38
- middle (アラインメント), 47
- mm (単位), 11
- monospace (フォント), 24
- left (フローティング), 42
- none (フローティング), 42
- right (フローティング), 42
- none (テキストの装飾), 28
- none (ボーダーの形状), 33
- none (マーカーのタイプ), 48
- none (回り込みの解除), 43
- normal (フォントのスタイル), 25
- normal (フォントの太さ), 24
- nowrap (折り返しの制御), 27
- n* 段組, 43
- oblique (フォントのスタイル), 25
- ol 要素, 47
- outset (ボーダーの形状), 33
- outside (マーカーの位置), 48
- overflow プロパティ, 40
- overline (テキストの装飾), 28
- p 要素, 30
- padding プロパティ, 31
- padding-bottom プロパティ, 32
- padding-left プロパティ, 32
- padding-right プロパティ, 32
- padding-top プロパティ, 32
- pc (単位), 11
- pre 要素, 30
- pt (単位), 11
- px (単位), 12
- rel 属性, 6
- ridge (ボーダーの形状), 33
- right (アラインメント), 26
- right (回り込みの解除), 43
- sans-serif (フォント), 24
- scroll (オーバーフロー), 41
- serif (フォント), 24
- solid (ボーダーの形状), 33
- span 要素, 30
- square (マーカーのタイプ), 48
- style 属性, 8
- style 要素, 7
- stylesheet, 7
- text-align プロパティ, 26
- text-decoration プロパティ, 28
- text-indent プロパティ, 26
- text-shadow プロパティ, 29
- top (アラインメント), 47
- transparent (背景色), 30
- ul 要素, 47
- underline (テキストの装飾), 28
- upper-alpha (マーカーのタイプ), 48
- upper-roman (マーカーのタイプ), 48
- vertical-align プロパティ, 47
- visited 擬似クラス, 18
- W3C, 4
- white-space プロパティ, 27
- width プロパティ, 39, 41
- アスタリスク, 15
- 値, 5
- アラインメント
  - 垂直方向の——, 47
  - テキストの——, 26
  - ボックスの——, 41
- アンカー, 18
- イタリック体, 25
- 位置

- マーカーの——, 48
- 一般兄弟結合子, 19, 21
- 色, 9
  - の 10 進数による記述, 9
  - の 16 進数による記述, 9
  - ボーダーの——, 35
- 色名, 10
- インチ, 11
- インデント, 26
- インポート, 8
- インラインスタイルシート, 6, 8
- インラインボックス, 30
  
- ウェブセーフカラー, 11
- ウェブページ, 4
- 上寄せ, 47
  
- 大きさ
  - ピクセルの——, 12
  - フォントの——, 12
- オーバーフロー, 40
- オブリーク体, 25
  
- 解除
  - 回り込みの——, 43
- 外部スタイルシート, 6
  - の HTML 文書への適用, 6
- 改ページ, 5
- 角括弧, 16
- 影
  - テキストの——, 29
- 箇条書き, 47
- 下線, 28
- 型セレクタ, 15
- 角丸, 36
- 間隔
  - セルの——, 45
  - 文字の——, 12
  
- 擬似クラス, 17
- 擬似クラスセレクタ, 14, 17
- 擬似クラス名, 17
- 擬似要素, 18
- 擬似要素セレクタ, 14, 19
- 擬似要素名, 18
- 行
  - の高さ, 27
  - 最初の——, 19
  
- 空セル, 46
- 空白, 5
- クラスセレクタ, 14, 15
- クラス名, 15
  
- 形状
  - ボーダーの——, 33
  
- 継承, 13
  - 相対単位による長さの——, 13
- 結合子, 19
- 結合ボーダーモデル, 45
  
- 構造
  - 文書の——, 4
  - 子供結合子, 19, 20
  - コメントアウト, 6
  - コメントイン, 6
  - コロソ, 17, 19
  - コンテンツエッジ, 29, 39, 40
  - コンテンツエリア, 29, 40
  - コンマ, 22
  
- 最初
  - の行, 19
  - の文字, 19
  
- 子孫結合子, 19, 20
- 下寄せ, 47
- シャープ, 16
- 種類
  - セレクタの——, 14
  - ボックスの——, 30
- 上下左右
  - パディングの——, 32
  - ボーダーの——, 35
  - マージンの——, 38
  - 領域の——, 32
- 詳細度, 22
- 上線, 28
  
- 垂直方向
  - のアラインメント, 47
- スタイル
  - の適用, 6
  - フォントの——, 25
  - 文書の——, 4
- スタイルシート, 4
  - の部品化, 8
- スタイルシート言語, 4
- スタイルシートファイル, 6
  
- 絶対単位, 11
- セリフ, 24
- セル
  - の間隔, 45
  - ボックスとしての——, 44
- セレクタ, 4, 5, 14
  - の種類, 14
- セレクタリスト, 14, 22
- 前景色, 5
- 宣言, 5
- 宣言ブロック, 4, 5, 14
- 全称セレクタ, 14, 15

- センタリング
  - テキストの——, 26
  - ボックスの——, 41
- センチメートル, 11
- 相殺
  - マージンの——, 39
- 装飾
  - テキストの——, 28
- 相対単位, 11, 12
  - による長さの継承, 13
- 属性セレクト, 14, 16
- 大なり, 20
- タイプ
  - マーカーの——, 47
- タイプセレクト, 14
- 高さ
  - 行の——, 27
- タグ, 4
- 縦
  - ボックスの——の長さ, 40
- タブ, 5
- 段, 43
- 単位
  - 長さの——, 11
- 段組, 43
- 中央配置, 47
- 注釈, 6
- 長方形
  - ボックスを構成する——, 29
- チルダ, 21
- テーブル
  - ボックスとしての——, 44
- テキスト
  - のアラインメント, 26
  - の影, 29
  - のセンタリング, 26
  - の装飾, 28
  - の左寄せ, 26
  - の右寄せ, 26
- 適用
  - HTML 文書への外部スタイルシートの——, 6
  - スタイルの——, 6
- ドット, 15
- 取り消し線, 28
- 内部スタイルシート, 6, 7
- 長さ
  - の単位, 11
  - 相対単位による——の継承, 13
  - ボックスの縦の——, 40
  - ボックスの横の——, 39
- パイカ, 11
- 背景色, 5, 30
- パディング, 29, 31
  - の上下左右, 32
- パディングエッジ, 29, 32
- 幅
  - ボーダーの——, 34
- 汎用フォントファミリー名, 24
- 光の三原色, 9, 10
- ピクセル
  - の大きさ, 12
- 左寄せ
  - テキストの——, 26
  - ボックスの——, 41
- 筆記体, 24
- フォント, 23
  - の大きさ, 12
  - のスタイル, 25
  - の太さ, 24
- フォントファミリー, 23
- フォントファミリー名, 23
- 複合セレクト, 14, 19
- 復帰, 5
- 太さ
  - フォントの——, 24
- 部品化
  - スタイルシートの——, 8
- プラス, 21
- フローティング要素, 42
- ブロックボックス, 30, 39
- プロパティ, 5
- プロポーショナル, 24
- 文書, 4
  - の構造, 4
  - のスタイル, 4
- 分離ボーダーモデル, 45
- ポイント, 11
- ボーダー, 29, 32
  - の色, 35
  - の上下左右, 35
  - の形状, 33
  - の幅, 34
- ボーダーエッジ, 29, 32
- ぼかし幅, 29
- ボックス, 29
  - としてのセル, 44
  - としてのテーブル, 44
  - のアラインメント, 41
  - の種類, 30
  - のセンタリング, 41
  - の縦の長さ, 40

- の左寄せ, 41
- の右寄せ, 41
- の横の長さ, 39
- を構成する長方形, 29
- を構成する領域, 29
- ホワイトスペース, 5, 20
  
- マーカー, 47
  - の位置, 48
  - のタイプ, 47
- マークアップ言語, 4
- マージン, 29, 37
  - の上下左右, 38
  - の相殺, 39
- マージンエッジ, 29
- 回り込み, 42
  - の解除, 43
  
- 右寄せ
  - テキストの—, 26
  - ボックスの—, 41
- ミリメートル, 11
  
- 文字
  - の間隔, 12
  - 最初の—, 19
  
- ユーザーアクション擬似クラス, 17
  
- 横
  - ボックスの—の長さ, 39
  
- リスト, 47
- リストアイテムボックス, 30, 47
- 領域
  - の上下左右, 32
  - ボックスを構成する—, 29
- リンク擬似クラス, 18
- 隣接兄弟結合子, 19, 21
  
- ルール, 4, 14
  
- ローマン体, 25