Before After
CSS テクニック
特別編

目指せ脱初心者！スマートなCSSの書き方教えます

スパゲティCSS解消テクニック

自分でも把握できない
ごちゃごちゃ「スパゲティ状態」の
スタイルシートにさようなら

text：長谷川恭久　*photo*：Tsushima Takao

CSSが広く使われるようになり、ウェブサイトの構築において情報とデザインの分離が明確になりました。つまり、XHTMLやHTMLに記述するのは情報そのものや文書構造だけとなり、視覚的なデザインはCSSに記述されるようになりました。

これにより、サイト構築の効率化が進み、ある程度のCSSの知識があればサイトの見た目を容易に変えられるようになりました。CSSでデザインされたサイトでは、ウェブ制作に詳しい人でなくても、CSSを数行修正するだけで、全ページの見た目を変えたりすることも簡単にできます。

CSSで高度なテクニックを駆使するのならば、古いブラウザーでも新しいブラウザーでも同じように表示されるようにする「ハック」と呼ばれるものや、特定のブラウザーに特有のCSS解釈に関するバグを回避する書き方などもありますが、そういった技法を使うのでなければ、CSSは比較的簡単に覚えることができます。ブログ運営に慣れてきた人も少しずつCSSに手を加えていることでしょう。

しかし、CSSファイルにスタイルを定義していくと、あっという間にCSSファイルの内容は何十行どころではない長さになってしまいます。凝ったデザインにすると、定義しなければならないプロパティーも増え、ファイルサイズが大きくなってしまいます。そして何よりも、CSSファイルが長くなりすぎて、どこに何を書いたのかが自分でもわからなくなってしまいます。いわゆる「スパゲティ状態」です。ブログサービスが提供している標準のテンプレートの中にも、長々とスタイルの定義リストが記述されていて、少し修正を加えたくても修正するべき場所を探すのに時間がかかることもしばしばあります。

そこで、CSSがこんがらがったスパゲティ状態にならないようにして、できる限り短く、そして整理された状態でCSSを記述するためのヒントを紹介します。

CSSでのスタイル指定の基本

CSSファイルのスマートな書き方を学ぶ前に、CSSでのスタイル指定の基礎をおさらいしておきましょう。

CSSでは、スタイルを指定するHTML／XHTMLのタグ名やクラス名、ID名などの要素を「セレクター」として指定して、中かっこで囲んで中にプロパティー名と指定する値をコロン（:）でつないで書きます。中かっこの中に複数のプロパティーと値のペアを指定する場合は、各ペアの最後にセミコロン（;）で区切りを指定します。

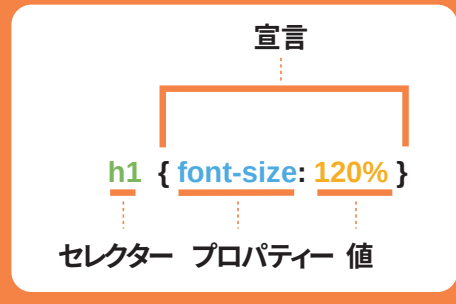

図1　基本的なCSSの文法

スパゲティCSS解消法①

親子の関係は大事です

スッキリ度

HTML／XHTMLタグにクラス名を指定して、付けた名前に従ってCSSにプロパティーを指定する方法はよく使われます。サイトの構造が図2のようになっている場合、navとsectionのそれぞれにあるh2見出しにスタイルを指定するのに、コード1-1のようにすることがあります。もちろん、こう書く必要がある場合もありますが、クラス名やID名が多くなると、あとからCSSを編集しづらくなるうえに、XHTMLファイルの情報を不必要に増やしてしまいます。

図2では、レイアウトを作る3つのセクションにID名が付けられています。このIDをうまく利用すれば、同じh2タグに対して異なるスタイルを指定できます。

コード1-1のHTMLコードには、#navと#sectionの中にそれぞれh2があります。このように、1つの要素の中に別の要素が存在するような階層的な上下関係（親子関係とも言います）がある場合、その関係をセレクターとして使うことが可能です。これを「子孫セレクター」と呼びます。コード1-2のように、親要素と子要素の間に半角スペースを空けて書くと子孫セレクターの指定になります。このように指定した場合、ほかにh2タグが使われていても、指定した親子関係にないh2にはそのスタイルが適用されません。

また、コード1-3のように、どこにあるh2にも共通する定義をまず指定しておいて、独自のスタイルを指定したい親子関係のh2に関する定義だけを子孫セレクターを使って指定すれば、よりCSSファイルを短くまとめることができます。

HTMLの階層構造を利用したセレクターはほかにもありますが、子孫セレクターはNetscape 4.xのような古いブラウザーでも対応しているので安心して使えます。子孫セレクターを使ってもCSSファイルを劇的に小さくすることはできませんが、HTMLをよりシンプルにできます。

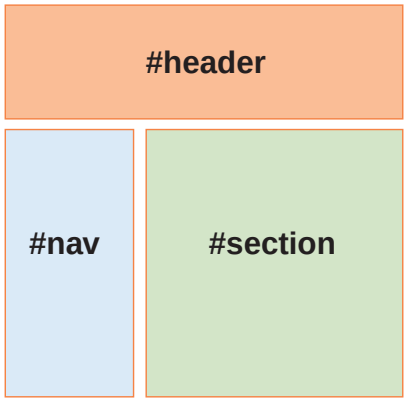

図2　ブログの典型的なブロック分け

スパゲティ　コード1-1
無駄にクラスが設定された例

HTML

```
<div id="nav">
  <h2 class="nav-head2">カテゴリー名</h2>
</div>
<div id="section">
  <h2 class="section-head2">カテゴリー名</h2>
</div>
```

CSS

```
.nav-head2{
  color: #300;
  font-size:100%;
}
.section-head2{
  color: #6CC;
  font-size:100%;
}
```

スッキリ　コード1-2
子孫クラス指定でHTMLを簡略化

HTML

```
<div id="nav">
  <h2>カテゴリー名</h2>
</div>
<div id="section">
  <h2>カテゴリー名</h2>
</div>
```

CSS

```
div#nav h2{
  color: #300;
  font-size: 100%;
}
div#section h2{
  color: #6CC;
  font-size: 100%;
}
```

スッキリ　コード1-3
共通プロパティーをまとめて記述

CSS

```
/* 共通プロパティー */
h2{
    font-size: 100%;
}

/* 独自プロパティー */
div#nav h2{
    color: #300;
}
div#section h2{
    color: #6CC;
}
```

スパゲティCSS解消法②

グループだと心強いです

スッキリ度

CSSでデザインしていると、さまざまなセレクターを扱うことになりますが、セレクターが多いとCSSの行数が増えてスパゲティ状態になりやすくなります。

コード2-1では、h1～h3の見出しにそれぞれプロパティーを定義していますが、よく見てみるとbackground-color、color、paddingにはすべて同じ値が指定されています。こういった場合、コード2-2のようにカンマ(,)で区切って複数のセレクターをグループとして記述し、共通しているプロパティーと値を記述できます。HTMLタグのセレクターだけでなく、クラスやIDを使ったセレクターや子孫セレクターなどもグループ化できます。

複数のセレクターに共通するプロパティーと値の指定はこのようにグループにしてまとめておいて、各セレクターで異なるプロパティーに関しては個々に記述すれば、それだけでもコードの量をずいぶん少なくできます。

また、同じ種類の文書構造はグループで指定できることが多いため、グループで指定しておけば、デザインを変えるときにh1とh2は修正したけれどもh3を修正し忘れたというミスがなくなります。

セレクターのグループ分けで注意しておきたいのがカンマ(,)の付け方です。グループにしたいセレクターごとにカンマを付けるという非常に単純なルールですが、うっかり最後のセレクターのあとにもカンマを付けてしまうと、指定したスタイルが適用されなくなります。

コード2-2ではh1～h3すべてに同じ「#633」の色を指定していますが、h1とh2には同じ色を指定するけれどもh3の色だけを変えたいということもあるでしょう。この場合、コード2-3のようにh1～h3にまとめて「color: #633」を指定しておいて、あとからh3だけ別の色を指定する方法が有効です。これは、同じセレクターの同じプロパティーへの値の指定が複数ある場合、あとから指定されたほうの値が有効になるCSSのルールを利用しています。CSSファイルの中で、グループで指定した値よりもh3単独のスタイル指定を後ろに書く必要があることに注意してください。

基本的なタグとそれに対する基本的なプロパティーは先に書いて、複雑な設定はあとに書くというように記述の仕方に一貫性を持たせると、CSSはすっきりすることでしょう。

スパゲティ　コード2-1
同じ値を複数のセレクターで指定

CSS

```
h1{
  background-color: #ffc;
  color: #633;
  font-size: 120%;
  padding: 0.5em;
}

h2{
  background-color: #ffc;
  color: #633;
  font-size: 110%;
  padding: 0.5em;
}

h3{
  background-color: #ffc;
  color: #633;
  font-size: 105%;
  padding: 0.5em;
}
```

スッキリ　コード2-2
同じ指定はセレクターをグループ化

CSS

```
h1, h2, h3{
  background-color: #ffc;
  color: #633;
  padding: 0.5em;
}

h1{
  font-size:120%;
}
h2{
  font-size:110%;
}
h3{
  font-size:105%;
}
```

スッキリ　コード2-3
指定した値をあとから上書き

CSS

```
h1, h2, h3{
  background-color: #ffc;
  color: #633;
  padding: 0.5em;
}

h1{
  font-size:120%;
}
h2{
  font-size:110%;
}
h3{
  font-size:105%;
  color: #300;
}
```

スパゲティCSS解消法③

backgroundプロパティーでダイエット

スッキリ度

CSSにはたくさんのプロパティーがあるため、1行に1プロパティーという形で書いていくと、CSSファイルはどんどん長くなってしまいます。そこで、まとめて記述できるプロパティーは1行にまとめてしまいましょう。一部のプロパティーは、関連する複数のプロパティーを1行にまとめて(簡略化して)指定できるのです。慣れてくると、簡略化されたプロパティーの書き方のほうが楽に書けるため、こちらの書き方をよく使うことになるでしょう。

簡略化されたプロパティーを記述する方法は、最近多くのユーザーに使われているほとんどのブラウザーで問題なく使えます。しかし、IE 5.0やNetscape 4.xのような古いブラウザーだと簡略化したプロパティーを一部正しく読み込めない場合もあるので、古いブラウザーでも見た目をできる限り同じにしたい場合は簡略化指定は控えたほうがいいかもしれません。

簡略化できるプロパティーとしては、background、margin、padding、border、list-style、fontがあります。それぞれのプロパティーをどのようにしたら短く書けるのか、その記述方法を簡単に紹介します。

まずはbackgroundプロパティーを見てみましょう。背景のスタイルを決めるプロパティーはコード3-1のように5種類あります。これらのプロパティーは、コード3-2のようにbackgroundプロパティーを使って1行にまとめられます。

backgroundプロパティーにどのような順番で値を指定するかの決まりは特にありませんが、background-positionにあたる2つの値は必ずセットにして横軸の位置、縦軸の位置の順番で書く必要があることにだけ注意してください。

コード3-2ではbackgroundプロパティーに指定できる値をすべて列挙しましたが、各プロパティーのデフォルトになる値はわざわざ書く必要はありません。たとえば、background-attachmentのデフォルト値は「scroll」なので、これは省略できます。また、background-positionのデフォルト値は「0 0」なので、この値もbackgroundでの記述では省略できます。このようにデフォルト値を省略すると、コード3-3のようにかなりスッキリします。デフォルト以外の値にしたいときはbackgroundの値に新しく追加すればいいだけなので、メインテナンスが非常に楽になります。

ほとんどのブラウザーで問題なく使えるbackgroundですが、Netscape 4.xでは指定した値がうまく適用されない場合があるので注意しましょう。

スパゲティ

コード3-1
背景のプロパティー5つを個別に指定

CSS

```
#section {
  background-color: #ccc;
  background-image: url(bg.gif);
  background-repeat: repeat-y;
  background-attachment: scroll;
  background-position: 0 0;
}
```

スッキリ

コード3-2
5つの値をまとめて指定

CSS

```
#section {
  background: #ccc url(bg.gif) repeat-y scroll 0 0;
}
```

スッキリ

コード3-3
デフォルト値の指定を省略

CSS

```
#section {
    background: #ccc url(bg.gif) repeat-y;
}
```

スパゲティCSS解消法④

marginとpaddingのプロパティーでダイエット

スッキリ度

marginとpaddingは、コード4-1やコード4-2のように上下左右の値をそれぞれ指定します。これらの値は、marginプロパティーやpaddingプロパティーを使えば、コード4-3のように、それぞれ1行にまとめて記述できます。

値を書く順番は、top→right→bottom→leftと決められています。これは、図3のように上から時計回りに記述していると考えれば覚えやすいでしょう。

コード4-1のように上下左右の値がバラバラの場合は順番に従ってコード4-3のmarginのようにそれぞれの値を書きますが、コード4-2のように上と下の値が同じ、左と右の値が同じの場合は値を4つ書く必要はありません（もちろん記述しても全然問題ありませんが）。コード4-3のpaddingのように上下の値と左右の値のみを書くだけでもきちんとスタイルは適用されます。

また、コード4-4のpaddingプロパティーのように、値を3つだけ指定する書き方もあります。この場合は、上の値、左右の値、下の値の順に指定されていることになります。上下左右すべてが同じ値の場合はコード4-4のmarginプロパティーのように1つの値を記述するだけで全方向に同じ値を指定したことになります。

marginとpaddingの簡略化した表記もbackgroundの場合と同様にNetscape 4.xではうまく動作しないことがあります。また、IE 5.0でもうまく表示されないことがあると言われています。

スパゲティ

コード4-1
上下左右の余白やを個別に指定

CSS

```
#nav{
  margin-top: 10px;
  margin-right: 5px;
  margin-bottom: 20px;
  margin-left: 10px;
}
```

スパゲティ

コード4-2
上下左右の詰めを個別に指定

CSS

```
#header{
  padding-top: 10px;
  padding-right: 20px;
  padding-bottom: 10px;
  padding-left: 20px;
}
```

スッキリ

コード4-3
4方向の余白や詰めをまとめて指定

CSS

```
#nav{
  margin:
      10px 5px 20px 10px;
}

#header{
  padding: 10px 20px;
}
```

スッキリ

コード4-4
一部の方向の値だけを指定

CSS

```
#header{
  padding: 5px 10px 10px;
  margin: 10px;
}
```

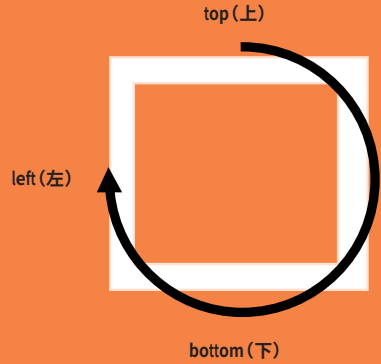

図3　marginやpaddingは上から右回りの順番で指定

スパゲティCSS解消法⑤

list-styleプロパティーでダイエット

スッキリ度

リスト（箇条書き）のスタイルを決めるプロパティーは、コード5-1のようにlist-style-image、list-style-type、list-style-positionの3つがあります。

これらの値は、list-styleプロパティーを使ってコード5-2のように1行にまとめることができます。backgroundプロパティーのときと同様に、値を書く順番は特に決まっていませんし、デフォルト値を使う場合は省略することもできます。

箇条書き項目の左側にデフォルトで表示される黒い丸の代わりになるマーカーを定義するプロパティーとしてlist-style-image（アイコン画像を指定）とlist-style-type（標準で定義されているマークを指定）がありますが、画像とマークの両方が指定されていると、どのような順番で書かれていても画像の指定が優先されます。つまり、コード5-2のように記述した場合、指定された画像が読み込めないときにだけ「circle」が使われます。

スパゲティ解消法②で説明したように、CSSでは、プロパティーの指定が複数あってそれぞれ異なる値を指定している場合は、あとに書かれた値が優先されるというルールがあります。このルールは、複数のプロパティーを1つにまとめて指定する場合でも同じで、まとめて指定しているプロパティーの一部を別の指定で上書きできます。

たとえば、コード5-3では、どのulタグにでも共通するスタイルをまず指定しておいて、そのあとにul.specialのスタイルを改めて指定しています。このようにすれば、ul.specialではマーカーの指定（list-style-imageとlist-style-typeにあたる値）だけが上書きされ、それ以外のプロパティー（list-style-positionにあたる値）はul共通のスタイルがそのまま使われます。

スパゲティ

コード5-1
箇条書きのプロパティーを個別に指定

CSS

```
ul{
  list-style-image: url(mark.gif);
  list-style-type: circle;
  list-style-position: inside;
}
```

スッキリ

コード5-2
箇条書きのプロパティーをまとめて指定

CSS

```
ul{
  list-style: url(mark.gif) circle inside;
}
```

スッキリ

コード5-3
まとめて指定した値の一部を上書き指定

CSS

```
ul{
  list-style: url(mark.gif) circle inside;
}

ul.special{
  list-style: url(mark2.gif) square;
}
```

ただし、IEではこの書き方で問題ないのですが、MozillaやFirefoxなどでは、あとに書かれたlist-style全体の指定が前の指定全体を上書きしてしまうため注意が必要です。たとえばコード5-3の書き方では、ul.specialのlist-styleでは、list-style-positionにあたる値が指定されていないため、前に指定した「inside」ではなくデフォルトの値が使われてしまうのです。

スパゲティCSS解消法⑥

fontで簡略プロパティー指定

スッキリ度

文字関連のプロパティーはコード6-1のようにたくさんあるため、細かく指定していくと、どんどんCSSファイルが長くなります。もちろん文字に関するプロパティーはここに示したもの以外にもいくつかあるのですが、コード6-1で指定されているプロパティーは、fontプロパティーを使えば、コード6-2のように1行にまとめて記述できます。backgroundプロパティーと同様に、値を書く順番は基本的に自由ですが、守らなければならないルールが2つだけあります。それはfont-sizeとfont-familyにあたる値は必ず記述する必要があり、しかもfontプロパティーに指定する値の最後にfont-size→font-familyの順に記述しなければいけないことです。このルールを守らないとfontに記述したすべての値が適用されないので注意が必要です。他の値は順番を変えたり省略したりできます。

コード6-2にある「95%/180%」は最初の値がfont-sizeで次がline-heightの値です。line-heightは省略できますが、記述する場合にはfont-sizeとセットにして「/」で区切って記述します。

スパゲティ

コード6-1
フォント関連の指定を個別に記述

CSS

```
p.specific{
  font-style: italic;
  font-variant: normal;
  font-weight: bold;
  font-size: 95%;
  line-height: 180%;
  font-family: "ＭＳ Ｐゴシック", "ヒラギノ丸ゴ Pro W4",
               HiraMaruPro-W4, Osaka, sans-serif;
}
```

スッキリ

コード6-2
フォント関連の指定をまとめて記述

CSS

```
p.specific{
  font: italic normal bold 95%/180% "MS Pゴシック",
       "ヒラギノ丸ゴ Pro W4", HiraMaruPro-W4, Osaka, sans-serif;
}
```

「継承」を利用しよう

CSSには、「継承」という便利な仕組みがあります。要素に親子関係がある場合、親要素に指定されたスタイルは、何も指定しなくても子要素にも適用されるようになっています。これが継承です。たとえば、次のようなHTMLがあるとします。

```
<p>CSSでは、デザインの基本となる<span
class="keyword">ブロックモデル</span>
を理解することが大切です。</p>
```

keywordクラスのspan要素はp要素の子要素になっています。この場合、CSSでp要素のスタイルとしてフォントサイズや色が指定されていれば、その指定はp要素の子要素であるspan.keywordにも自動的に適用されます。このため、p要素のスタイルに指定しているプロパティーは、わざわざspan.keywordのスタイルとして改めて指定する必要はありません。

継承をうまく使えば、不必要なCSSの記述を減らせて便利なのですが、注意が必要な場合もあります。色や枠線の太さなどならば親要素に指定したのと同じものを子要素に指定しても問題ないのですが、サイズ指定は重ねて使ってはいけません。たとえば、親要素でfont-size: 90%を指定している場合、子要素にも同じfont-size: 90%を指定してしまうと、継承のために子要素は結果的に90%×90%＝81%相当のサイズになってしまうのです。

スパゲティCSS解消法⑦

borderで簡略プロパティー指定

スッキリ度

デザインにメリハリを付けるために、ボックスに枠線を付けることも多いでしょう。枠線のプロパティーはコード7-1のようなものがありますが、これはコード7-2のようにborderプロパティーにまとめて1行で書くことができます。

上下左右すべて同じ枠線を使いたい場合はこれでいいのですが、4方向で違う枠線を使いたい場合は少し事情が異なります。marginやpaddingでは上下左右の値を1つのプロパティーでまとめて指定できましたが、borderでは上下左右についてそれぞれ3つの値を指定できるため、1つのプロパティーで4方向の枠線に別々の指定をすることはできません。この場合、すっきりとCSSを指定するには2つの方法があります。1つは、border-topやborder-leftなどそれぞれの方向に対応するプロパティーにそれぞれ3つの属性の値を指定する方法（コード7-3）、もう1つは、border-widthやborder-styleなどの属性のプロパティーにそれぞれ4方向の値を指定する方法（コード7-4）です。4方向の枠線がバラバラになることは少ないでしょうから、コード7-4のようにするほうがCSSファイルに書く内容は減りますが、あとからメインテナンスすることを考えると、コード7-3のように書くほうがわかりやすいでしょう。

スパゲティ

コード7-1
枠線関連の指定を個別に記述

```
.side{
  border-width: 1px;
  border-style: solid;
  border-color: #633;
}
```

スッキリ

コード7-2
枠線関連の指定をまとめて記述

```
.side{
  border: 1px solid #633;
}
```

スッキリ

コード7-3
上下左右に違った枠線を指定（方向別）

```
.side{
  border-top: 2px solid #ccc;
  border-right: 1px dotted #666;
  border-bottom: 1px solid #999;
  border-left: 2px dotted #666;
}
```

スッキリ

コード7-4
上下左右に違った枠線を指定（属性別）

```
.side{
  border-width: 2px 1px 1px 2px;
  border-style: solid dotted;
  border-color: #ccc #666 #999;
}
```

自分なりのルールを作ろう

今回紹介したのはCSSをスマートに書くための方法の一部にしか過ぎませんが、基本的なところはカバーしています。

ここで紹介したプロパティーの書き方を実行するだけでもコードの行数はかなり減り、見通しのいいCSSファイルにできるはずです。このように細かいルールは存在するものの、CSSには「こういう順番でこう書かなければならない」という絶対的なルールが存在しません。だからと言って場当たり的にCSSを書いていると、すぐにスパゲティ化してしまいます。CSSを書くときには、自分なりのルールを作り、一貫性をもったコードを書く必要があります。スマートにコードを書くことにより、あとでどこに何を書いたのかがわからなくなることを防げるのです。

［インターネットマガジン　バックナンバーアーカイブ］ ご利用上の注意

このPDFファイルは、株式会社インプレスR&D（株式会社インプレスから分割）が1994年～2006年まで発行した月刊誌『インターネットマガジン』の誌面をPDF化し、「インターネットマガジン　バックナンバーアーカイブ」として以下のウェブサイト「All-in-One INTERNET magazine 2.0」で公開しているものです。

http://i.impressRD.jp/bn

このファイルをご利用いただくにあたり、下記の注意事項を必ずお読みください。

- 記載されている内容（技術解説、URL、団体・企業名、商品名、価格、プレゼント募集、アンケートなど）は発行当時のものです。
- 収録されている内容は著作権法上の保護を受けています。著作権はそれぞれの記事の著作者（執筆者、写真の撮影者、イラストの作成者、編集部など）が保持しています。
- 著作者から許諾が得られなかった著作物は収録されていない場合があります。
- このファイルやその内容を改変したり、商用を目的として再利用することはできません。あくまで個人や企業の非商用利用での閲覧、複製、送信に限られます。
- 収録されている内容を何らかの媒体に引用としてご利用する際は、出典として媒体名および月号、該当ページ番号、発行元（株式会社インプレスR&D）、コピーライトなどの情報をご明記ください。
- オリジナルの雑誌の発行時点では、株式会社インプレス R&D（当時は株式会社インプレス）と著作権者は内容が正確なものであるように最大限に努めましたが、すべての情報が完全に正確であることは保証できません。このファイルの内容に起因する直接的および間接的な損害に対して、一切の責任を負いません。お客様個人の責任においてご利用ください。

このファイルに関するお問い合わせ先
株式会社インプレスR&D
All-in-One INTERNET magazine 編集部
im-info@impress.co.jp