Movable Type

我流のカスタマイズはもう卒業

特集2

プロが教えるMovable Typeの構造デザイン

Business Architects Inc.

text/template & css design :
森田 雄
visual design :
荻野 智成／増尾 金弥
layout adjustment :
小久保 浩大郎

株式会社ビジネス・アーキテクツは、企業のウェブサイトを通じた情報発信を助ける、ウェブのプロ集団である。
URL http://www.b-architects.com/

本誌用に書き下ろしデザイン！

記事オリジナルのテンプレートとCSSをダウンロード

この記事で作ったテンプレートファイルとCSSファイルがダウンロードできます。
URL http://internet.impress.co.jp/im/200311mt/
誌面の都合上解説では一部のコードを省略していますので、ダウンロードしたテンプレートファイルを参照しながら読むとより理解しやすくなるでしょう。

インターネットマガジン／株式会社インプレスR&D
©1994-2007 Impress R&D

Part.1

構造も文法もよろしくないMTテンプレートのXHTML
正しくないテンプレートからは正しい出力はされない！

ページテンプレートとCSSで内容や見栄えを制御

最近流行のMovable Type（以下「MT」）は、細かいXHTMLの文法などを気にしなくても、エントリー（記事内容）を入力すれば一定の構造と見栄えのページ（図1）が生成される「ウェブログ」ツールだと言われている。ウェブログはコンテンツが重要だとはよく言われるが、やはりその見栄えも気になることだろう。

入力されたエントリーをどのように処理してページを作るかを決めるのに「テンプレート」という仕組みが使われていることは、MTを使っている人ならよく知っているはずだ。XHTMLでウェブページを出力する部分だけでなく、ページの見た目を決めるスタイルシートや、ウェブログならではのRDF／RSSによるメタデータファイルも、やはりテンプレートから生成されている。

MTでは、ページに対応したテンプレート（たとえばMain Indexテンプレートはトップページの内容を決める）は、ページの内容をどのような構造のXHTMLで出力するかを定義し、Stylesheetテンプレートはそのまま文書の構造をページとして「どう見せるか」を定義する。ページに対応したテンプレートはMTタグと呼ばれる独自のテンプレートタグを含んだXHTMLで作られているが、Stylesheetテンプレートの内容は単なるCSS（Cascading Style Sheets）だ。

たとえばウェブログサイトの顔とも言えるMain Indexテンプレートのページの構成要素が標準の状態のままでいいのであれば、あとはStylesheetテンプレートを変更すれば、その見栄えをどうにでもできるというのが一般的な認識だろう。しかし注意しなければいけないのは、どのようなStylesheetテンプレートにしても、Main Indexテンプレートなどが生成するページの構成要素は変わらないということだ。

土台のテンプレートあなたのは「正しい」？

皆さんも、MTのデザインを変更しようと思って、「面倒くさい」だとか「難しい」だとか感じたことはないだろうか。筆者は前述のとおりのことを考えていたので、Stylesheetテンプレートだけを書き換えられれば、つまりはCSSでのデザインやレイアウトというものがわかってさえいれば自由自在だと思っていた。しかし、実際のところ、Stylesheetテンプレートの変更だけでは設計どおりのデザインにはならず、もどかしい思いをすることになった。

さて、これはどういうことなのだろう。なんといっても大前提として、CSSはまず文書構造（XHTMLページの構造）ありきなのだ。CSSをうまく適用するには、適当に組み立てたXHTML文書ではなく、正しいとか妥当とかいう表現の似合う文書構造を持ったXHTMLが求められるわけだ。あくまで構造の外観しか定義できないCSSでは、Main Indexテンプレートが生成する文書構造を正しく直すことができない。つまり、Stylesheetテンプレートで思いどおりにデザインするには、Main Indexテンプレートで文書構造を適切に設計しておく、すなわち構造をデザインしておく必要があるのだ。

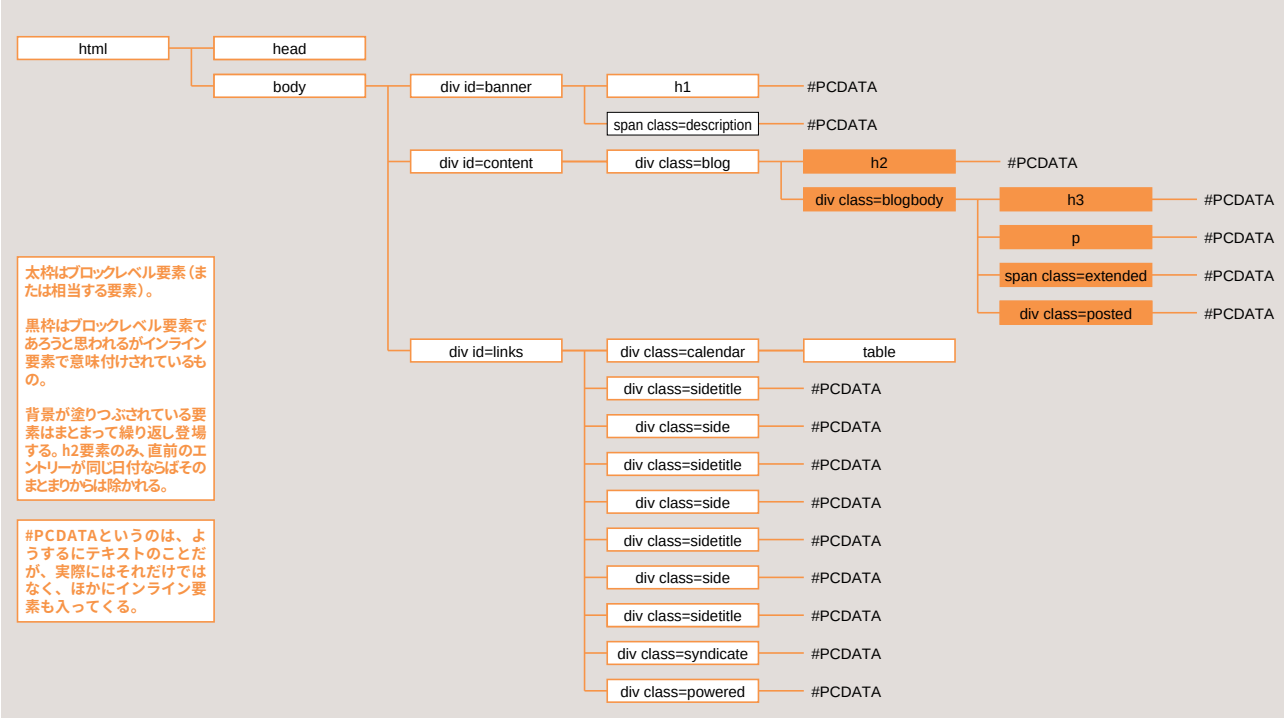
というわけで、標準のMain Indexテンプレートが生成する文書構造を、まあこんなものだろうという辺りまでの粗さで図にしてみた（図2）。正直な感想を言うと、「目も当てられない状態」だ。これをどうにかして妥当なものにしなければならないの

図1　MTの標準テンプレートから出力されるページ

標準のMovable Typeテンプレートで作られたページは一見問題ないように見えるが……。

©1994-2007 Impress R&D

図2　MTの標準のMain Indexテンプレートの構造ツリー図



標準のMain Indexの構造は冗長で複雑だ。

で、まずは現状を把握しよう。どこがよろしくないのか、代表的なものをざっと挙げてみるので、MTの標準Main Indexテンプレートと図2、図3を参照しながら読み進めてほしい。

標準MTテンプレートの問題点：div要素の階層構造がよろしくない

div要素というのは、汎用的なブロックレベルの要素で入れ子にすれば階層構造も作れるものだ。まずdiv id=banner要素を見てみよう。このdiv要素の中には、ウェブログの名前であるh1要素と、ウェブログの説明がspan class=description要素として入っている。ウェブログの名前はそのページ全体に影響を及ぼしている一番大きな見出しなので、h1要素なのは問題ない。しかし、となるとdiv id=banner要素が、ウェブログの名前と説明だけを内包したところで終了してしまうのはおかしなことになってくる。h1要素を大見出しだとすると、h2要素は中見出し、h3要素は小見出しだ。これは親子関係であると考えられ、たとえば部・章・節の各見出しのような関係だと言える。div id=content要素内にあるh2要素やh3要素は、それぞれ確かに見出しであると考えられるが、h1要素とブロックレベルで違う構造になってしまっている。親要素のbody要素から見たとき、h1要素を内包するdiv id=banner要素とh2／h3要素を内包するdiv id=content要素とが、並列の関係になってしまっているわけだ。これは、文書構造としてあまりいい関係だとは言えない。

また、div id=content要素の直下にdiv class=blog要素があるのも変だろう。だいたい、div id=content要素の中にはdiv class=blog要素以外のものが何もないのだから、このどちらかのdiv要素は余計なブロックレベル要素であると言える。

これらの階層構造を文書構造として適切なものにしなければいけない。

標準MTテンプレートの問題点：関係あるものが並列の構造になっている

カレンダーやリンク集が入っているdiv id=links要素だが、ここも階層構造の問題点と同様に特にひどい部分だ。

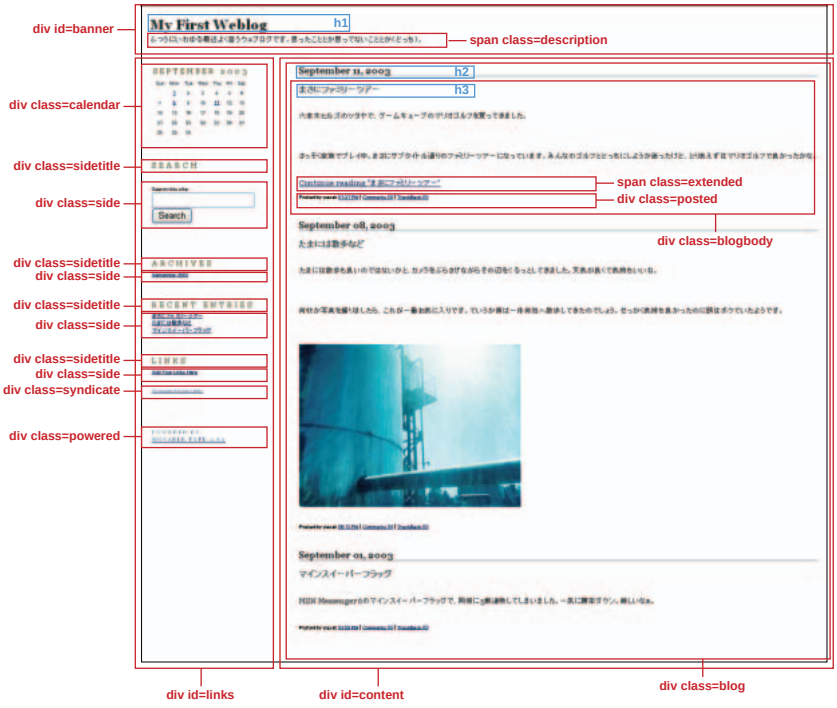
たとえば標準では検索フォームの入力フィールドの上には「Search」という文字列（入力フィールドに対する見出し）が表示されるが、その文字列はsidetitleというクラス名を備えたdiv要素で意味づけされているのだ。入力フィールドはsideというクラス名を備えたdiv要素として意味づけされている。つまり、sideクラスの内容に対する見出しのようなものだから、sidetitleクラスだというわけなのだろう。

この2つのdiv要素は意味的にはセットだが、構造上は並列扱いであってセットだという意味づけがなされていないのが問題だ。

同様の組み合わせが、続く月別アーカイブ一覧である「Archives」、最近のエントリー一覧の「Recent Entries」、そして好きなリンクを追加できる「Add Your Links」のリンク一覧にも現れる。上からこの順に並ぶように表示することしか頭にないのならともかく、たとえば月別アーカイブ一覧のタイトルと内容をほかのところへ表示したいときなどは、2つのブロック

インターネットマガジン／株式会社インプレスR&D
©1994-2007 Impress R&D

図3　標準のMain Indexテンプレートの出力と構造の関係



複雑な構造のために、スタイルシートを作るのも一苦労だ。

レベル要素をそれぞれ動かさなければならない。そもそも、タイトルと内容が全然違う場所に配置されるということは考えられないので、これらのdiv要素はそれぞれ、対であることが明確な要素として意味づけしておくべきものだ。

標準MTテンプレートの問題点：きちんと意味づけされていない

div要素というのは、特に何だという意味づけであるという定義がなされていない、汎用的なブロックレベルの要素だと説明した。使いどころはと言うと、XHTMLが備えている各要素ではうまく意味づけしきれない場合や、要素間の結びつきを明示してやろうという場合などが考えられる。ブロックレベルではなくインライン要素の汎用的なものとしては、span要素がある。こちらはブロックレベルではないために、要素間の結びつきを明示することには使いづらい。span要素を使うことがある場面と言えばp要素の中だろう。もっと明確な意味づけをしたいときなどに、p要素中の該当箇所でspan要素を使ったりする。

となると、div id=banner要素の中にあるウェブログの説明の部分が、span class=description要素だというのはいただけない。ウェブログの説明は1つの段落を成すものと考えられるので、ブロックレベルの要素であるp要素とするのが順当だ。

さらに、div class=blog要素には、h2要素とdiv class=blogbody要素が入っており、div class=blogbody要素にはh3、p、span class=extended、div class=postedという各要素が入っている。div class=blog要素とその内容までは問題なさそうだが、div class=blogbody要素の中身の後半2つはあまりよろしくない。span class=extended要素の内容は「この記事の続きを読む」といったような文章になるはずなので、ここにはp要素を使うのが適切だろう（少なくともspan要素ではない）。div class=posted要素も、含まれる内容を考えるとdiv要素単独だというのはおかしいだろう。

そして、先ほども触れたdiv id=links要素の中身だが、ここはdiv要素の構造がよろしくないだけでなく、内包しているほとんどのデータをそのままdiv要素でもって意味づけしてしまっている。XHTMLで定義済み要素に適切なものがないというのならともかく、定義済みの要素で適したものがあるのならば、それらで意味づけすることで、要素を適切に明示できる。困ったときのdiv要素というのもアリだが、困っていないならdiv要素だけで意味づけするのではなく定義済みの要素を使うべきだ。div class=sidetitle要素とdiv class=side要素の対などは、たとえばdl要素を使うとすっきりする。

XHTMLの文法とHTMLの文法は違うものである

MTの標準では出力にXHTMLを使う。XHTMLは文法が厳格であり、正しく書かなければXML文書としての利用ができなくなってしまう。最低限の文法は覚えてほしい。

- 要素名と属性名はすべて小文字
「<TABLE>」ではなく「<table>」とするのが正しい。大文字と小文字が区別されるので、別のタグだと見なされるのだ。
- 終了タグを省略できない
HTML 4では省略を許されていた終了タグを省略してはならない。
- 空要素タグを終結させる
たとえば<img />や<hr />というようにタグを「/>」で終わらせる必要がある。
- 属性値は引用符で囲む
<td colspan=3>のような属性値は、XHTMLではすべて<td colspan="3">のように引用符で囲むのが正しい。
- 属性を最小化しない
XHTMLは属性の最小化を許容しないため、たとえば<dl compact>ではなく、<dl compact="compact">と書く。

また、これは文法ではないが、XHTML文書はXML文書でもあるため、文書で使っているXMLのバージョンや文字コードを示すXML宣言が文書の先頭に必須だ。

©1994-2007 Impress R&D

Part.2

コレが「正しい」構造のテンプレートだ!

意味づけを考えて ゴチャゴチャしない理想の形に

テンプレートをどう直したら「正しく」なる?

簡単に言えば、Part 1で述べたところを直せばおおむね正しくなる。div要素をはじめとするブロックレベル要素の階層構造に、もっと一貫性を持たせるわけだ。その上で、div要素やspan要素で意味づけするという横着をせずに、可能な限りXHTMLで定義済みの各要素で意味づけしていけばOKだ。

さらに、標準のMain Indexテンプレートでは、たとえばXML宣言がないだとかいう、そもそもXHTMLの文法的な誤りもあるので、その辺りも併せて直しておくのが「正しい」だろう。

というわけで修正後の要素の階層構造のイメージ図を用意したので、ひとまず見てほしい(図4)。これは、だいたいの完成予想図だと思ってもらいたい。

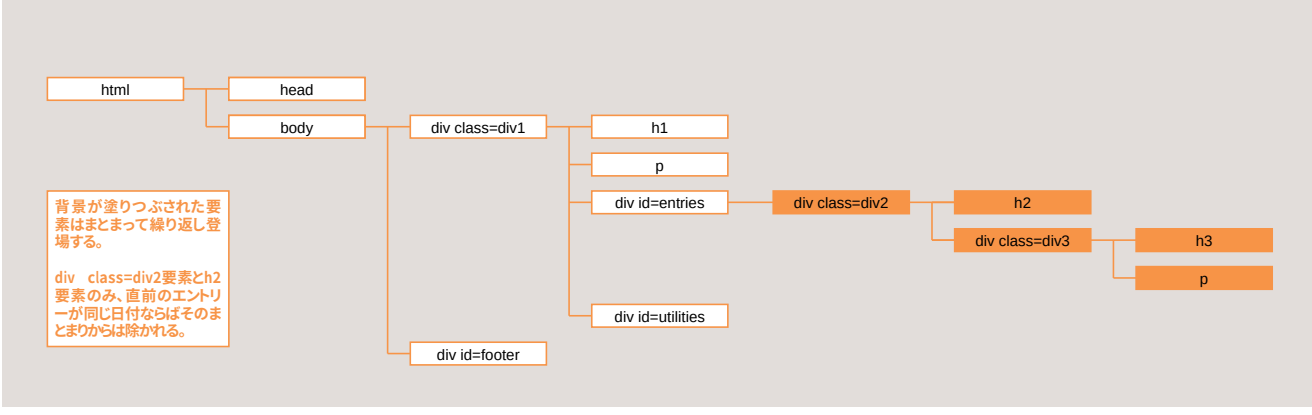
- まず、ページ全体に影響力のあるh1要素の大見出しと、それに続く内容をdiv class=div1要素という大きなブロックレベル要素でくくり、h1要素を大見出しとする内容なのだと主張させる。
- 続いて、中見出しのh2要素から始まるブロックと小見出しのh3要素から始まるブロックを、順にネストしたブロックレベル要素として、div class=div2、div class=div3とする。これは、いわばISO-HTMLやXHTML 2.0の見出しとセクションによる包含関係に近い形で、階層構造を明確にしていくものだ。
- div class=div2要素以下は、各エントリーが相当する。エントリーは複数あるので、これらをまとめてdiv id=entries要素で囲むことにより、エントリー群であることをより明確にすると良さそうだ。
- div id=utilities要素が内包する各要素の具体例は、イメージ図からはひとまず省略している。実際には、カレンダーや月別アーカイブ一覧といった、明らかにページの内容を構成する要素だけども、伝えたいコンテンツとしての主体ではなく、むしろ機能を提供しているような要素群をdiv id=utilities要素でまとめる。
- div id=entries要素とdiv id=utilities要素は、div class=div1要素の中で、h1要素とまったく対等の位置づけとなっているが、これはh1要素の大見出しはページ全体に影響を及ぼすものだから、各div要素の中に入る要素はh1要素より下の階層に位置づけられるだろうという判断による。
- 一方、div id=footer要素はdiv class=div1要素と同じ階層にある。ここはXHTMLで言うとaddress要素のようなものが入るようなブロックとして用意してある。このような階層のブロックは、ページ全体に影響を及ぼす大見出しのような要素の下に並ぶようなデータではなく、むしろサイト全体に一貫して挿入される、たとえば作者名などといったデータを入れるのに適している。

これが「正しい」テンプレートだ!

もうウンチクはたくさん、早く「正しく」直したMain Indexテンプレートを見せてくれという声が聞こえる頃合だろうから、先に新生Main Indexテンプレートの全貌を明かす。今まで述べてきたことすべてを踏まえた結果、図5のようになった。

まず注意点だが、id属性とclass属性の

図4 新生Main Indexテンプレートの構造ツリー図



新生Main Indexはこういう構造にする。100ページの構造と比べてみるとかなりシンプルになっている。

インターネットマガジン／株式会社インプレスR&D
©1994-2007 Impress R&D

値には、すべて「mt」という接頭辞を付けている。これは他のテンプレートやら既存のセレクターやらと競合してしまう可能性を排除するために、あえて付けた。たとえば、図4でdiv class=div1要素を意図していたものは、この新生Main Indexテンプレートでは「<div class= ”mtDiv1 ”>」となっている。これ以降は、この「mt」接頭辞つきの属性値で説明していく。

問題点を改めて「正しく」意味づけ

まず、全体のdiv要素の階層構造を、前に述べたように適切に直してある。

実際にエントリーの内容となるdiv class=mtDiv3要素は、エントリー本文と投稿者や投稿した時間、エントリーに付いたコメントとトラックバックの情報も記されるところだ。ここを、エントリー本文とそれ以外というように大きく分けて、エントリー本文のp要素群はdiv class=mtEntryBody要素でくくり、それ以外の部分をdiv class=mtEntryFooter要素でくくっている。div class=mtEntryFooter要素の中は、「何時にだれが投稿した」「コメントがいくつ付いている」「トラックバックが何個あった」という情報が単に列挙されているものと考えて、ul／li要素で意味づけしている。

先ほど詳細を省いていた、div id=mtUtilities要素の中身も大幅に変わった。この要素の中身は、標準のMain Indexテンプレートで一番困ったところとされていた、セットを想定しているとしか思えないdiv class=sidetitle要素とdiv class=side要素のまとまりだ。これらは対であることをより明確にするために、それぞれdl／dt／dd要素を用いて意味づけし直している。div class=sidetitle要素はdt要素に変更され、div class=side要素だったものはdd要素となった。dt要素とdd要素は、dl要素によって対の関係が明示される。

また、div class=side要素の内容は、箇条書き的なデータが入ることが想定されているにもかかわらず、標準の状態ではbr要素で1行ごとに改行していくだけのものだった。これはせっかく箇条書きの内容なのだから、ul要素で正しく意味づけしなければならない。つまりdd要素の内容にul要素が入る形になる。もちろん、箇条書きだから、ul要素の中にli要素が必要な数だけ生成されるというわけだ。

検索フォームのところは、fieldset要素やlegend要素を使うようにしてみた。

図5　新生Main Indexテンプレートのコード概要

```
<?xml version="1.0" encoding="<$MTPublishCharset$>" ?>          XHTMLとして必要な定義を正しく入れる
<!DOCTYPE html ……文書型宣言またはDOCTYPE宣言……>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja" lang="ja" dir="ltr">
<head profile="http://purl.org/net/uriprofile/">                head要素に必要な情報を正しく入れる
……Content-Typeなどの必要なMeta要素を入れる……
<title><$MTBlogName$></title>
……スタイルシートやRSSはlink要素で記述……
……JavaScriptやCCライセンスもhead要素内で……
</head>
<body>
<div class="mtDiv1">                                            mtDiv1：ページの根となるブロック
  <h1><$MTBlogName$></h1>
  <p><$MTBlogDescription$></p>
  <div id="mtEntries">                                          mtEntries：記事部分全体のブロック
    <MTEntries><$MTEntryTrackbackData$>
    <MTDateHeader>
      <div class="mtDiv2">                                      mtDiv2：日付ごとのかたまりのブロック
      <h2><$MTEntryDate format="%x"$></h2>
      </MTDateHeader>
        <div class="mtDiv3">                                    mtDiv3：各記事のブロック
        <h3><a name="blog<$MTEntryID pad="1"$>" id="blog<$MTEntryID pad="1"$>"><$MTEntryTitle$></a></h3>
          <div class="mtEntryBody">                             mtEntryBody：記事と記事の追記部分
            <$MTEntryBody$>
            <MTEntryIfExtended>……追記部分……</MTEntryIfExtended>
          <!--/mtEntryBody--></div>
          <div class="mtEntryFooter">                           mtEntryFooter：投稿者、コメント、トラックバックのブロック
            <ul>
              <li class="mtPosted">Posted by …… at ……</li>
              <MTEntryIfAllowComments>
              <li class="mtComments">……Comments ……</li>
              </MTEntryIfAllowComments>
              <MTEntryIfAllowPings>
              <li class="mtTrackback">……TrackBack ……</li>
              </MTEntryIfAllowPings>
            </ul>
          <!--/mtEntryFooter--></div>
        <!--/mtDiv3--></div>
       <MTDateFooter>
      <!--/mtDiv2--></div>
    </MTDateFooter>
    </MTEntries>
  <!--/mtEntries--></div>
  <div id="mtUtilities">                                        mtUtilities：カレンダー、検索、リンクなど記事以外のページ構成要素
    <table …… class="mtTableCalendar">
    ……カレンダーの内容……
    <form action="<$MTCGIPath$><$MTSearchScript$>" method="get">
      ……検索フォームの内容……
    </form>
    <dl>
      <dt>Archives</dt>
      <dd><ul><MTArchiveList archive_type="Monthly">
        <li><a href="<$MTArchiveLink$>"><$MTArchiveTitle$></a></li>
      </MTArchiveList></ul></dd>
    </dl>
    ……同様にdlとulを使って「最近のエントリー」や「リンク」を入れる……
  <!--/mtUtilities--></div>
<!--/mtDiv1--></div>
<div id="mtFooter">                                             mtFooter：ページ全体のフッター部分
  <p>Powered by <a href="http://www.movabletype.org/">Movable Type <$MTVersion$></a></p>
<!--/mtFooter--></div>
</body>
</html>
```

誌面の都合上すべてはお見せできないが、階層構造と論理的な意味づけを見直してXHTMLの文法を正しくした新生Main Indexはこうなる。完全なコードについてはダウンロードした内容を確認してほしい。

レイアウト情報はXHTMLから排除する

そして、カレンダーの部分は、col要素を書き足したり、th要素のabbr属性をちゃんと略称にしてみたりしたほか、本来はCSSで定義するべきレイアウト情報をXHTMLから排除している。

標準のMain Indexテンプレートでは、カレンダー周辺は図6のように書かれてい

完全版のテンプレートをダウンロード！ URL http://internet.impress.co.jp/im/200311mt/

©1994-2007 Impress R&D

る。このうちレイアウト情報はどれかと言うと、div要素やth／td要素に書かれているalign属性、table要素に書かれているsummary以外の各属性、td要素の中身が空だったときに挿入されるように書かれている「 」という値だ。

HTMLのalign属性でセンタリングしなくても、CSSのtext-alignプロパティーや左右マージンを同じにするなどでセンタリングできる。table要素のボーダーサイズやセル間の幅なども、それぞれ対応するプロパティーがCSSに用意されている。そして、td要素の中身が空だとそのセルの周囲にボーダーが引かれないことへの対策で「 」を値として入れておくという、何だかレガシーな技が使われているが、これはCSSのborder-collapseプロパティーにcollapseという値を、empty-cellsプロパティーにshowという値をそれぞれ設定することで解消できるのだ。だから「 」を入れる必要はない。

また、これだけではレイアウトのためなのか判断できないとはいえ、td要素の中にいちいち入っているspan class=calendar要素にしても、それはtd要素にスタイルを付ければいいだけの話であるからして、要するに冗長であるし意図がいまひとつ不明瞭だ。そこでspan class=calendar要素のようなものは排除してある。

最後に、div id=mtFooter要素には、XHTMLのaddress要素を入れることを目論んでいたのだが、今回紹介しているこの新生Main Indexテンプレートは筆者だけのテンプレートではないので、address要素の中身に適切な値を用意できない。別に筆者の連絡先を突っ込んでおいてもいいのだが、新生Main Indexテンプレートの外観をデザインするときは、基本的にStylesheetテンプレートだけをいじりたいと思っているためだ。とりあえず、movabletype.orgへのリンク情報を入れる場所として使用することにしておいた。

図6　標準のカレンダーに使われているXHTMLコード

```
<div align="center" class="calendar">
<table border="0" cellspacing="4" cellpadding="0" summary="Monthly calendar with links to
each day's posts">
<caption class="calendarhead"><$MTDate format="%B %Y"$></caption>
<tr>
<th abbr="Sunday" align="center"><span class="calendar">Sun</span></th>
<th abbr="Monday" align="center"><span class="calendar">Mon</span></th>
<th abbr="Tuesday" align="center"><span class="calendar">Tue</span></th>
<th abbr="Wednesday" align="center"><span class="calendar">Wed</span></th>
<th abbr="Thursday" align="center"><span class="calendar">Thu</span></th>
<th abbr="Friday" align="center"><span class="calendar">Fri</span></th>
<th abbr="Saturday" align="center"><span class="calendar">Sat</span></th>
</tr>
<MTCalendar>
<MTCalendarWeekHeader><tr></MTCalendarWeekHeader>
<td align="center"><span class="calendar">
<MTCalendarIfEntries><MTEntries lastn="1"><a
href="<$MTEntryPermalink$>"><$MTCalendarDay$></a></MTEntries></MTCalendarIfEntries>
<MTCalendarIfNoEntries><$MTCalendarDay$></MTCalendarIfNoEntries>
<MTCalendarIfBlank>&amp;nbsp;</MTCalendarIfBlank>
</span></td>
<MTCalendarWeekFooter></tr></MTCalendarWeekFooter></MTCalendar>
</table>
</div>
```

標準のMain Indexに使われているカレンダーのコードには、文書構造だけでなくレイアウトデザインのための指示が多く入っていて、しかも無駄が多い。ダウンロードした新生Main Indexのカレンダー部分のコードと見比べてほしい。

divタグのid属性とclass属性はどう使い分ける？

ページ内にその属性値が一度しか出てこないならid属性を、何度も出てくるのならclass属性を使うというような説明をよく目にするかもしれない。しかしこの説明は間違いでこそないものの、適切なものであるとは言えないだろう。

id属性は、その要素に固有の名前を与えるために使う。class属性は、その要素がどういった属性というか分類なのかというような区別のために使われる。たとえば「森田　雄」という筆者の名前だが、これはまさにid属性的なものだ。そして「森田　雄」というidを持つ筆者は、たとえば「サラリーマン」だとかいうclass属性的な値を持っていたりもするのだ。この名前をh1要素にする場合があったとしたら、前者は「<h1 id="MoritaYu">森田　雄</h1>」となり、後者は「<h1 class="salaryman">森田　雄</h1>」というようになる。

ちなみに、1つの要素に対してid属性とclass属性を併用することもできる。一度なのか何度もなのかということだったら、何らかの値がそのページ内に出てくる回数によってどちらの属性にするかという話になってしまうので、併用するという発想が出てこないかもしれない。しかし固有の名前を持ちつつ何らかの分類に属する要素というのはありえるし、そしてそのように記述できるのだ。さっきの例で言えば「<h1 id="MoritaYu" class="salaryman">森田　雄</h1>」になるということだ。

また、CSS的な側面から言うと、id属性やclass属性はセレクターとしての重要な役割を担っている。このとき、id属性のほうがスタイル適用の優先順位が高くなる特徴もある。最終的な優先順位の決定は、文脈セレクターや!importantキーワードがいろいろ絡んでくると、その計算式が煩雑になってきて、id属性のほうが絶対的にclass属性よりも強い結果になるのかと言うと、そうでもない場合もあるだろう。しかし現状のブラウザーのCSSの実装を考えれば、というよりもシェアが広いウィンドウズ版IE 6などの実装状況からすると、そもそも複雑な文脈セレクターを記述したCSSを用意できないという理屈もある。したがって、だいたいid属性が絡んだセレクターのほうが優先順位が高いと考えておいて、あまり問題はないだろう。

インターネットマガジン／株式会社インプレスR&D
©1994-2007 Impress R&D

3

確かな土台ができたらカッコいいデザインを描こう!

ウェブのプロ集団ビジネス・アーキテクツはこうデザインした

構造はできた 次はデザインだ!

さあ、めでたく新生Main Indexテンプレートができあがった。これで思う存分、デザインに取り掛かれるというわけだ。もちろん、CSSによるデザインやレイアウトに慣れていないとちょっと難しいと思うかもしれないが、ここではある程度慣れているという前提で進めさせてもらう。

デザインの手順だが、少しずつStylesheetテンプレートを修正しては表示して、といった感じではなく、新生Main Indexテンプレートが生成する文書としての各要素をどのように配置していったらカッコいいか、読みやすいか、そういうノリでやっていくといいだろう。コンテンツの構成要素を取り出して、それらを肴に、MTでどうなるとかにはこだわらずにデザインしてしまおうということだ。

ということでデザインしてみた。今回は2つのデザインを用意した(図7、図8)ので、それぞれについて解説していこう。ここでも誌面の都合でスタイルシート本体は紹介できないが、ダウンロードしたものと比べながら読み進めてほしい。

図7　オリジナルデザイン1

新生Main Indexに合わせてオリジナルのスタイルシートを作った。空をイメージしたページデザインだ。必要な情報は失われていないのがわかるだろう。各エントリーの横にある投稿者／コメント数／トラックバック数のアイコンやページ最下部などにうまく画像が使われているのがポイントとなっている。

図8　オリジナルデザイン2

もう1種類スタイルシートを作ってみた。引き締まったイメージのページデザインだ。こちらはテンプレートもさらに少し変更してあるのでダウンロードして確認してほしい。一番特徴的なのはカレンダー部分だろう。

完全版のテンプレートをダウンロード! URL http://internet.impress.co.jp/im/200311mt/

©1994-2007 Impress R&D

オリジナルスタイルシート①～Sky：空～

図7のページはどういったCSSを書いたら実現できるのだろう。コンテンツの構成要素が、それぞれブロックレベル要素をともなってどのように配置されているのかを考えてみよう（図9）。

あたかもブロックレベル要素にボーダーを付けて表示させているかのようなこの状態を見ればもうピンと来たかと思うが、ようするにこういう配置になるようなスタイル定義をがんばって創ればいいということなのだ。

テンプレートデザインのポイント

レイアウト的に注目するのは、まずdiv id=mtEntries要素とdiv id=mtUtilities要素とで段組みにされているところだろう。段組みにはいくつかの手法があるが、今回はpositionプロパティーによる絶対配置を採用した。絶対配置にしないほうのブロックの縦サイズが絶対配置にしているブロックの縦サイズより長くなる必要があるという前提条件が付くが、ブラウザー間での実装上の挙動が比較的一致している手法だ。div id=mtEntries要素の左マージンを大げさに付けることで、左側に空白地帯を設ける。そしてその空白地帯の上に重ねるように、div id=mtUtilities要素を絶対配置にするわけだ。これで、見た目は段組みにされているようになる。

```
div#mtEntries {
    margin-left:200px;
    margin-right:18px;
    padding-top:1px;
    background-
color:#F0F0F0
}
div#mtUtilities {
    position:absolute;
    top:81px;
    right:auto;
    bottom:auto;
    left:20px;
    width:168px;
    padding-top:90px;
    background:transparent
url(“ptn1utilbg.gif”) top
left no-repeat;
}
```

段組みにされたそれぞれのカラムの中は、おおむね上から下へと積み重ねられるレイアウトになっているので、マージンやパティングなどをきっちり寸法を測って設定していけば、ほとんど問題なく配置できるだろう。

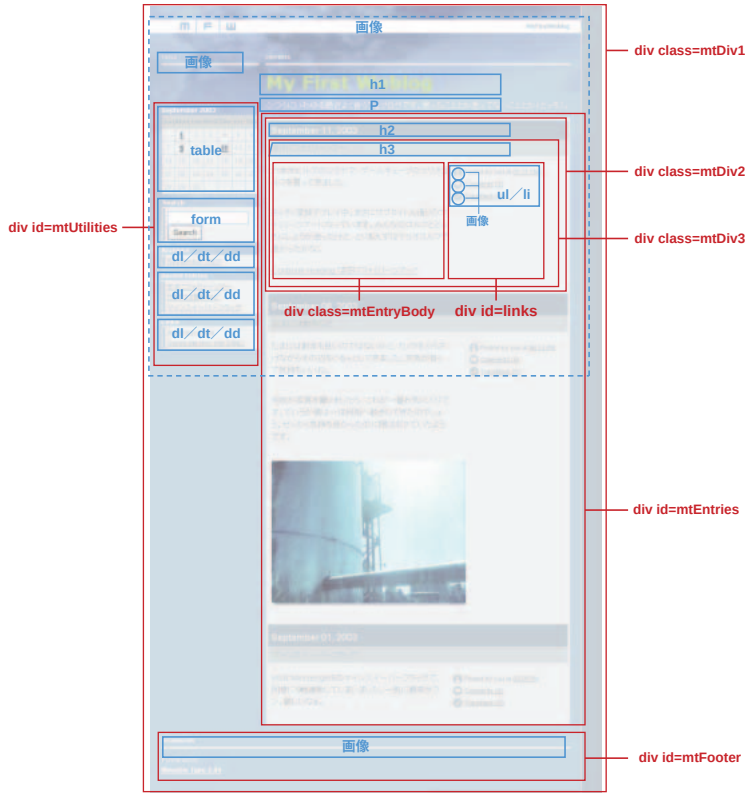
問題があるとすれば、やはりdiv class=mtEntryBodyの右にdiv class=mtEntryFooterを段組み的に配置しているところだろうか。こちらは今度はdiv class=mtEntryBodyにfloatプロパティーを用いて、div class=mtEntryFooter要素を回り込ませている。ちなみに回り込ませっ放しだと、後に続くエントリーが困ってしまうので、h2要素やh3要素でclearしておくのを忘れてはいけない。

```
div.mtDiv3 div.mtEntryBody
{
    padding-right:5px;
    margin-left:3px;
    padding-bottom:20px;
    width:330px;
    float:left;
}
```

構造とデザインの完全な分離

このデザインを実装する場合のポイン

図9　オリジナルデザイン1の出力と構造の関係



全体としてシンプルなドキュメント構造ながらも、含まれる内容は標準のものと変わらない。

インターネットマガジン／株式会社インプレスR&D
©1994-2007 Impress R&D

トは以上だ。基本的なCSSデザインがわかっていれば、ポイントになってくるところなんてだいたいこれくらいなのだ。特に新生Main Indexテンプレートが生成するXHTMLなら、さらに簡単だろう。XHTML側は完全に見栄えにこだわらずに構造のみを定義しており、CSS側は構造に見栄えをただひたすら付けていくだけという、かなり双方を分離した作業になる。さらには、MTにこだわらない既存の他のウェブページ(MTが生成したわけではないページ)で使用しているCSSを、ほぼそのままに流用することもできると思われる。

オリジナルスタイルシート②〜SPACE:宇宙〜

もう1つの違うデザインのもの(図8)には、前のデザインと比べて、とても大きな違いがある。そう、カレンダーが表ではなく縦列になっているのだ。もうこの時点で、さっきの新生Main Indexテンプレートそのままでは、こちらのデザインを実現することができない。このように、カレンダーをtable要素で意味づけしないようなパターンも十分にありえるので、こちらのバージョンの新生Main Indexテンプレートも別途用意しておいた。基本的な構造の階層などは変わらないのだが、いくつか多少異なる構造が含まれる。どちらの構造が妥当かという比較は困難で、いうなればどちらも正しいと考えられるものだ。こちらの新生Main Indexテンプレートの内容は、ダウンロードしてじっくり見てほしい。

1つ目のテンプレートとの違い

1つ目の新生Main Indexテンプレートと大きく違うのは、まず前述のとおりカレンダー自体の意味づけがul/li要素になっていることだが、このカレンダーを内包するブロックレベル要素が、div id=mtUtilities要素ではなくなっているという点も挙げられる。div id=mtEntries要素とdiv id=mtUtilities要素、この2つと同等の階層にdiv id=mtCalendar要素を用意してある。このdiv id=mtCalendar要素には、カレンダーのul/li要素だけでなく、月別アーカイブ一覧のdl要素も含むように、構造を変更している。このデザインでは、時系列的な意味合いをもつ構成要素を1つの段組みカラムに配置するようになっているが、これはカレンダーと月別アーカイブ一覧のまとまりを強く関連づける設計をしているためだ。それを意味づけに反映して、このような構造に作り変えたというわけだ。

また、div class=mtEntryFooter要素の中も、単なる箇条書きの内容とはせずに、「投稿したのは:だれ、投稿した時間は:何時」という表現で構成要素を組み替えたこのデザイン意図に基づいて、dl要素で意味づけされるように書き換えてある。

ブラウザーによって表示が変わるのはどうする?

ブラウザーのCSS実装に問題があって特定のプロパティーを解釈できないためにCSSでの表現が限られてくる、ということが往々にしてある。解釈できないプロパティーを無視してくれるのであればまだいいのだが、がんばって解釈しようとしてめちゃくちゃな表示結果を出してしまうということが多々ある。それは要するに、ブラウザーがバグっているのだが……。

最新版でない古めのブラウザーでCSSに対応してしまっているものは、だいたいCSS実装がかなりヤバイ感じだ。それらに対しては、可能ならCSSそのものを読み込ませないような細工をするのが楽だ。たとえばネットスケープ4.xであれば、link要素のmedia属性にscreen以外の値を設定しておけば、そこで指定しているCSSファイルを読み込まなくなる。

```
<link  rel="stylesheet"
       type="text/css"
       src="styles-site.css"
       media="screen,tv" />
<!--ネットスケープ4.xはこのファイルを
読めない-->
```

こういった、全体的に実装がボロボロだからCSSを無効にさせたいという場合の対策はまだ楽なのだが、特定のブラウザーに特定のプロパティーだけどうしても読み込ませたくないとかいうことが発生することがある。こういうときは、そのブラウザーが未実装なCSSの文法を織り交ぜて読ませなくするだとか、やはりバグを利用してその部分を無視させるだとかいったやり方で対応する。

個人的な経験則だが、CSSのコメントの終了印(*/)の前に半角円マーク(\)を入れるとコメントが終わっていないとマックOS版IE 5が勘違いするエスケープ技は、かなりよく使うので紹介しておこう。たとえばletter-spacingプロパティーに正の数値を設定していると、マックOS版のIE 5はその分をボックス幅の計算からなぜか除外するために、ボックスに設定したwidthを越えて中身の文章がはみ出してしまうことがある。だから筆者は、letter-spacingプロパティーを使うときは、そこだけを絶対にマックOS版のIE 5に読み込ませなくしている。

```
/* MacIE5はここから \*/
letter-spacing:5px;
/* ここまでをコメントだと勘違いする */
```

テンプレートデザインのポイント

さて、こちらのデザインについても、CSSをどう書いたらいいかのポイントをいくつかピックアップしていこう。まずは先ほどと同様に、ブロックレベル要素がどの配置に対応しているのかを表した図10を見てほしい。

一番困りそうなのはやはり段組みだろうか。こちらも芸がないようだがposition:absoluteを使って絶対配置にして解決してみた。div id=mtUtilities要素とdiv id=mtCalendar要素をそれぞれ絶対配置にして、div id=mtEntries要素は左マージンを大きく空けるという感じだ。

ただ、この3段カラムにはどれも半透明のグレーの背景が敷かれているのだが、

完全版のテンプレートをダウンロード! URL http://internet.impress.co.jp/im/200311mt/

©1994-2007 Impress R&D

この背景は一番縦に長いブロックの長さまで追従したものになっている。通常、絶対配置とかフロートさせた要素の背景を、させていない要素の縦の長さまで追従させることはできない。だから、見かけ上は追従しているかに見えるようにしてみたのだ。div id=mtEntries要素の背景が、もともと3段カラムの段組みの背景になっているわけだ。この場合、div id=mtEntries要素の左側を空けるためにマージンを設定してしまうと、背景の左上開始地点がそのマージンをとった後からになってしまうので、マージンではなくパディングで左側を空ける必要が出てくる。

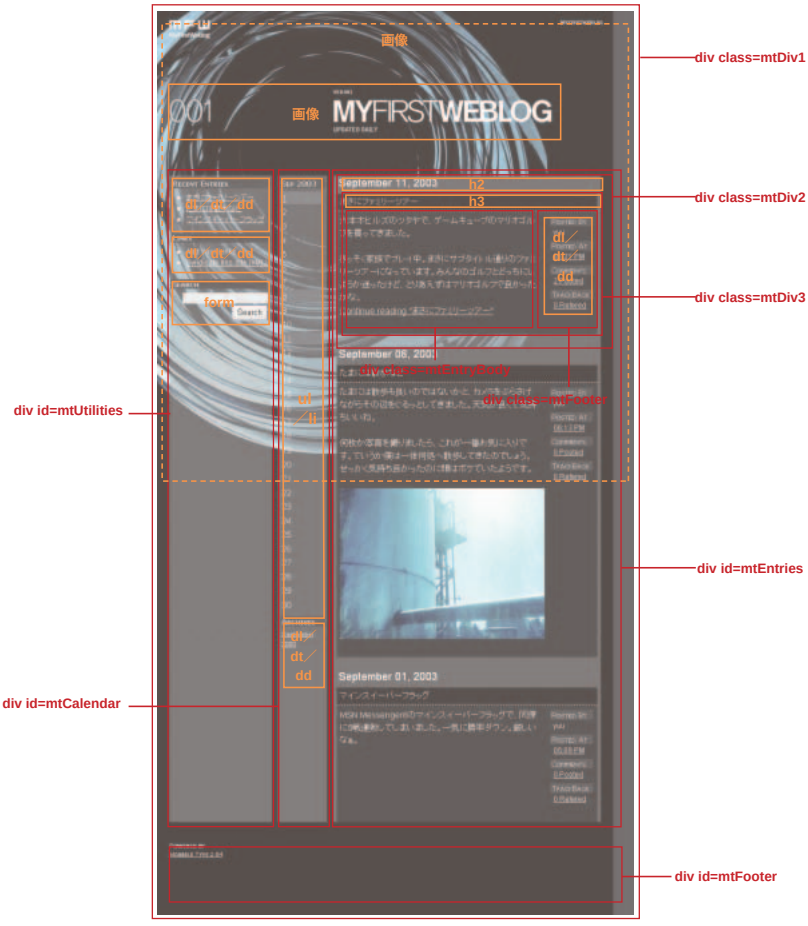
```
div#mtEntries {
    padding-left:294px;
    background:transparent
url(“ ptn2h2bg.gif ”) top
left repeat-y;
}
div#mtCalendar {
    position:absolute;
    top:270px;
    left:198px;
    width:80px;
}
div#mtUtilities {
    position:absolute;
    top:270px;
    left:20px;
    width:168px;
}
```

その他の要素配置や表示のスタイルに関しては、基本的なところは1つ前のCSSとおおむね変わりないので、ひたすら書いていけば完成するかと思われる。

正しいXHTMLがMTの世界を広げる

今回作り直した2つの新生Main Indexテンプレートは、どちらもXHTML 1.0 Transitionalとして妥当なXHTMLを生成する。さらに言えば、XHTML 1.0 Strictとしても妥当な内容になっているので、少し手を入れればXHTML 1.1としても妥当な内容に書き換えることができる。話が最初に戻ってしまうが、MTはコンテンツ管理システムなので、XHTMLの文法などを気にせずに、おもしろおかしいエントリーなどを登録していくことで、魅力的なウェブログサイトが作れることだろう。しかし、エントリーの内容に、採用しているXHTMLのバージョンに見合わない文法で書いたXHTMLを含めてしまうと、そのページは妥当なXHTML文書ではなくなってしまう。そういう意味では、やはりある程度、XHTMLの仕様にも通じてないとよろしくないと言える。

図10　オリジナルデザイン2の出力と構造の関係



こちらもドキュメント構造はシンプルだ。カレンダーに表を使っていない分だけコードもシンプルになっている。

よくあるblockquoteの文法間違い

よく見受けるのが、引用文をblockquote要素で意味づけしているときの誤りだ。引用のために使うブックマークレットが、誤った文法を助長させるようなblockquote要素の塊を吐き出したりしているのだろう。標準のCMS.pmファイルではそのようなことは発生しないが、引用文を挿入しやすくするために、CMS.pmファイルの該当箇所の出力内容をblockquote要素を伴うものに変更する方法が、インターネットで公開されている。しかし、それが誤った文法のblockquote要素を伴っているのだ。blockquote要素の直下に、テキストやインライン要素を書くことはできない。p要素やdiv要素などのブロックレベル要素を

インターネットマガジン／株式会社インプレスR&D
©1994-2007 Impress R&D

書いて初めてテキストやインライン要素が書けるようになるのだ。この誤った文法のblockquote要素をそもそも生成させないように、その変更済みのCMS.pmファイルをさらに変更しておくといいだろう。標準で「sprintf qq(%s\n\n%s)」となっていた部分を、たとえば次のように書き直す（改行せずに1行で書く）。

```
sprintf
qq(<blockquote><p><cite><a
title="%s" href="%s"
>%s</a></cite></p><div>%s</
div></blockquote>)
```

「%s」はそれぞれ前からリンクする対象のページのタイトル、リンクのURL、引用する内容に置き換えられるもので、この例では最後の「%s」をdiv要素でくくるようにしている。引用したのが単純な文章ならば、このdiv要素はp要素に書き直して意味づけを明確にしたほうがいいだろう。しかし必ずしも引用の内容が単純な文章だとは限らないので、あえてdiv要素にしているわけだ。最初からp要素が生成されてしまうと、p要素にはブロックレベル要素を含めることができないため、たとえば箇条書き（ul要素）などを引用できなくなってしまうからだ。

◇　◇　◇

これまでに解説してきたように、テンプレートが生成するXHTML自体を「正しい」ものにしていけば、あとは自由自在なCSSデザインができるようになる。ここではMain Indexテンプレートしか取り上げなかったので、実際にはまだまだやることが盛りだくさんなわけだが、すべてのテンプレートを一貫した構造で意味づけするのを忘れてはいけない。Stylesheetテンプレートの定義内容をテンプレートごとにどんどん増やしていくのではなく、MTが備える各ページ全体を通して同じスタイルを適用するようなやり方が可能になるわけだ。つまり、まずは、とにかくStylesheetテンプレート以外のテンプレートを正しく書き直すということが肝心だといえるだろう。

RSSやトラックバックなどのXML技術をうまく使ってこそウェブログだと言われる。せっかくXHTMLで作ってある出力ページだ。文書構造の定義と見栄えの定義とをしっかり切り分けることで、XHTMLページは純粋に情報そのものを取り扱えるようになる。これにより、ブラウザー以外のさまざまなアプリケーションからの利用も容易になり、世界がさらに広がるだろう。

MT、XHTML、CSSをさらに知るための情報

- 『Movable Typeで今すぐできるウェブログ入門』
  インプレス刊／ISBN4-8443-1812-8
  240ページ／本体1,800円＋税
- 『CSS 2 スタイルシート大辞典』
  インプレス刊／ISBN4-8443-1740-7
  496ページ／本体1,980円＋税
- 「XHTMLの書き方と留意点」（The Web KANZAKI）
  URL http://www.kanzaki.com/docs/html/xhtml1.html
- XHTMLの参考日本語訳（どら猫本舗）
  URL http://www.doraneko.org/xml/xhtml10/REC000126.html
- HTML 4.01の参考日本語訳（HTML 4仕様書邦訳計画補完委員会）
  URL http://www.asahi-net.or.jp/~sd5a-ucd/rec-html401j/cover.html
- CSS 2の参考日本語訳（Y-ADAGIO）
  URL http://www.y-adagio.com/public/standards/tr_css2/toc.html

スクリプトをXHTMLファイルに直接書くのはOK？

HTML 4までは script要素の内容は「CDATA」であるとされたが、XHTMLでは「#PCDATA」として扱われる。CDATAと#PCDATAはどちらもテキスト情報だが、ブラウザーやパーサーがそのテキストが書かれたとおりに解釈するのがCDATAで、HTML／XHTMLとして扱える部分はそう解釈されるのが#PCDATAだ。
たとえば「<」「>」「&」は、CDATAでは文字でしかないが、#PCDATAの場合はHTML／XHTMLで特別な意味のある文字として扱われる。#PCDATA領域内に「<!-- -->」があると、そこはコメント扱いになるわけだ。そこで本来ならばCDATAとして扱わせるためには、CDATAマーク区間を明示する。

```
<script type="text/javascript">
<![CDATA[
 （スクリプトの内容）
]]>
</script>
```

しかし、このCDATAマーク区間に対応しているブラウザーはまだ少ないため、スクリプトを外部ファイルにしてしまうのがいいだろう。これならば、要素内容をCDATAで扱おうが#PCDATAで扱おうが関係ないということになり、万事解決となる。

```
<script  type="text/javascript"
src="script.js"></script>
```

ただ、MT標準のMain Indexテンプレートでも、新生Main Indexテンプレートでも、スクリプトをscript要素内に直接書いている。これはたまたま書けたからそうしているのだ。「<」「>」「&」などのような、HTML／XHTMLで特別な文字とされるものが含まれている場合に、#PCDATAだと意図したものと違う動作になるということなのだから、それらを含まないスクリプトであればそのまま書いても問題ないとも言える（もちろんコメント扱いにならないように、「<!-- -->」は使っていない）。だが、たまたまうまくいく場合だけだというのを忘れてはいけない。基本的には、スクリプトは外部ファイルを用意してそこに入れるものだと考えておけば間違いない。

©1994-2007 Impress R&D

［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

Copyright © 1994-2007 Impress R&D, an Impress Group company. All rights reserved.