第10回　floatを使った段組みレイアウト（2段組）

前回解説した「floatプロパティ」を使ったテキスト回り込みを応用して、ボックスが横に2つ整列する2段組のレイアウトを作ってみましょう。このfloatプロパティを使った段組レイアウトの手法は、ページ全体のレイアウトだけでなくページ内に含まれる個々の情報ブロックのレイアウトなどでも利用できるものです。まずは、その基本形ともいえる2段組のレイアウトをマスターしましょう。
（解説：こもりまさあき）

要素の大きさを制限してボックスを右へ、左へ

情報が含まれるボックスを横に2つ整列させるためには、まず「widthプロパティ」を使ってそれぞれの要素の横幅を制限します。ブロックレベル要素の横幅を制限したとしても、ブラウザ内の表示上ではその要素は縦方向に整列したままの状態です。横幅を制限したそれぞれの要素に対してfloatプロパティを適用すれば、要素のボックスはフロート指定した方向にスライドし、結果として横方向に整列します。これが2段組のレイアウトの基本的な考え方です。

図1

【HTML ソース】

```
（中略）
<body>
 <h1> 大見出しのテキスト </h1>
 <p> 本文テキスト </p>
</body>
</html>
```

【CSS ソース】

```
h1 {
 background-color: #000;
 color: #fff;
 padding: 1%;
 width: 28%;
}
p {
 background-color: #ccc;
 padding: 1%;
 width: 68%;
}
```

ブロックレベル要素にwidthプロパティを指定するだけでは、それぞれの要素は縦方向に整列したままである。これは正しい挙動

図2

【CSS ソース】

```
h1 {
 background-color: #000;
 color: #fff;
 padding: 1%;
 width: 28%;
 float: left;
}
p {
 background-color: #ccc;
 padding: 1%;
 width: 68%;
 float: right;
}
```

▼クリックすると大きく表示されます。

それぞれの要素に対してfloatプロパティを使って方向を指定する。サンプルでは、h1要素は左、p要素は右にフロートさせてみた。これは双方の要素に「float: left;」を指定しても、HTMLの構造順で先にあるものから順番にフロートするため、表示結果は同じになる

サンプルでは、h1要素に「float: left;」、p要素には「float: right;」を指定しました。HTMLソースには「h1要素→p要素」の順番で記述されています。HTMLは上から記述順に読みこまれるため、先にh1要素が左方向にフロートした後、続くp要素が右方向にフロートしたということになります。h1要素とp要素に対して指定したfloatプロパティの値を左右逆にすれば、右側にh1要素、左側にp要素が配置されます。floatプロパティを使って段組レイアウトすれば、ボックス位置の変更も非常に簡単に処理できます。

図3

【CSS ソース】

```
h1 {
 background-color: #000;
 color: #fff;
 padding: 1%;
 width: 28%;
 float: right;
}
p {
 background-color: #ccc;
 padding: 1%;
 width: 68%;
 float: left;
}
```

▼クリックすると大きく表示されます。

h1要素とp要素の表示位置を左右逆にするには、h1要素に「float: right;」、p要素に「float: left;」を指定する。HTMLの構造順で先にあるものから順にフロートすることから、この場合はp要素に「float: right;」を指定しても同じレイアウトになる

ここでfloatプロパティを使った段組レイアウトの注意点をひとつ。CSSのボックスモデルのことを思い出しましょう。ボックスの横幅は、widthとpadding、border、marginの合計です。ブラウザウィンドウ内で使用できる領域を100%と考えた場合、それぞれのボックスの大きさを足した合計値がその使用可能な領域（100%）を超えてしまうとページレイアウトが崩れます。CSSを勉強し始めた頃は「widthの値だけをボックスの横幅」と考えがちなので注意が必要です（※IE 5.xなどこのボックスモデルの処理に問題のあるブラウザも存在します。ブラウザの違いによって引き起こされる問題については次回解説します）。

図4

【CSS ソース】

```
h1 {
 background-color: #000;
 color: #fff;
 padding: 1%;
 width: 30%;
 float: left;
}
p {
 background-color: #ccc;
 padding: 1%;
 width: 70%;
 float: right;
}
```

▼クリックすると大きく表示されます。

ひとつ前のCSSソースは、h1要素とp要素のwidthと左右のpaddingを合計すれば、「28%+1%+1%+68%+1%+1%=100%」となる。仮にこのソースのようにpaddingの値を無視した場合、合計「104%」となり100%を超えてブラウザウィンドウ内に収まらないため、結果としてp要素が下に落ちて表示されることになる

複数の要素をまとめてレイアウトする

前述した例は、h1要素とp要素の2つの要素を横方向に並べました。しかし、実際のWebサイトはこんなに単純なものではなく、実にさまざまな要素が含まれます。floatプロパティの使い方がわかったとはいえ、そのページ内に含まれる要素を個々にfloat指定するのは現実的ではありません。HTML文書では、そこに記載された情報は「見出し＋その本文」というように、グループ化することができるひとつの情報の固まりとして並んでいることが多いはずです。これらの情報の固まりは、div要素を使ってひとつのグループ化されたブロックレベル要素として扱うことが可能です。

図5

【HTML ソース】

```
(中略)
<h2> 記事タイトル 1</h2>
<p> 本文テキスト 1</p>
<h2> 記事タイトル 2</h2>
<p> 本文テキスト 2</p>

<h2> ナビゲーション見出し </h2>
<ul>
 <li> ナビゲーションテキスト </li>
 <li> ナビゲーションテキスト </li>
 <li> ナビゲーションテキスト </li>
</ul>
</body>
</html>
```

【HTML ソース】

```
(中略)
<div id="mainContents">
 <h2> 記事タイトル 1</h2>
 <p> 本文テキスト 1</p>
 <h2> 記事タイトル 2</h2>
 <p> 本文テキスト 2</p>
</div>
<div id="mainNavigation">
 <h2> ナビゲーション見出し </h2>
 <ul>
  <li> ナビゲーションテキスト </li>
  <li> ナビゲーションテキスト </li>
  <li> ナビゲーションテキスト </li>
 </ul>
</div>
</body>
</html>
```

現在のWebサイトは、ひとつのページ内に実に多様な情報が記載されていることが多い。これを個別にレイアウトするのはあまりに現実的ではない。複数のブロックレベル要素は、div要素を使って一つのブロックレベル要素としてグルーピングすることができる。情報の内容や種別を考慮して、まとめることが可能な複数のブロックレベル要素をひとつの大きな固まりとしてまとめてみよう。それぞれのグループ化したdiv要素のボックスは、id属性などを使うことで固まり全体をフロート処理できる

div要素を使った複数の要素のグルーピングができれば、前述した簡単なサンプルと同じようにそれぞれのdiv要素の横幅と適用したいフロート処理を指定するだけです。div要素にはid属性やclass属性を割り当てて、それぞれのブロックが区別できるようにしておきましょう。

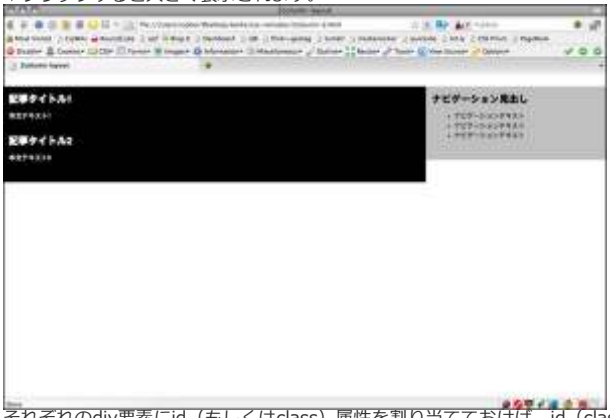
図6

【CSS ソース】

```
h1, h2, h3 {
 margin: 0 0 0.5em;
}
p, ul {
 margin: 0 0 2em;
}

div#mainContents {
 background-color: #000;
 color: #fff;
 padding: 1%;
 width: 68%;
 float: left;
}
div#mainNavigation {
 background-color: #ccc;
 padding: 1%;
 width: 28%;
 float: right;
}
```

▼クリックすると大きく表示されます。



それぞれのdiv要素にid（もしくはclass）属性を割り当てておけば、id（class）セレクタを使ってwidthやfloatを指定できる。div要素を使ったグルーピングさえできれば、手順や考え方は個々の要素をレイアウトした前述のサンプルと同じである

今回は、基本となる2段組のレイアウト手法に焦点を絞って解説しました。HTMLに記述された順番でボックスの大きさを指定しながら右に左にと、まるで積み木を組み立てる感じでレイアウトできる内容であれば、このフロート処理を基本として内容がレイアウトされていることが多いでしょう。情報量やレイアウトによって、必要であれば「clearプロパティ」でフロートを解除しながら、あとに続く要素をレイアウトして、ひとつのページを作り上げていくのです。

Point
・HTMLの順番を考慮して、ボックスをフロートさせて配置する
・ボックスモデルの存在を忘れない
・複数の要素をまとめる場合は、div要素でグルーピングする

次回は、もう少し高度なfloatを使ったレイアウトを解説し、一般的なWebサイトのレイアウトに挑戦する予定です。
