第11回　floatを使った段組レイアウト（3段組）

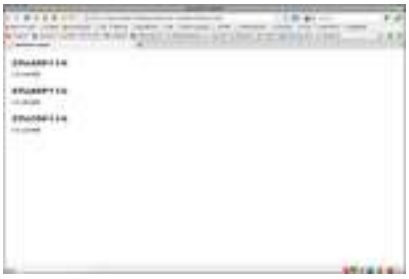
今回は2段組の「カラムレイアウト」を応用し、多くのWebサイトで採用されている3段組のカラムレイアウトを作成しましょう。前回解説したようにHTML文書の情報構造を考えて、それぞれのブロックの幅を決めて上から順に「floatプロパティ」で左右に振り分ければ3段組のカラムレイアウトの完成です。ここでは、floatプロパティでカラムレイアウトを作る際の注意点などもあわせて紹介します。
（解説：こもりまさあき）

2段組も3段組も考え方は同じ

前回解説した2段組は、HTML文書中に大きな情報ブロックが2つある状態でfloatプロパティを使って、それらのブロックを左右に並べました。3段組のカラムレイアウトも基本的な手法は同じものです。HTML文書の情報構造の順番を考え、「左」、「中央」、「右」に配置するそれぞれのカラム幅を指定し、floatプロパティを適宜指定して横に整列させます。

図1（▼右の画像はクリックすると大きく表示されます。）
【HTML ソース】

```
（中略）
<body>
<div id="columnA">
 <h2> カラム A のタイトル </h2>
 <p> カラム A の内容 </p>
</div>
<div id="columnB">
 <h2> カラム B のタイトル </h2>
 <p> カラム B の内容 </p>
</div>
<div id="columnC">
 <h2> カラム C のタイトル </h2>
 <p> カラム C の内容 </p>
</div>
</body>
</html>
```



サンプルソースは非常に簡単な内容にした。実際は、ページに含まれるコンテンツの内容や構造順を考えてHTMLソースを記述する必要がある。複数の要素をまとめてレイアウトする場合は、div要素を使って1つのブロックとしてグルーピングすれば作業はしやすくなる

図2（▼右の画像はクリックすると大きく表示されます。）

【CSS ソース】

```
div#columnA {
 float: left;
 width: 21%;
 padding: 2%;
 background-color: #c0c0c0;
}
div#columnB {
 float: left;
 width: 46%;
 padding: 2%;
 background-color: #cacaca;
}
div#columnC {
 float: left;
 width: 21%;
 padding: 2%;
 background-color: #f0f0f0;
}
```

サンプルではブラウザウィンドウ全体（100%）を使用すると想定して、各カラムにはそれぞれ「%」を使った横幅や余白の指定をしている。すべてのdiv要素に対して「float: left;」を指定しているため、文書構造順に「A」、「B」、「C」の順番で横並びになっている

図3（▼右の画像はクリックすると大きく表示されます。）

【CSS ソース】

```
div#columnA {
 float: right;
 width: 21%;
 padding: 2%;
 background-color: #c0c0c0;
}
div#columnB {
 float: left;
 width: 46%;
 padding: 2%;
 background-color: #cacaca;
}
div#columnC {
 float: right;
 width: 21%;
 padding: 2%;
 background-color: #f0f0f0;
}
```

HTMLの文書構造はそのままに、CSSのfloatプロパティの指定だけを変更すれば2カラム同様レイアウトの変更も可能だ

このように、floatプロパティを使った3カラムのレイアウトは、2カラムのレイアウトの応用に過ぎません。元のHTMLの文書構造を活かしながらfloatプロパティとwidthプロパティを駆使し、特定の情報ブロックの位置を左や右にスライドさせて配置していくのです。

一般的なWebサイトのレイアウトを作ってみよう

ここまでは極めてシンプルなHTMLを使って、floatプロパティを使ったカラムレイアウトの手法を紹介してきました。しかし、皆さんもご存知のように現在のWebサイトはそのようにシンプルなレイアウトばかりではありません。ここからは、これまで説明した2カラムと3カラムの実践として、普段良く目にするWebサイトのレイアウトに挑戦してみましょう。

図4

【HTML ソース】

```
（中略）
<body>
<div id="wrapper">

<div id="header">
 <h1>Site Title Inc.</h1>
 <p>float property layout example.</p>
</div>

<div id="columnA">
 <h2> カラム A のタイトル </h2>
 <p> カラム A の内容 </p>
</div>
<div id="columnB">
 <h2> カラム B のタイトル </h2>
 <p> カラム B の内容 </p>
</div>
<div id="columnC">
 <h2> カラム C のタイトル </h2>
 <p> カラム C の内容 </p>
</div>
<div id="footer">
 <address>Site Title Inc. All rights reserved.</address>
</div>

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

（▼下の画像はクリックすると大きく表示されます。）

一般的なWebサイトの構成要素にあたる内容をHTMLでマークアップした。id「header」のdiv要素はヘッダー、id「columnA」、「columnB」、「columnC」のそれぞれのdiv要素は主コンテンツ、id「footer」のdiv要素はフッターにあたるブロックになる。さらにこれらすべてを1つのブロックに内包して中央揃えを実現するため、id「wrapper」のdiv要素でコンテンツ全体を囲んでいる点に注意してほしい

ここでは、このHTML文書の構造順でCSSを記述していきます。まずは、コンテンツ全体を囲んでいるid「wrapper」にwidthプロパティを指定して中央揃えにします。コンテンツ全体の横幅を指定して左右のマージンを自動調整することで、多くのブラウザで中央揃えを実現可能です。id「header」の横幅はコンテンツ幅でよいので、ここでは「width: auto;」として自動調整しています（これはなくても構いません）。

図5（▼右の画像はクリックすると大きく表示されます。）
【CSS ソース】

```
（中略）
div#wrapper {
 width: 900px;
 height: auto;
 margin: 0 auto;
 border: 1px solid #222222;
}
div#header {
 width: auto;
 padding: 20px 10px;
 background-color: #990000;
 color: #fff;
}
```

コンテンツ全体の横幅を固定するためには、コンテンツ全体を1つのグループとして取り扱った方が簡単だ。全体を囲むdiv要素に対してwidthプロパティを指定し、左右のマージンに「自動」を指定することで中央揃えが実現できる

続いて主コンテンツであるカラムの「A」、「B」、「C」の3つのブロックを横に整列させてみましょう。ここでは、一番左にid「columnB」のdiv要素を配置し、その右側にid「columnC」とid「columnA」を配置します。コンテンツ全体の横幅は「900px」ですので、その幅を超えないようそれぞれのブロック幅を指定します。

図6
【CSS ソース】

```
（中略）
div#columnA {
 float: right;
 width: 180px;
 padding: 10px;
 background-color: #c0c0c0;
}
div#columnB {
 float: left;
 width: 479px;
 padding: 10px;
 background-color: #ffffff;
 border-right: 1px solid #222222;
}
div#columnC {
 float: right;
 width: 180px;
 padding: 10px;
 background-color: #f0f0f0;
}
```

それぞれのカラムの横幅を指定し、floatプロパティで配置位置をコントロールする。ここに記述された値以外にもいろいろと試してみるとよい。レイアウトが崩れてしまった場合は、widthやpadding、borderなどの値をもう一度確認してみよう

最後にフッターのブロックのCSSを記述しましょう。直前にある3カラムのフロート指定をここで解除しま

す。セレクタ「div#footer」に対して「clear: both;」を指定します。

図7（▼右の画像はクリックすると大きく表示されます。）
【CSS ソース】

```
（中略）
div#footer {
 clear: both;
 width: auto;
 background-color: #222222;
 text-align: center;
}
div#footer address {
 font-style: normal;
 font-size: 0.8em;
 line-height: 2;
 color: #999999;
}
```

最後にフッターエリアのCSSを記述する。ヘッダー同様ここは1カラムでレイアウトすると仮定し、コンテンツ幅いっぱいを利用するため「width: auto;」を指定した。直前の3カラムのフロートを解除するため、id「footer」のdiv要素に対して「clear: both;」を指定

ここまでで一般的なWebサイトの基本レイアウトが完成しましたが、現実のWebサイトではさらにそれぞれのカラムの中で別の情報ブロックが2段組になっているなど複雑なレイアウトが多いものです。さらなる応用として、カラムBのコンテンツを追加して内容を2段組で整列してみましょう。

図8
【HTML ソース】

```
（中略）
<div id="columnB">
 <h2> カラム B のタイトル </h2>
 <p> カラム B の内容 </p>
 <div id="subContents1" class="subContents">
  <h3> カラム B のサブコンテンツ 1</h3>
  <p> サブコンテンツ 1 の内容 </p>
 </div>
 <div id="subContents2" class="subContents">
  <h3> カラム B のサブコンテンツ 2</h3>
  <p> サブコンテンツ 2 の内容 </p>
 </div>
</div>
```

カラムBの内容を変更する。カラムBのdiv要素内にid「subContents1」とid「subContents2」のdiv要素を追加して、それぞれに内容を記述

図9（▼右の画像はクリックすると大きく表示されます。）
【CSS ソース】

```
（中略）
div#columnB div.subContents {
 float: left;
 width: 216px;
 padding: 8px;
 border: 1px solid #222222;
 margin-right: 10px;
 background-color: #fcfcfc;
}
div#columnB div#subContents2 {
 margin-right: 0;
}
```

カラムBに追加した2つのdiv要素をカラム内で横に並べるため、それぞれのブロックをフロートさせる必要がある。この場合のカラムBの横幅は「479px」であるため、2つのブロックの合計がその値を超えないように注意しなければならない。ここでは、HTMLに指定したclass「subContents」を利用して、双方のブロックにいったん同じ値とフロート処理を適用する。しかし、このままでは右側のカラムにも右マージンが適用されてしまいレイアウトが崩れてしまうため、idセレクタを使って右側のカラムだけ右マージンを「0」に上書きしている

このようにレイアウトによっては、複数のカラムレイアウトを組み合わせる必要も出てきます。それぞれのブロックに個別にCSSを適用することもできますが、共通化できるスタイル指定を見つけ出せれば上書きなどCSSの特性を利用することで少ない手順で実装することも可能です。慣れないうちは個別の指定でも構わないので、いろいろとチャレンジしてみましょう。

ここでは左側のコンテンツが縦方向に伸びたため、右側のコンテンツの下に空きができています。しかし、これは正しい挙動です。一番簡単な対処方法は、カラムAとCの双方にheightプロパティを使って高さを明示することでしょう。しかし、その対処方法ではコンテンツがそれぞれで伸びた場合にまた問題になります。別の対処方法については、また別の機会に解説します。

フロート時に注意したいIEのバグ

前回と今回の2回にわたって、floatプロパティを使ったカラムレイアウトを解説してきました。すべてのブラウザが同じようにCSSを解釈してくれると問題はないのですが、このfloatプロパティを使ったレイアウトでは気をつけておきたいことがあります。いわゆるFirefoxやSafariのようなモダンブラウザでは問題は起きませんが、WindowsのIEのバージョン6以下ではfloatプロパティを使う際に特定のCSSの指定でバグが発生します。先ほどの応用で記述したCSSの内容を以下に変更します。

図10
【CSS ソース】

```
（中略）
div#columnB div.subContents {
 float: left;
 width: 216px;
 padding: 8px;
 border: 1px solid #222222;
 margin-left: 10px;   ※←「margin-right」を「margin-left」に変更
 background-color: #fcfcfc;
}
div#columnB div#subContents1 {   ※←「subContents2」を「subContents1」に変更
 margin-left: 0;   ※←「margin-right」を「margin-left」に変更
}
```

セレクタ「div#columnB div.subContents」の「margin-right」を「margin-left」に変更する。さらに、セレクタ「div#columnB div#subContents2」を「div#columnB div#subContents1」に変更し、「margin-right」を「margin-left」に変更してみよう。先ほどまでは右側に付与されていたマージンを今度は左側に付与するようにした格好だ

この変更によって、カラムBの中で2つ横並びになっているブロックは「左側にフロートして、左側にマージンが付いている」状態になります（subContents1は左マージン0）。このソースをIE 6以下のブラウザで閲覧すると、右側のカラムの左側の余白が「20px」と誤って解釈されてレイアウトが崩れてしまうのです。IE 6以下のブラウザでは、「フロートさせた方向と同じ方向にマージンを指定すると値が2倍になる」という重大なバグがあるのです。マージンではなく余白で調整できれば問題はありませんが、もし余白で調整できないような場合は以下のように「display: inline;」という1行の記述を追加すれば回避できます。ぜひ覚えておきましょう。

図11
【CSS ソース】

```
div#columnB div.subContents {
（中略）
 display: inline;   ※←「display: inline」を追加
}
```

どうしてもフロートと同じ方向にマージンが必要な場合は、「display: inline;」という1行を追加することでこのバグは回避できる。これはブロックレベル要素に対して起こるバグなので、ブロックレベル要素をインライン要素に変更することでバグを回避するという仕組みだ

Point

・3カラムのレイアウトも基本は2カラムレイアウトの応用
・共通化できるスタイル指定は、まとめて指定して上書きで修正
・フロート処理をする際は、マージンの指定方向に注意
