SUPPLEMENTARY LESSON

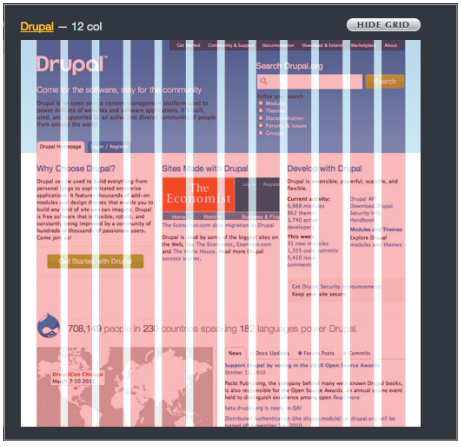
補講

グリッドシステム

本書 Chapter09 で作ったサンプルサイトのデザインは、「960 グリッドシステム」というものを活用してデザインされています。グリッドシステムとは、レイアウトを格子状の分割線（グリッド）に分解し、その線に整列してボックスを配置することで、情報の見やすい安定したレイアウトを構築するデザイン手法です。情報の見やすさを重視する Web デザインとグリッドシステムは相性が良いため、960px をコンテナサイズとして 12、16、24 分割する「960 グリッド」や、978px をコンテナサイズとする「978 グリッド」などの各種グリッドシステムが提唱されています。

●図 31-6　960 grid system（http://960.gs/）

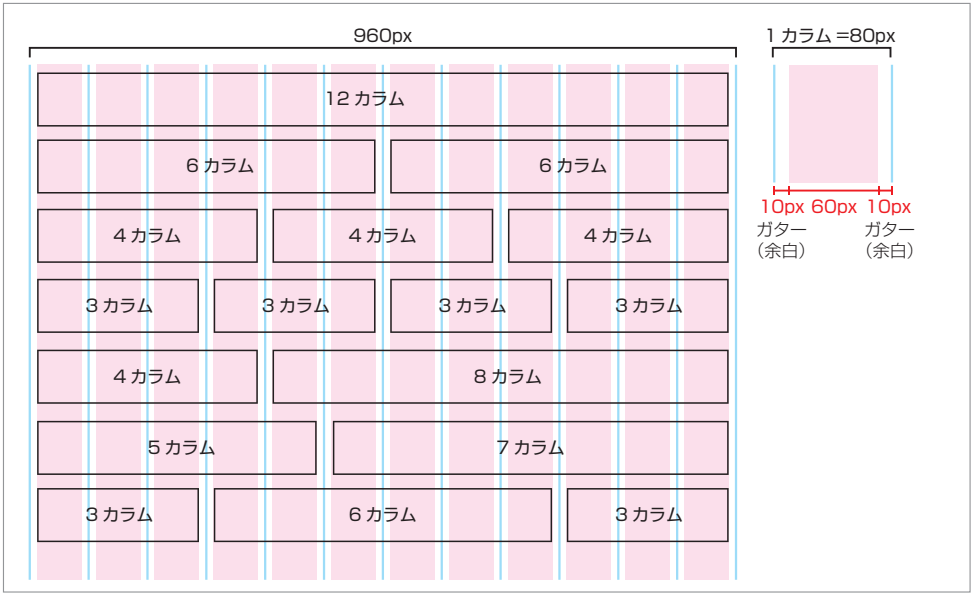
●12 分割例



CSS グリッドシステム

今回のデザインでは 960px の 12 分割を使っています。1 カラムは 80px、1 カラムには左右に 10px ずつのガター（余白）を含んでいます。グリッドシステムでは、このガターを含むカラムをいくつ使用するか、という観点でレイアウトを考えます。例えば実習サイトのメニュー領域のような均等 3 分割の場合、4 カラム× 3、「お知らせ」と「スタッフの一言」のような 2 分割の場合は 6 カラム× 2 と考えます。つまり横に並ぶボックスが使用するカラム数が合計 12 になるようにシステム化されているということです。

●図 31-7　グリッドシステムによるマルチカラムレイアウト



これを CSS で再現し、ボックスが使用するカラム数を class で指定するだけで簡単にマルチカラムレイアウトを実現できるようにしたものが CSS グリッドシステムです。Chapter09 の実習ではレスポンシブコーディングの仕組みを理解するために敢えて固有のデザインをそのまま数値化してレイアウトしましたが、元のデザインがグリッドシステムでデザインされているので、当然今回のサイトも CSS グリッドシステムでレイアウトすることができます。

グリッドシステムを使ったレイアウト

グリッドシステムを使う最も大きなメリットは、レイアウト情報を汎用化できる点です。例えばメニューとバナーはコンポーネントとしては別のものですが、「均等 3 カラム」というレイアウトの仕組みは同じです。固有のコンポーネントスタイルの中にレイアウト情報を組み込んだ場合、同じような 3 カラムレイアウトが別の場所で出てきたとき、繰り返し同じ記述をする必要が出てきます。

```
/*Pickup Menu
--------------------*/
.menu-list li{
    float: left;
    width: 31.9148%;
    margin-right: 2.1276%;
}
.menu-list li:nth-child(3n){
    margin-right: 0;
}

/*banner
--------------------*/
.banner-list li{
    float: left;
    width: 31.9148%;
    margin-right: 2.1276%;
}
.banner-list li:nth-child(3n){
    margin-right: 0;
}
```

※3カラムに設定する部分のコードの中身は全く同じ。

しかしレイアウト情報だけを専用の class で管理するグリッドシステムなら、あらかじめ設定された class を必要なボックスに指定するだけで良いため、CSS レイアウトに詳しくない人でも非常に楽にマルチカラムのレイアウトを実装できるようになります。

以下のコードは 960px グリッド 12 分割の CSS グリッドシステムの例になります。具体的な CSS コードの仕組みには「左右のガターを margin とするか padding とするか」「ガターも % にするか」「両端のガターをどう処理するか」「レスポンシブの場合 1 カラム化するブレイクポイントをどこに設定するか」などの点で様々な作り方があります。

```
/*コンテナ*/
.container{
    max-width: 960px;/*ガター込みで計算するのでコンテナ幅は960*/
    margin: 0 auto;
}
.container:after{
    content:"";
    display: block;
    clear:both;
}

/*グリッド設定*/
.grid1,.grid2,.grid3,.grid4,.grid5,.grid6,
.grid7,.grid8,.grid9,.grid10,.grid11,.grid12{
    width: 100%;
    padding-left: 10px;/*ガターはpx固定*/
    padding-right: 10px;
    box-sizing: border-box;/*グリッド幅にガターを含む*/
    float:left;
}

/*マルチカラム設定*/
@media screen and (min-width: 640px) {
.grid1{width: 8.3333%;} /* 80/960 */
.grid2{width: 16.6666%;}/* 160/960 */
.grid3{width: 25%;}    /* 240/960 */
.grid4{width: 33.3333%} /* 320/960 */
.grid5{width: 41.6666%;}/* 400/960 */
.grid6{width: 50%;}    /* 480/960 */
.grid7{width: 58.3333%;}/* 560/960 */
.grid8{width: 66.6666%;}/* 640/960 */
.grid9{width: 75%;}    /* 720/960 */
.grid10{width: 83.3333%;}/* 800/960 */
.grid11{width: 91.6666%;}/* 880/960 */
.grid12{width: 100%;}   /* 960/960 */
}
```

このコードの場合、

- ガターは padding（左右 10px 固定）で設定
- border-box にしてガター込み 1 カラム 80px 換算で計算
- ガター込みでカラム配置するので両端のガター処理は特に行わない
- 1 カラム化するブレイクポイントは 640px
- グリッドのネストは想定しない

といった設定で作られています。

これを実習のデザインに当てはめると、次のようなレイアウト構造となります（※実際のマークアップと CSS のコーディング例は、Chapter09 ＞参考＞ grid-system にありますので、確認してください）。

●図 31-9　グリッドシステムでのレイアウト

グリッドシステムを使う場合の注意点

グリッドシステムでは、原則として「各コンポーネント固有のデザインスタイル」と「レイアウト情報」を切り離して考えます。グリッドシステムのレイアウト枠それ自体に固有のデザイン（背景・余白・border など）をつけてしまうと、カラム幅計算に影響を及ぼしてしまう恐れが高くなります。そこでグリッド枠にはあくまでレイアウト用としてカラム幅のみ設定し、固有のスタイルは子要素を作ってそちらに指定するのが基本です。

このような考え方から、必然的にグリッドシステムを採用した Web ページの HTML は文書構造に関係のないレイアウト用の div 要素やレイアウト情報を指定するための class が増え、各コンポーネントのセレクタに直接レイアウト情報を記述した場合と比べてどうしてもソースコードの見通しは悪くなりがちです。

また、グリッドシステムではカラム間の余白サイズは全て一律となります。したがって「特定のコンポーネントに限ってカラム間余白のサイズを変えたい」等、システムの想定外となるデザインを行おうとした場合にはカスタマイズが必要となり、少々手間がかかる可能性があります。つまり、グリッドシステムを使ってレイアウトの手間を減らしたいのであれば、デザインする時からグリッドシステムの利用を想定したデザインにしておくことが重要だということです。特にデザインとコーディングが分業となっているような場合には、そもそもデザイン自体がグリッドシステムにのっとったものになっているかを判断してからでないと、うまくメリットを活かせない恐れがありますのでその点には注意しましょう。

グリッドシステムを利用したコーディングに修正したデータは Chapter09 ＞参考＞ grid-system にありますので、興味のある方はコードとマークアップや CSS の設定の違いを比較してみてください。

COLUMN

グリッドシステムと CSS フレームワーク

「Bootstrap」や「Foundation」といったCSSフレームワークには、最初からグリッドシステムに必要なコードが含まれています。こういったCSSフレームワークにはグリッドシステムだけでなく、よく使われるボタンや見出しなどの各種コンポーネント、スライダー・モーダルといった動的なUI構築に必要なファイルやコード一式が用意されているものもあり、マニュアルに従って必要なコードを記述すれば、HTML・CSSの専門家でなくても簡単に今どきのレスポンシブサイトを構築できるようになっています。各フレームワーク独自のルールやクセを覚える必要はありますが、うまく使えばサイト構築の手間を大幅に減らしてくれる可能性があります。

全て一から自力で構築するのも良いですが、近年はCSSフレームワークでベースを構築し、独自のデザインやカスタマイズが必要なところだけ独自に開発するといった手法を採用するところも増えています。本書でHTML・CSSの基礎を理解したら、こういった便利なツールを利用して制作工程を効率化することも検討してみるとよいでしょう。

- 「Bootstrap」 URL http://getbootstrap.com/
- 「Foundation」 URL http://foundation.zurb.com/