

# 第2章

## Bootstrapの レイアウト

Bootstrapには、12列のグリッドシステム、レスポンシブのためのユーティリティクラスなど、プロジェクトをレイアウトするためのコンポーネントや機能が組み込まれています。Bootstrapで自由なレイアウトをするために、これらのしくみや使い方の基本を理解しておきましょう。

# Bootstrapのグリッドシステム

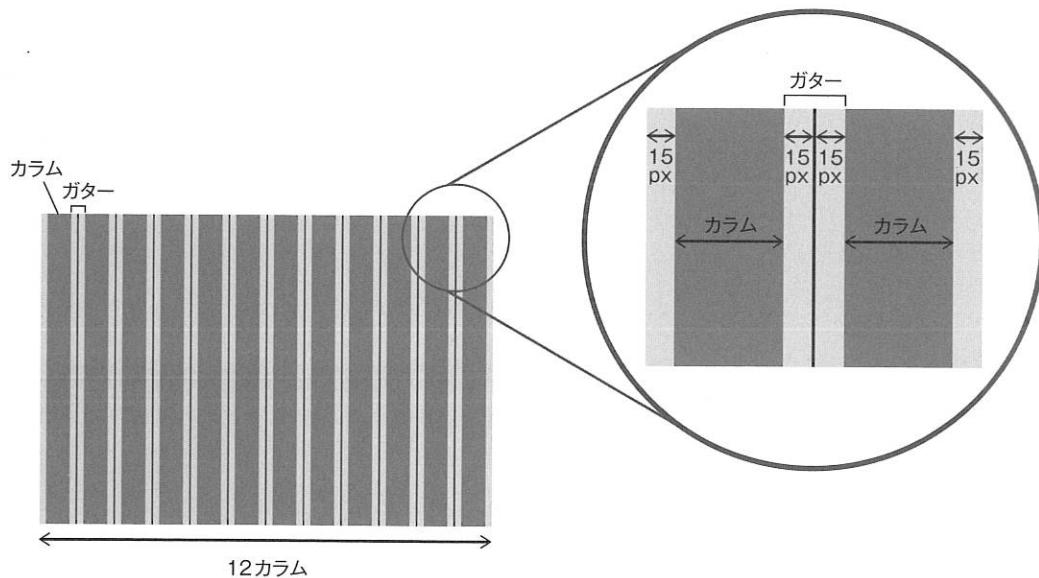
Bootstrap のレイアウト方法には、ページの幅を 12 列に分割した**グリッドシステム**が採用されています。この機能は、ページを 2 段組みや 3 段組みにしたり、コンポーネントを横に並べて配置したりする場合などに活用できます。それでは、Bootstrap のグリッドシステムを構成する要素を見ていきましょう。

## 2.1.1 カラム (column:列) とガター (gutter:溝)

12 列に分割されたグリッドの列を**カラム** (column:列) といい、各カラムの間の余白を**ガター** (gutter:溝) といいます。

各カラムの左右内側 (padding) には 15px のガターが設定されているため、コンテンツ間の余白は 30px となります (図 2-1)。

▼図 2-1 カラムとガター

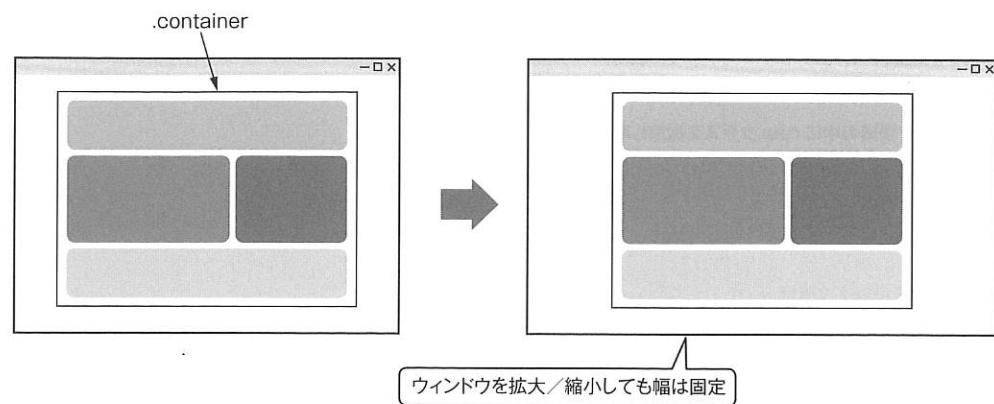


## 2.1.2 コンテナ (container: 箱)

Bootstrap のグリッドシステムを使用してレイアウトする際、最初に使用するのが**コンテナ** (containers) です。コンテナは、コンテンツを入れる箱で、ページの水平中央にコンテンツを配置する役割を持っています。

コンテナには、**固定幅コンテナ**と**可変幅コンテナ**の2種類があり、画面サイズによってコンテンツの最大幅 (max-width) を切り替えるには、固定幅コンテナの **container クラス**を使います（リスト 2-1、図 2-2）。

▼図 2-2 固定幅コンテナ



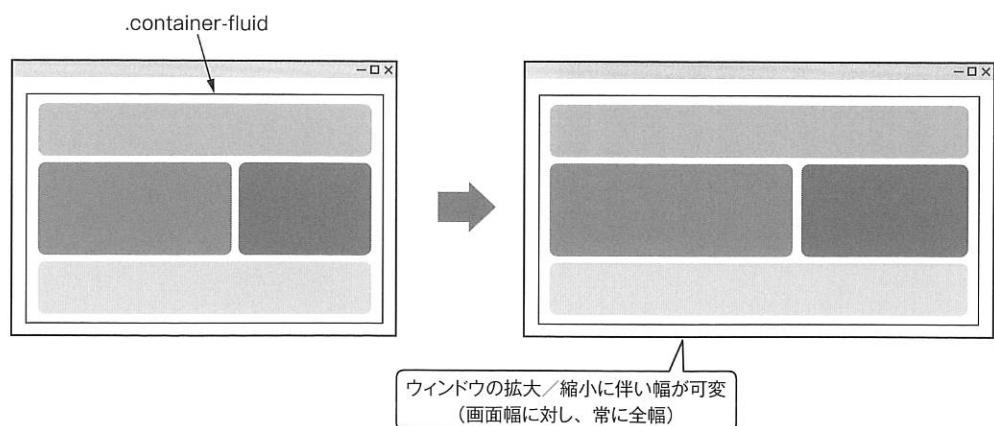
▼リスト 2-1 固定幅コンテナ

```
<div class="container">
  <!-- ここにコンテンツを入れる --&gt;
&lt;/div&gt;</pre>

```

画面サイズ全幅に渡る流動的なコンテンツ幅を持たせるには、可変幅コンテナの **container-fluid クラス**を使用します（リスト 2-2、図 2-3）。

▼図 2-3 全幅コンテナ



## ▼リスト 2-2 全幅コンテナ

```
<div class="container-fluid">
  <!-- ここにコンテンツを入れる -->
</div>
```

**2.1.3 row クラス（行）**

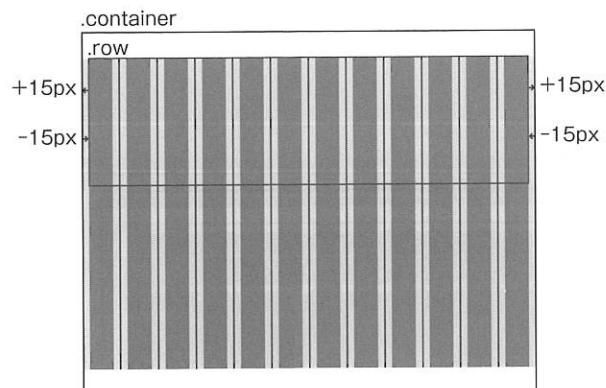
コンテナ（箱）の中には、**row クラス**（行）を入れます。row クラスは、一連のカラムを正しくレイアウトするために使用するクラスで、この中にカラムをまとめます。複数行にしたいときは、container の中に row クラスを指定した要素を追加するだけです（リスト 2-3）。

## ▼リスト 2-3 コンテナ要素の中に row クラスを指定した要素を入れる

```
<div class="container">
  <div class="row"><!-- 1行目 -->
    <!-- ここにカラムを入れる -->
  </div>
  <div class="row"><!-- 2行目 -->
    <!-- ここにカラムを入れる -->
  </div>
</div>
```

row クラスの左右にはマイナスマージン (-15px) が設定されており、行内にまとめられた一連のカラムの左右端のガターサイズ (15px) を相殺します。これにより、一連のコンテンツが正しく整列されるようになります（図 2-4）。

▼図 2-4 .row 要素による左右ガターの相殺イメージ



## 2.1.4 グリッドシステムの使い方

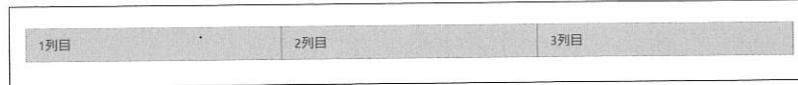
Bootstrap のグリッドシステムは、一連のコンテナ（container クラスまたは container-fluid クラス）、行（row クラス）、およびカラム（col クラス）を使用してコンテンツをレイアウトします。

次の例では、3 つの等幅カラムを作成しています（リスト 2-4、図 2-5）。col クラスを指定した 3 つのカラムは、row クラスを指定した要素で 1 行に取りまとめられます。さらに、container クラスを指定した要素に格納されることでページ中央に配置されます。

▼リスト 2-4 Bootstrap のグリッドシステムの基本構造（gridsystem-basic.html）

```
<div class="container"><!-- container : 箱 -->
  <div class="row"><!-- row : 行 -->
    <div class="col">1列目</div><!-- col : カラム -->
    <div class="col">2列目</div><!-- col : カラム -->
    <div class="col">3列目</div><!-- col : カラム -->
  </div>
</div>
```

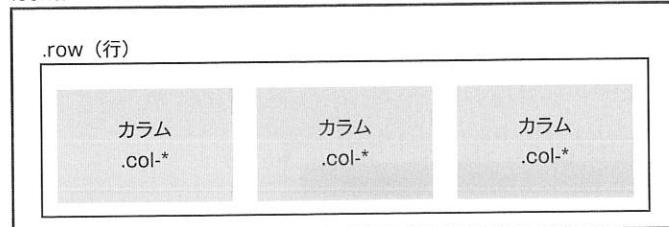
▼図 2-5 グリッドシステムの基本構造



基本構造は、**コンテナ（container クラスまたは container-fluid クラス）** > **行（row クラス）** > **カラム（col クラス）** となることを覚えておきましょう（図 2-6）。

▼図 2-6 Bootstrap のグリッドシステムの基本構造

.container または .container-fluid （コンテナ）



## 2

SECTION

## 2

## 列の自動レイアウト

それでは、Bootstrap でグリッドレイアウトをするための具体的なコードを見ていきましょう。本節では、デバイスによってカラム数を切り替える必要のない、全デバイス共通のレイアウト実装方法を説明します。

## 2.2.1 等幅カラム

行内のカラムを等幅にするには、**col クラス**を使用するのがもっとも簡単です。1 行を 2 列にしたい場合は、col クラスを指定した要素を 2 つ、3 列にしたい場合は 3 つ配置します。このクラスを使用した場合、1 行に入る各カラムは自動的に等幅でレイアウトされます（リスト 2-5、図 2-7）。

▼リスト 2-5 等幅カラム (equal-width.html)

```
<div class="container"><!-- container : 箱 -->
  <div class="row"><!-- row : 1行目 -->
    <div class="col">1列目</div><!-- col : 等幅カラム -->
    <div class="col">2列目</div><!-- col : 等幅カラム -->
  </div>
  <div class="row"><!-- row : 2行目 -->
    <div class="col">1列目</div><!-- col : 等幅カラム -->
    <div class="col">2列目</div><!-- col : 等幅カラム -->
    <div class="col">3列目</div><!-- col : 等幅カラム -->
  </div>
</div>
```

▼図 2-7 等幅カラム



## 2.2.2 指定幅カラム

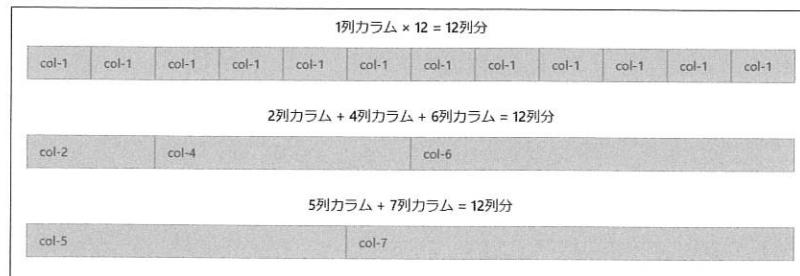
カラムの幅を指定するには、**col-\* クラス**（「\*」には 1 ~ 12 までの数値が入ります）を使用します。\* に入る数字は、合計が 12（グリッド 12 列分のカラム）になるように、使用したい列分の数を設定します。等幅カラムが必要な場合は、既述の col クラスを指定した要素を 3 つ並べる代わりに col-4 クラスを指定した要素を 3 つ並べても、4 列カラム × 3 = 12 カラム列分となり、3 つの等幅カラムになります。

カラムの幅は「%」で定義されます。これによって親要素との相対値でカラムの幅サイズが指定されるため、流動的なサイズになります（リスト 2-6、図 2-8）。

▼リスト 2-6 指定幅カラムのレイアウト（setting-column.html）

```
<h3>1列カラム × 12 = 12列分</h3>
<div class="container"><!-- container : 箱 -->
  <div class="row"><!-- row : 行 -->
    <div class="col-1">col-1</div><!-- col-1 : 1列分のカラム -->
  </div>
</div>
<h3>2列カラム + 4列カラム + 6列カラム = 12列分</h3>
<div class="container"><!-- container : 箱 -->
  <div class="row"><!-- row : 行 -->
    <div class="col-2">col-2</div><!-- col-2 : 2列分のカラム -->
    <div class="col-4">col-4</div><!-- col-4 : 4列分のカラム -->
    <div class="col-6">col-6</div><!-- col-6 : 6列分のカラム -->
  </div>
</div>
<h3>5列カラム + 7列カラム = 12列分</h3>
<div class="container"><!-- container : 箱 -->
  <div class="row"><!-- row : 行 -->
    <div class="col-5">col-5</div><!-- col-5 : 5列分のカラム -->
    <div class="col-7">col-7</div><!-- col-7 : 7列分のカラム -->
  </div>
</div>
```

▼図 2-8 指定幅カラム



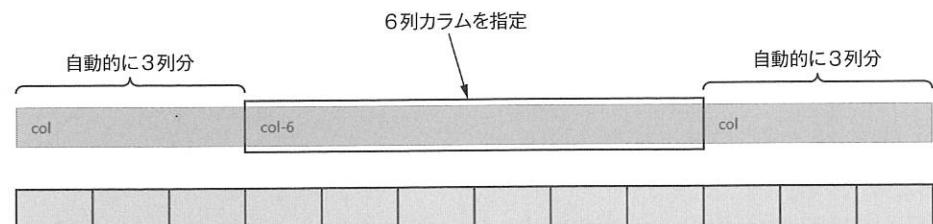
### 2.2.3 1 カラムのみ幅を設定（残りのカラムは自動的に等幅）

flexboxに基づくレイアウトでは、1つのカラム幅を指定すると、それに合わせて残りのカラムは自動的に等幅にリサイズされます。次の例では、2つ目のカラムに col-6 クラスを指定しました。残りのカラムは、幅を指定していませんが、自動的に3列分のカラムになります（リスト 2-7、図 2-9）。

▼リスト 2-7 1 カラムのみ幅を設定（setting-one-column.html）

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col-6">col-6</div><!-- 指定幅：6列カラム -->
    <div class="col">col</div>
  </div>
</div>
```

▼図 2-9 1 カラムのみ幅を設定



#### flexbox とは？

Bootstrap のグリッドシステムには、**flexbox (Flexible Box Layout Module)** が採用されています。flexbox とは、その名のとおり、柔軟なレイアウトを実現できる CSS3 の新しいレイアウトモジュールで、より複雑なレイアウトをよりシンプルなコードで実現することができます。従来の float を使ったレイアウトでは、横並びにするのに要素の幅を指定しなければならなかつたり、float を解除するコードが必要でしたが、flexbox に基づくレイアウトでは、要素の幅が未確定でも、任意の方向に任意の順番で要素を配置することができます。float では実装が難しい垂直方向の整列や要素の高さを合わせることも簡単に実現できます。より詳しい仕様については、W3C の「CSS Flexible Boxes Layout (<https://www.w3.org/TR/css-flexbox-1/>)」を参照してください。

# レスポンシブなグリッドシステム

続いてデバイスごとにカラム数を切り替えるレスポンシブなクラスを見ていきましょう。Bootstrapには、レイアウトを切り替えるためのブレイクポイントが5つ用意されており、カラムに `col-md-*` のようなクラスを追加することでレイアウトを5段階で制御できます。この節では、複数の異なるクラスを組み合わせ、さらに複雑なグリッドレイアウトを構築していきます。

## 2.3.1 5段階のレイアウト制御

レスポンシブWebデザインでは、ある画面サイズを境目として適用させるスタイルを切り替えますが、この境目を**ブレイクポイント**と言います。Bootstrapには、レスポンシブなグリッドレイアウトを作るためのブレイクポイントが、「Extra small (< 576px)」「Small (≥ 576px)」「Medium (≥ 768px)」「Large (≥ 992px)」「Extra large (≥ 1200px)」の計5層用意されています。画面幅が576px以上のSmallであれば「`col-sm-`」、768px以上であれば「`col-md-`」のように、`col`クラスに接頭辞を加えることで、レスポンシブ時のレイアウトを5段階で制御できます。

ブレイクポイントの各層に設定されたレイアウトは、その上にあるすべての層にも適用されます。たとえば、「`col-sm-*`」として「Small」層に設定されたレイアウトは、上層の「Medium」「Large」「Extra large」にも適用されます。「`col-md-*`」として「Medium」層に設定されたレイアウトは、「Large」「Extra large」にも適用されます。

Bootstrapのグリッドシステムが、各デバイスでどのように機能するかを表2-1にまとめます。

▼表2-1 画面サイズと機能との対応表

クラス接頭辞 (「*」は1~12の数字)	.col-*	.col-sm-*	.col-md-*	.col-lg-*	.col-xl-*
ブレイクポイント (画面サイズ)	Extra small 0以上	Small 576px以上	Medium 768px以上	Large 992px以上	Extra large 1200px以上
主な対象デバイス	すべて	スマートフォン	タブレット	PC	大型ディスプレイ
containerの最大幅	なし(自動)	540px	720px	960px	1140px
カラム数			12		
ガターの幅		30px(各カラムの両サイドに15pxずつ)			
ネスト		可			
カラムの並べ替え		可			

また、Bootstrap では、ほとんどのサイズを「em」または「rem」で定義していますが、ブレイクポイントとコンテナ幅については「px」で定義しています。これは「px」で指定されているビューポートの幅が、フォントサイズによって変化しないようにするためです。

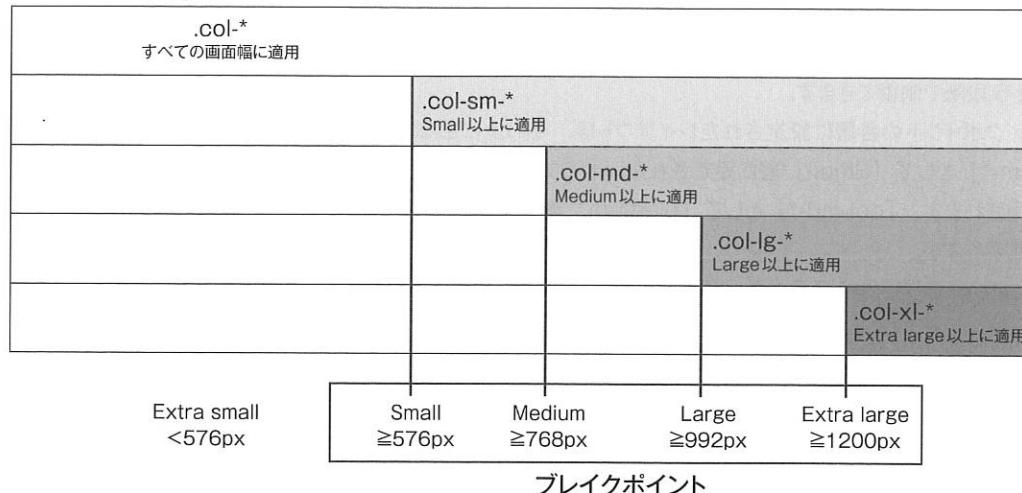
## NOTE

**em と rem と px について**

「em」は font-size の高さを 1 とする単位です。「rem」はルート要素の font-size の高さを 1 とする単位です。「px」はディスプレイ上の 1 ピクセルを最小単位にしたものです。em や rem は相対指定になるのに対し、px は絶対指定の単位になります。

Bootstrap は**モバイルファースト**のコンセプトに基づいて設計されています。従来のように PC 画面先行でデザインし、後からモバイル画面用のスタイルを上書きするしきみではなく、まずモバイル画面を基準として設計し、以降、画面サイズの小さいものからブレイクポイントによってスタイルを上書きしていくしきみになっています。そもそもモバイル画面が基準となっているため、モバイル画面用のスタイルとして付ける col-xs のような接頭辞はありません（図 2-10）。

▼図 2-10 Bootstrap のブレイクポイント



### 2.3.2 ブレイクポイントによる切り替え

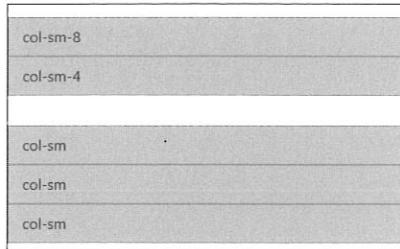
既述のように、Bootstrap では、**col-sm-\*** のようなブレイクポイント付きのクラスによって、表示を切り替えることができます。次の例では、col-sm-\* クラスを使用して、Small (576px) より小さなデバイスでは縦に積み重なり、Small 以上になると水平に並ぶ基本的なグリッドシステムを作成しています。ブラウザのウィンドウサイズを変化させて試してみてください（リスト 2-8、図 2-11）。

▼リスト 2-8 ブレイクポイントによる切り替え (stacked-horizontal.html)

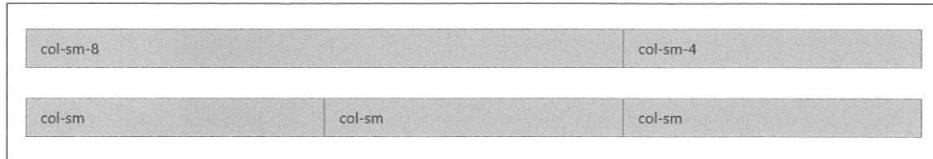
```
<div class="container">
  <div class="row">
    <div class="col-sm-8">col-sm-8</div>
    <div class="col-sm-4">col-sm-4</div>
  </div>
</div>
<div class="container">
  <div class="row">
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
  </div>
</div>
```

▼図 2-11 ブレイクポイントによる切り替え

Small 未満



Small 以上



### 2.3.3 可変幅カラム

**col-auto クラス**を使用したカラムは、コンテンツによって幅が変化するようになります。また、ブレイクポイントによる接頭辞をプラスして、たとえば、Medium (768px) 以上で可変幅にしたい場合、**col-md-auto** のような指定も可能です。

次の例で、col-auto クラスを指定した要素は幅が可変、残りの col を指定した要素は等幅で表示されます。また、col-md-auto クラスを指定した要素は、通常（Medium 未満）は幅が 100%、Medium 以上で可変幅になります（リスト 2-9、図 2-12）。

## ▼リスト 2-9 可変幅カラム (col-auto.html)

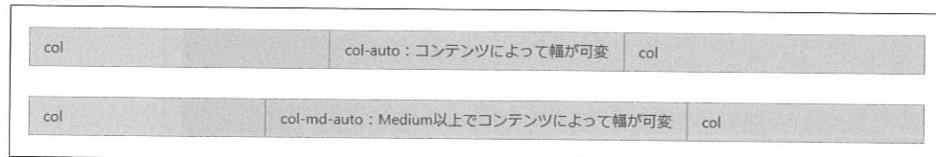
```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col-auto">col-auto : コンテンツによって幅が可変</div>
    <div class="col">col</div>
  </div>
  <div class="row">
    <div class="col">col</div>
    <div class="col-md-auto">col-md-auto : Medium以上でコンテンツによって幅が可変</div>
    <div class="col">col</div>
  </div>
</div>
```

▼図 2-12 可変幅カラムの例

Medium 未満

col	col-auto : コンテンツによって幅が可変
col	
col	
col	col-md-auto : Medium以上でコンテンツによって幅が可変
col	

Medium 以上



### 2.3.4 等幅カラムを複数行に分割

一連の等幅カラムを複数行に分割する場合、改行する箇所に **Sizing ユーティリティ** (P.314 参照) の **w-100 クラス**を挿入する方法も有効です。これは、幅 100%を指定するクラスです。内容が空なので見えませんが、ここで改行され、後に続く要素は 2 行目に表示されます。次項のブレイクポイントで行分割を変化させる場合に便利です（リスト 2-10、図 2-13）。

## ▼リスト 2-10 等幅カラムを複数行に分割する (equal-width2.html)

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
```

```
<div class="w-100"></div>
<div class="col">col</div>
<div class="col">col</div>
</div>
</div>
```

▼図 2-13 等幅カラムを複数行に分割

col	col
col	col

### 2.3.5 行の分割をブレイクポイントで切り替え

ブレイクポイントごとに分割の仕方を切り替える場合には、**Display ユーティリティ**（P.310）を使用する方法もあります。たとえば、表示をしない **d-none クラス** (`display:none`) と、Medium 以上で表示する **d-md-block クラス** (`medium 以上は、display:block`) を組み合わせると、Medium 以上の画面幅では表示され、それ未満では表示されない要素となります。

次の例を見てください（リスト 2-11、図 2-14）。

▼リスト 2-11 行の分割をブレイクポイントで切り替え（multi-row.html）

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="w-100 d-none d-md-block"></div>><!-- Medium以上では{display:block}、未満では{display:none} --&gt;
    &lt;div class="col"&gt;col&lt;/div&gt;
    &lt;div class="col"&gt;col&lt;/div&gt;
  &lt;/div&gt;
&lt;/div&gt;</pre>

```

▼図 2-14 行の分割をブレイクポイントで切り替え  
Medium 未満

col	col	col	col
-----	-----	-----	-----

Medium 以上

col	col
col	col

通常（モバイルの画面サイズ）では、d-none クラス（display:none）が指定された要素は表示されないため、1行（row）に4つのカラムが並んで配置されます。同時に、d-md-block と w-100 の複数のクラスが指定されているため、Medium 以上の画面幅になると幅 100% の要素が表示されます。内容が空なので見えませんが、ここで改行が入り、後に続く要素が2行目に表示されます。

### 2.3.6 複数クラスの組み合わせ

複数のクラスを組み合わせ、より複雑な表示の切り替えを行うことができます（リスト 2-12）。

▼リスト 2-12 複数クラスの組み合わせ（mix-and-match.html）

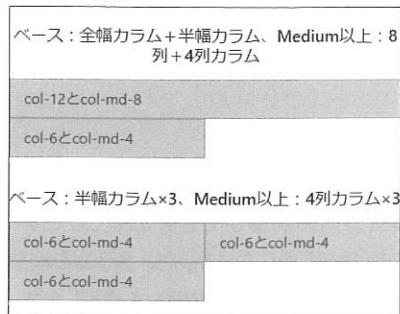
```
<h3>ベース：全幅+半幅カラム、Medium以上：8列+4列カラム</h3>
<div class="container">
  <div class="row"> _____①
    <div class="col-12 col-md-8">col-12とcol-md-8</div> _____②
    <div class="col-6 col-md-4">col-6とcol-md-4</div> _____③
  </div>
</div>

<h3>ベース：半幅カラム×3、Medium以上：4列カラム×3</h3>
<div class="container">
  <div class="row"> _____④
    <div class="col-6 col-md-4">col-6とcol-md-4</div>
    <div class="col-6 col-md-4">col-6とcol-md-4</div>
    <div class="col-6 col-md-4">col-6とcol-md-4</div>
  </div>
</div>
```

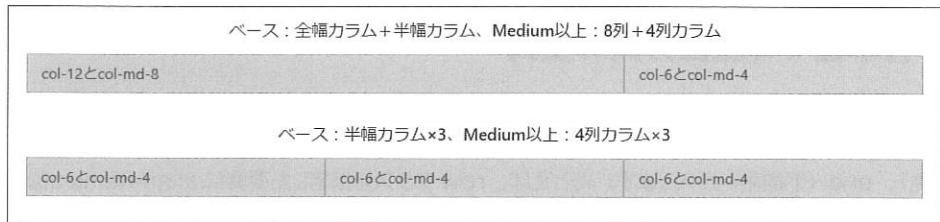
①の row を見てください。「col-12」は12カラム分、「col-6」は6カラム分の幅でコンテンツを表示するクラスです。Bootstrap は**12カラム**でできているので、②の「col-12」は、1行目に12カラム分、つまり、全幅で表示されます。③の「col-6」は2行目に6カラム分、つまり、半分の幅で表示されます。ただし、それぞれ「col-md-8」と「col-md-4」の複数クラスが指定されているので、Medium（768px 以上）のとき、②の要素は8カラム分、③の要素は4カラム分で表示されます。「8カラム+4カラム=12カラム」となるので、Medium サイズ以上のときは、12カラムに収まり、1行で表示されます（図 2-15）。

同じように④の row を見てみましょう。「col-6」が3つあります。6カラム分なので、1行目に2つの要素がちょうど収まり、2行目に最後の要素が半分の幅で表示されます。ただし、それに「col-md-4」も指定されているので、Medium サイズ以上のときは4カラム分が3つ、計12カラムとなり、1行に収まって表示されます。

▼図 2-15 上: Medium 未満、下: Medium 以上  
Medium 未満



Medium 以上



以上見てきたように、Bootstrap では、12 カラムのグリッドに沿ってコンテンツを配置し、ブレイクポイントによってコンテンツのカラム数を切り替えることで**レスポンシブ Web デザイン**を実装できるしくみになっています。カラム数を切り替える方法として、基本の **col クラス**や、ブレイクポイントの接頭辞の付いた **col-md** や **col-lg クラス**などが用意されている他、w-100 クラスや d-none クラスなどの**ユーティリティクラス**を組み合わせることにより、より複雑なレイアウトを可能にしています。

# カラムの整列

Bootstrap 3 で採用されていた float を使ったレイアウトでは、垂直方向の制御ができませんでしたが、Bootstrap 4 からは flexbox によるレイアウトが導入され、垂直方向の制御が可能になりました。カラムを水平方向・垂直方向に整列するには、Bootstrap で定義済みの **Flex ユーティリティ** (P.322 参照) を使用します。

## 2.4.1 行単位での垂直方向の整列

行の中で垂直方向の整列をしたいときは、row クラスを指定した要素に、**align-items ユーティリティ**を追加します。クラス名の表記は、align-items-{ プロパティ } となっていて、プロパティには、start (上揃え)、center (中央揃え)、end (下揃え) が入ります。たとえば、row クラスを指定した要素に align-items-start クラスを指定すると上揃えになります (表 2-2)。

▼表 2-2 行単位での垂直方向の整列

クラス	説明	スタイル
align-items-start	上揃え	align-items:flex-start
align-items-center	中央揃え	align-items:flex-center
align-items-end	下揃え	align-items:flex-end

次の例では、各行にそれぞれ上揃え、中央揃え、下揃えのクラスを指定しています (リスト 2-13、図 2-16)。

▼リスト 2-13 行単位での垂直方向整列 (vertical-alignment.html)

```
<h3>垂直上揃え : align-items-start</h3>
<div class="container">
  <div class="row align-items-start">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
</div>
<h3>垂直中央揃え : align-items-center</h3>
<div class="container">
  <div class="row align-items-center">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
```

```
</div>
<h3>垂直下揃え : align-items-end</h3>
<div class="container">
  <div class="row align-items-end">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
</div>
```

▼図 2-16 行単位での垂直方向整列



## 2.4.2 カラム単位での垂直方向の整列

行単位ではなく、カラム単位で垂直方向の制御をしたいときは、col クラスを指定した要素に、**align-self-{プロパティ}クラス**（プロパティには start、center、end が入ります）を追加します。たとえば、align-self-start クラスを指定すると、そのカラムは上揃えになります（表 2-3）。

▼表 2-3 カラム単位での垂直方向の整列

クラス	説明	スタイル
align-self-start	上揃え	align-self:flex-start
align-self-center	中央揃え	align-self:flex-center
align-self-end	下揃え	align-self:flex-end

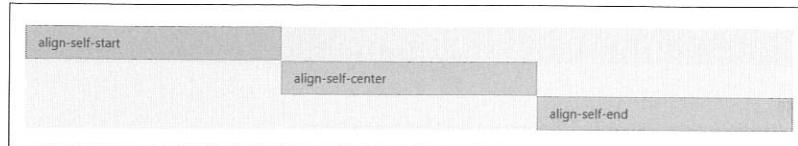
次の例では行内の各カラムに、上揃え、中央揃え、下揃えのクラスを指定しています（リスト 2-14、図 2-17）。

▼リスト 2-14 カラム単位での垂直方向整列（col-align-self.html）

```
<div class="container">
  <div class="row">
```

```
<div class="col align-self-start">align-self-start</div><!-- 垂直上揃え -->
<div class="col align-self-center">align-self-center</div><!-- 垂直中央揃え -->
<div class="col align-self-end">align-self-end</div><!-- 垂直下揃え -->
</div>
</div>
```

▼図 2-17 カラム単位での垂直方向整列の例



### 2.4.3 水平方向の整列

水平方向の整列には、rowを指定した要素に **justify-content-{ プロパティ } クラス**（プロパティには start、center、end、around、between が入ります）を追加します（表 2-4）。

▼表 2-4 水平方向の整列

クラス	説明	スタイル
justify-content-start	左揃え	justify-content:flex-start
justify-content-center	中央揃え	justify-content:flex-center
justify-content-end	右揃え	justify-content:flex-end
justify-content-around	等間隔に配置	justify-content:space-around
justify-content-between	両端から均等配置	justify-content:space-between

justify-content-around は、カラムを等間隔に配置、各カラムの左右には半分ずつのスペースが付きます。justify-content-between は、カラムを両端にくつつけ、残りを等間隔に配置します（リスト 2-15、図 2-18）。

▼リスト 2-15 水平方向の整列 (justify-content.html)

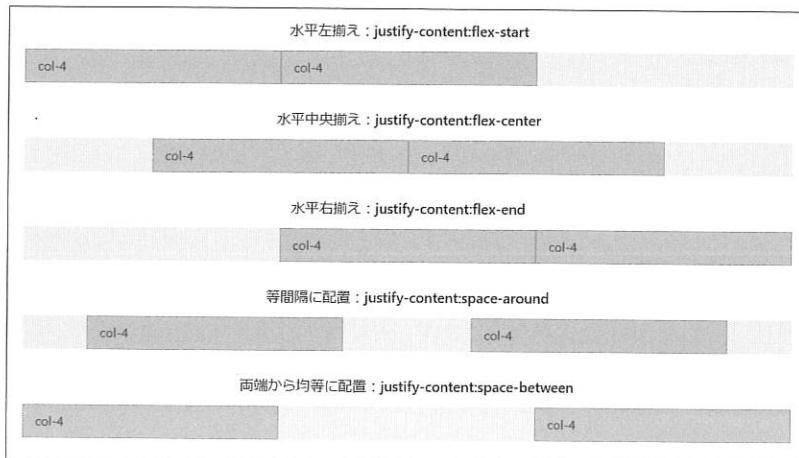
```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">col-4</div>
    <div class="col-4">col-4</div>
  </div>
</div>
<h3>水平中央揃え : justify-content:flex-center</h3>
<div class="container">
  <div class="row justify-content-center">
    <div class="col-4">col-4</div>
    <div class="col-4">col-4</div>
  </div>
</div>
<h3>水平右揃え : justify-content:flex-end</h3>
```

```

<div class="container">
  <div class="row justify-content-end">
    <div class="col-4">col-4</div>
    <div class="col-4">col-4</div>
  </div>
</div>
<h3>等間隔に配置：justify-content:space-around</h3>
<div class="container">
  <div class="row justify-content-around">
    <div class="col-4">col-4</div>
    <div class="col-4">col-4</div>
  </div>
</div>
<h3>両端から均等に配置：justify-content:space-between</h3>
<div class="container">
  <div class="row justify-content-between">
    <div class="col-4">col-4</div>
    <div class="col-4">col-4</div>
  </div>
</div>

```

▼図 2-18 水平方向の整列



#### 2.4.4 ガターの削除

カラムに設定されているガターサイズは、**no-gutters クラス**を使用して取り除くことができます。no-gutters クラスにはリスト 2-16 のスタイルが設定されています。

▼リスト 2-16 no-gutters クラスのスタイル

```
.no-gutters {
  margin-right: 0;
```

```

    margin-left: 0;
}
.no-gutters > .col,
.no-gutters > [class*="col-"] {
  padding-right: 0;
  padding-left: 0;
}

```

これによって、row を指定した要素については左右のマイナスマージン (-15px) が、その直接の子要素であるカラムについては左右のパディング (15px) が「0」に上書きされ、ガーターサイズが取り除かれます。

また、親要素の container クラス（あるいは container-fluid クラス）を指定した要素を外せば、Edge to Edge（幅が画面の端から端まであるような）デザインが可能になります（リスト 2-17、図 2-19）。

▼リスト 2-17 ガターを削除したカラム（no-gutters.html）

```

<h3>no-guttersありの場合</h3>
<div class="container">
  <div class="row no-gutters">
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
</div>
<h3>no-guttersなしの場合（参考）</h3>
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
</div>
<h3>コンテナなし、no-guttersありの場合</h3>
<div class="row no-gutters">
  <div class="col">col</div>
  <div class="col">col</div>
</div>

```

▼図 2-19 ガターを削除したカラム



ガターサイズのさらなるカスタマイズは、Bootstrap で定義済みの Spacing ユーティリティ (P.318 参照) を使って行うこともできます。

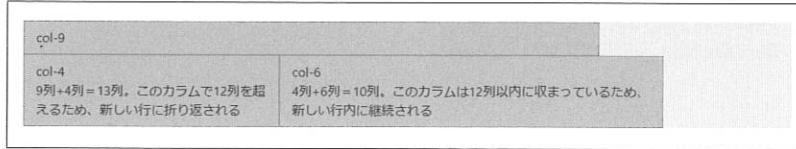
## 2.4.5 カラムの折り返し

1つの行に 12 列分以上のカラムが配置されている場合、12 を超える分のカラムは新しい行に折り返されます。次の例では、col-9 の後に col-4 を指定しています。9 列 + 4 列 = 13 列で 12 列を超えるため、col-4 のカラムで行が折り返されます (リスト 2-18、図 2-20)。

▼リスト 2-18 カラムの折り返し (column-wrapping.html)

```
<div class="container">
  <div class="row">
    <div class="col-9">col-9</div>
    <div class="col-4">col-4<br>
      9列+4列=13列。このカラムで12列を超えるため、新しい行に折り返される</div>
    <div class="col-6">col-6<br>
      4列+6列=10列。このカラムは12列以内に収まっているため、新しい行内に継続される</div>
  </div>
</div>
```

▼図 2-20 カラムの折り返し



# 2

# 5 カラムの並べ替え

**order-\* クラス** (\*には数字が入ります) を使用すると、HTML の構造を変えずにコンテンツの視覚的順序を変えることができます。

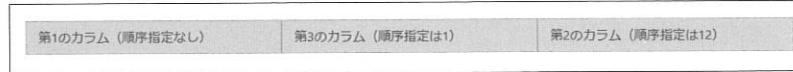
## 2.5.1 order-\* クラスで並べ替え

HTML の構造を変えずに、「order-1」～「order-12」のクラスを指定することで、並べ替えることができます。また、このクラスにも「order-sm-\*」や「order-md-\*」のように、ブレイクポイントごとの順序を設定することができます（リスト 2-19、図 2-21）。

▼リスト 2-19 コンテンツの並べ替え（order-1～order-12）（order-1.html）

```
<div class="container">
  <div class="row">
    <div class="col">第1のカラム（順序指定なし）</div>
    <div class="col order-12">第2のカラム（順序指定は12）</div>
    <div class="col order-1">第3のカラム（順序指定は1）</div>
  </div>
</div>
```

▼図 2-21 コンテンツの並べ替え（order-1～12）



この他、**order-first クラス**を使用して、カラムにスタイル {order: -1} を適用し、簡単に順序を入れ替えることもできます。order プロパティは、フレックスアイテムの順序を指定するもので、指定された値が小さい要素から配置されます。負の値も指定でき、-1 を指定することで順番を最初にしています（リスト 2-20、図 2-22）。

▼リスト 2-20 コンテンツの並べ替え（order-first）（order-first.html）

```
<div class="container">
  <div class="row">
    <div class="col">第1のカラム（順序指定なし）</div>
    <div class="col">第2のカラム（順序指定なし）</div>
    <div class="col order-first">第3のカラム（順序指定は1）</div>
  </div>
</div>
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

▼図 2-22 コンテンツの並べ替え (order-fitst)



## 2.5.2 カラムのオフセット

カラムのオフセット（右移動）には、**offset-\* クラス**を使用する方法と、Spacing ユーティリティ（P.318 参照）のマージン設定を使用する方法の 2 つがあります。

### 1 オフセット用クラスを使用したオフセット

**offset-\* クラス**を使用して列を右に移動します。「\*」に 0 ~ 12 の列数を指定し、カラムの左マージンを \* 列分だけ増加させます。また、このクラスにも **offset-sm-\*** や **offset-md-\*** のように、ブレイクポイントごとのオフセットを設定することができます。

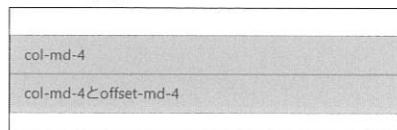
次の例では、2 つ目のカラムに offset-md-4 を指定しています。Medium 以上になると、4 カラム分右に表示されます（リスト 2-21、図 2-23）。

▼リスト 2-21 オフセット用クラスによるカラムのオフセット (offset.html)

```
<div class="container">
  <div class="row">
    <div class="col-md-4">col-md-4</div>
    <div class="col-md-4 offset-md-4">col-md-4 と offset-md-4</div><!-- Medium以上で4列分左に移動 -->
  </div>
</div>
```

▼図 2-23 オフセット用クラスによるカラムのオフセット

Medium 未満



Medium 以上



特定のブレイクポイントでオフセットをリセットする必要がある場合、**offset-0 クラス**を指定します。次の例では、Small 以上で 2 列分オフセットされ、Medium 以上になるとオフセットがリセットされます（リスト 2-22、図 2-24）。

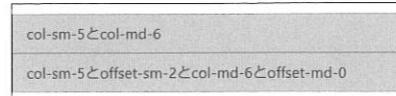
▼リスト 2-22 オフセットのリセット (offset-reset.html)

```
<div class="container">
  <div class="row">
```

```
<div class="col-sm-5 col-md-6">col-sm-5とcol-md-6</div>
<!-- Medium以上でオフセットをクリア -->
<div class="col-sm-5 offset-sm-2 col-md-6 offset-md-0">col-sm-5とoffset-sm-2とcol-md-6と
offset-md-0</div>
</div>
</div>
```

▼図 2-24 オフセットのリセット

Small 未満



Small 以上



Medium 以上



## 2 Spacing ユーティリティのマージン設定を使用したオフセット

Spacing ユーティリティ（P.318 参照）のマージン設定を使用して、隣接カラム同士をオフセットさせることができます。次の例では、**ml-md-auto クラス**を使って、Medium 以上で margin-left を自動設定しています（リスト 2-23、図 2-25）。

▼リスト 2-23 オフセットのリセット (offset-spacing.html)

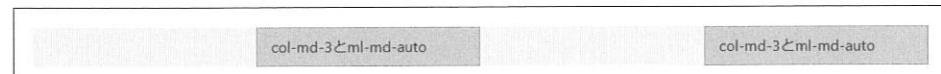
```
<div class="container">
<div class="row">
<!-- Medium以上でml (margin-left) を自動設定 -->
<div class="col-md-3 ml-md-auto">col-md-3とml-md-auto</div>
<!-- Medium以上でml (margin-left) を自動設定 -->
<div class="col-md-3 ml-md-auto">col-md-3とml-md-auto</div>
</div>
</div>
```

▼図 2-25 Spacing ユーティリティのマージン設定を使用したオフセット

Medium 未満



Medium 以上



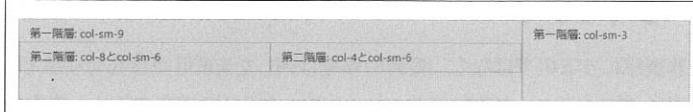
### 2.5.3 グリッドレイアウトの入れ子（ネスト）

グリッドレイアウトを**入れ子（ネスト）**することで、より複雑なコンテンツのレイアウトが可能になります。col-\* クラスを指定した要素内に、新たに row クラスと col-\* クラスを指定した要素のセットを埋め込むことで、コンテンツのレイアウトをネストできます（リスト 2-24、図 2-26）。

▼リスト 2-24 グリッドレイアウトの入れ子（grid-nesting.html）

```
<div class="container">
  <div class="row">
    <div class="col-sm-9">第一階層: col-sm-9
      <!-- /.row と /.col-* のセットをネスト -->
      <div class="row">
        <div class="col-8 col-sm-6">第二階層: col-8とcol-sm-6</div>
        <div class="col-4 col-sm-6">第二階層: col-4とcol-sm-6</div>
      </div>
    </div>
    <div class="col-sm-3">第一階層: col-sm-3</div>
  </div>
</div>
```

▼図 2-26 グリッドレイアウトの入れ子



1

2

3

4

5

6

7

8

9

10

# レイアウトのための ユーティリティ

Bootstrap には、コンテンツの表示、非表示、整列、余白の調整などに使用できる数十種類のユーティリティクラスが組み込まれています。ここではレイアウトによく使用する代表的なクラスを抜粋して紹介します。

## 2.6.1 Display ユーティリティ

display プロパティの値をレスポンシブで切り替えるには、**Display ユーティリティ**のクラスを使用します。たとえば、d-none クラスを指定すると display: none が適用され、要素が非表示になります。Display ユーティリティは、グリッドシステム、コンテンツ、コンポーネントと組み合わせて使用でき、特定のデバイスで要素を表示または非表示にしたりすることができます（P.310 参照）。

## 2.6.2 Visibility ユーティリティ

Display ユーティリティで要素を表示・非表示にするのではなく、要素の領域は残したまま可視性を切り替えるには、**Visibility ユーティリティ**（visibility プロパティ）が有効です。invisible クラスを指定すると、要素の領域自体は残したまま、内容を非表示にすることができます（P.354 参照）。

## 2.6.3 Flex ユーティリティ

Bootstrap のコンポーネントのほとんどは、flexbox 対応で構築されていますが、自分で追加したコンテンツを flexbox 対応にするには、**d-flex クラス**（display:flex）または d-sm-flex クラスなどを追加する必要があります。このクラスを指定した要素の子要素は**フレックスアイテム**と呼ばれ、Flex ユーティリティで自由に配置できるようになります（P.322 参照）。

## 2.6.4 Spacing ユーティリティ

要素とコンポーネントの間隔（マージンやパディング）は、**Spacing ユーティリティ**のクラスを使用して制御できます。たとえば、mt-0 クラスを使用すると margin-top:0 が適用され、上マージンが 0 になります（P.318 参照）。

各ユーティリティの詳細は第 8 章を参照してください。