

# CSS Multi-column Layout Module Level 1

**W3C Working Draft, 12 February 2021**

## 抄録

本仕様書は、Web用スタイルシート言語であるCSSにおける複数カラムレイアウトについて規定する。本仕様で規定されている機能を利用して、コンテンツを複数の列の間に隙間やルールを設けて流すことができる。

CSSは、構造化された文書（HTMLやXMLなど）の画面や紙面などでの描画を記述するための言語である。

## この文書のステータス

このセクションでは、この文書の発行時の状態を説明します。他の文書がこの文書に取って代わることがある。現在のW3C出版物及びこの技術報告書の最新改訂版のリストは、<https://www.w3.org/TR/> のW3C技術報告書インデックスに掲載されている。

この文書は、CSS作業部会が作業草案として発行したものである。ワーキングドラフトとして出版されたことは、W3Cメンバーシップによる承認を意味するものではない。

この文書は草案であり、いつでも他の文書に更新されたり、置き換えられたり、廃止されたりする可能性がある。この文書を作業中以外のものとして引用することは不適切です。

以下のように、タイトルに仕様コード "css-multicol" を含めてGitHubに課題を提出することでフィードバックを送ってください(推奨)。"[css-multicol] ...コメントの要約...." のようにしてください。すべての問題とコメントはアーカイブされます。また、(アーカイブされた)公開メーリングリスト [www-style@w3.org](mailto:www-style@w3.org) にフィードバックを送ることもできます。

この文書は、2020年9月15日のW3Cプロセス文書に準拠する。

この文書は、W3C特許ポリシーの下で運営されているグループによって作成された。W3Cは、グループの成果物に関連して行われた特許開示の公開リストを維持している。本質的クレームを含むと考えられる特許を実際に知っている個人は、W3C特許ポリシーの第6項に従って情報を開示しなければならない。

## 1. 序章

(このセクションは規範ではありません)。

このモジュールでは、CSSにおける複数列レイアウトについて説明する。このドキュメントで説明する機能を使用することで、スタイルシートは、要素の内容を複数の列にレイアウトすることを宣言することができます。

CSS の他のレイアウト方法は、親要素に適用されると、直接の子要素の表示プロパティを変更する。例えば、3列のグリッド・レイアウトが作成された場合、グリッド・コンテナの直接の子要素はグリッド・アイテムとなり、必要に応じて追加の行が作成され、セルごとに 1つの要素がカラム・トラックに配置されます。

しかし、マルチカラムコンテナの子要素は通常のフローを継続し、そのフローは複数のカラムに配置されます。これらの列は柔軟なインラインサイズを持っているため、表示される列のサイズや数を変更することで利用可能なスペースに対応します。

複数カラムのレイアウトは、CSSで記述するのは簡単です。以下に簡単な例を示します。

### 例1

```
body { column-width: 12em }
```

正確なカラム数は利用可能なスペースに依存します。

カラム数はスタイルシートで明示的に設定することもできます。

### 例2

```
body { column-count: 2 }
```

この場合、カラムの数は固定で、カラムの幅は利用可能な幅に応じて変化します。

短縮型のcolumnsプロパティを使用して、1つの宣言でどちらか、または両方のプロパティを設定することができます。

### 例3

これらの例では、カラムの数、カラムの幅、および数と幅の両方がそれぞれ設定されています。

```
body { columns: 2 }
body { columns: 12em }
body { columns: 2 12em }
```

このモジュールで導入されたもう一つのプロパティ群は、カラム間のギャップとルールを記述します。

### 例4

```
body {
  column-gap: 1em;
  column-rule: thin solid black;
}
```

上の例の最初の宣言では、隣接する2つの列の間のギャップを1emに設定しています。列の隙間はパディング領域に似ています。隙間の中央には、column-ruleプロパティによって記述されるルールがあります。

column-ruleプロパティの値は、CSSのborderプロパティの値と似ています。borderと同様に、column-ruleは短縮プロパティです。

### 例5

この例では、上記の例の短縮型のcolumn-rule宣言を展開しています。

## css\_syntax

```
body {  
    column-gap: 1em;  
    column-rule-width: thin;  
    column-rule-style: solid;  
    column-rule-color: black;  
}
```

column-fillとcolumn-spanプロパティは、スタイルシートに複数列レイアウトの視覚的表現の幅を与えます。

### 例6

この例では、列はバランスがとれている、つまりほぼ同じ長さになるように設定されています。また、h2要素はすべての列にまたがるように設定されています。

```
div { column-fill: balance }  
h2 { column-span: all }
```

この仕様では、10個の新しいプロパティを導入していますが、そのすべてが上記の例で使用されています。

すべての列プロパティが初期値を持つ場合、要素のレイアウトは、1つの列だけを持つ複数列レイアウトと同じになります。

### 例7

列の段間（斜めのハッチング）と列のルーラー（定規）は、パディング（十字のハッチング）を持つ複数列コンテナのこのサンプルのレンダリングで示されています。ハッチングされた部分は、説明のためだけに存在しています。実際の実装では、これらの領域は背景によって決定されます。2番目の画像は、列のルーラー（定規）を持つ複数列コンテナのレンダリングを示しています。

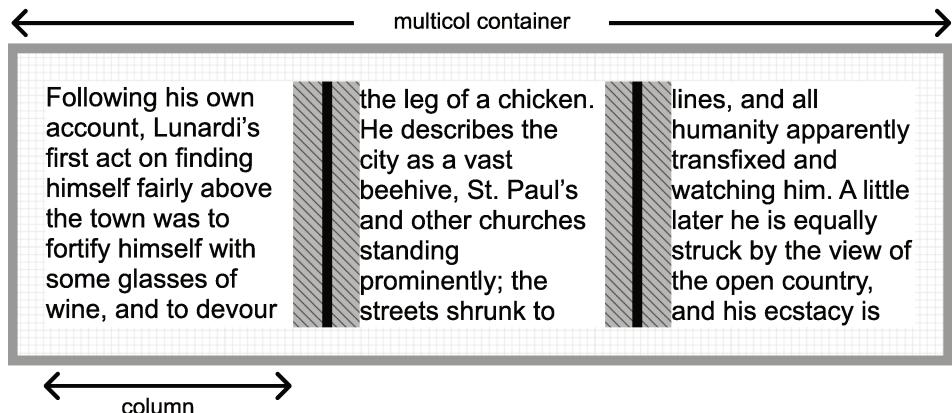


figure1 A multi-column layout with the non-visible column-span and padding inside the multicol container

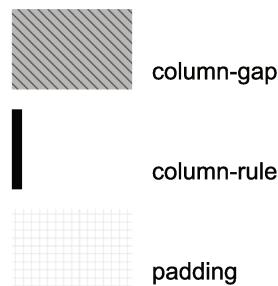


figure1-2

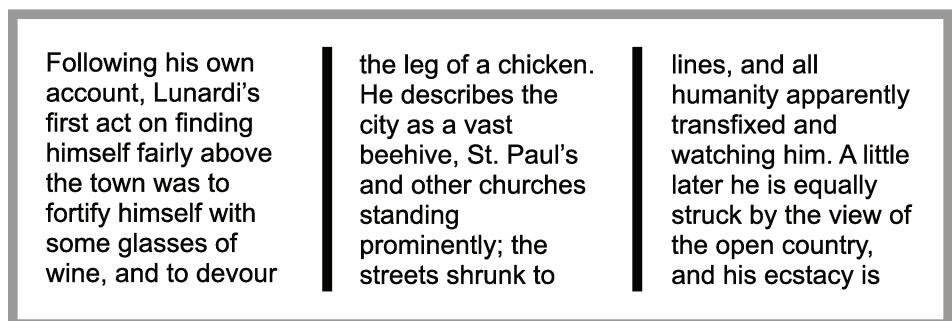


figure2 The same layout as in the first image, as it would be displayed by an implementation.

## 1.1. 値の定義

本仕様は、[CSS-VALUES-3]の値定義構文を用いて、[CSS21]のCSSプロパティ定義規定に従う。本仕様で定義されていない値の型は、CSS Values & Units [CSS-VALUES-3]で定義されている。他のCSSモジュールと組み合わせることで、これらの値型の定義が拡張されることがある。

定義に列挙されているプロパティ固有の値に加えて、本仕様で定義されているすべてのプロパティは、CSS全体のキーワードをプロパティ値としても受け入れる。可読性のために、それらは明示的に繰り返していない。

## 2. マルチカラムモデル

`column-width` または `column-count` プロパティが `auto` でない要素は、マルチカラムコンテナ（略して `multicol` コンテナ）を確立し、マルチカラムレイアウトのためのコンテナとして機能する。

従来のCSSボックス・モデルでは、要素の内容は対応する要素のコンテンツ・ボックスに流れ込みます。マルチカラム・レイアウトでは、カラム・ボックス（略してカラム）と呼ばれる匿名の断片化コンテナで形成された断片化コンテキストが導入されます。これらのカラム・ボックスは、マルチカラム・コンテナのコンテンツが流れる独立したブロック・フォーマット・コンテキストを確立し、その非配置子要素のための包含ブロックを形成します。

### 例8

この例では、このようなルールで画像の幅を設定しています。

```
img {  
  display: block;  
  width: 100%;  
}
```

カラムボックスが新しいブロックフォーマットのコンテキストを作成すると考えると、幅はカラムボックスに対して相対的に計算されます。そのため、画像がカラムボックスからはみ出することはありません。

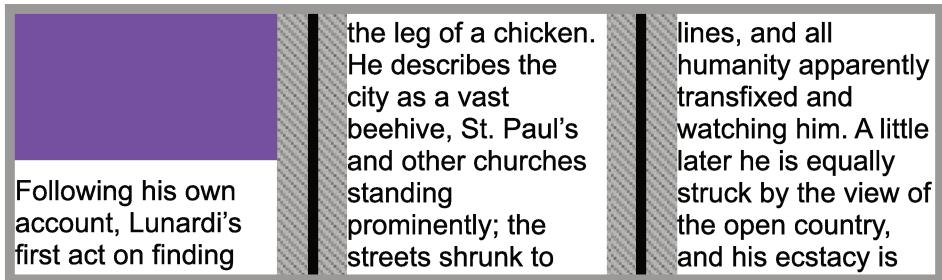


figure3 画像は、表示されるカラムボックスによって拘束されます。

#### 例9

カラムボックスが新しいブロックフォーマットのコンテキストを作成すると考えると、multicolコンテナの最初の子要素に設定されたトップマージンは、multicolコンテナの余白では崩れません。

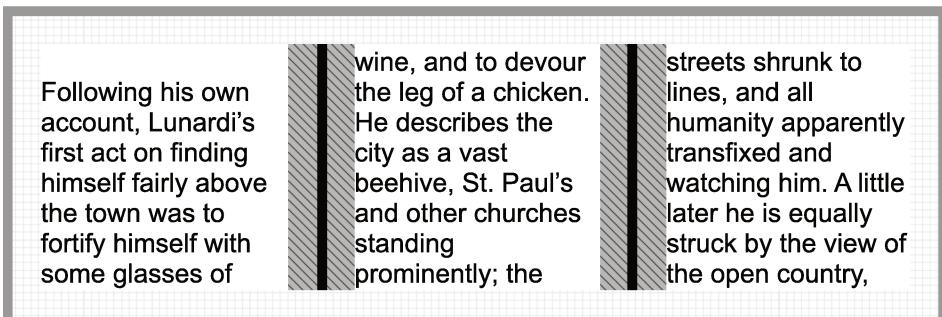


figure4 最初の段落の上の余白は崩れておらず、multicolコンテナ内の最初の行の上に1emの余白を残しています。

複数列レイアウト内に表示されるフロートは、フロートが表示される列ボックスを基準にして配置されます。

## 例10

この例では、この CSS フラグメントは画像の表示を記述しています。

```
img {  
    display: block;  
    float: right;  
}
```

HTMLでは、「鶏の足」という文末の後に画像が表示されます。

Following his own account, Lundardi's first act on finding himself fairly above the town was to fortify himself with some glasses of wine, and to devour

the leg of a chicken.  
He describes the city as a vast beehive, St. Paul's and other churches standing prominently; the

streets shrunk to lines, and all humanity apparently transfixed and watching him. A little later he is equally struck by the view of the open country,

figure5 画像は、表示されているカラムボックス内に浮かんでいます。

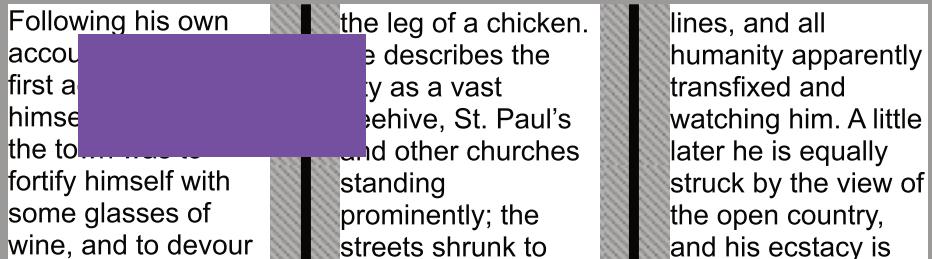
ブロック軸の列ボックスでオーバーフローした内容が断片化され、次の列ボックスで継続されます。

注：無名のボックスである列ボックスは、絶対的に配置されたボックスの包含ブロックにはなりません。そのようなボックスのための包含ブロックを確立する position プロパティは、マルチコルコンテナに適用され、それが主ボックスとなります。

**例11**

この例では、『multi-column container』は『position: relative』、これによって内容ブロックになります。画像は複数列コンテナの直接の子であり、『position: absolute』を持ちます。画像は、複数列コンテナの位置を取り、カラムボックスの位置ではなく、複数列コンテナの位置を取ります。

```
.container {
  position: relative;
  column-count: 3;
}
img {
  position: absolute;
  top: 20px;
  left: 40px;
}
```



**figure6** 図は、絶対的に配置された画像がマルチコルコンテナを参照して配置され、カラムボックスではないことを示しています。

カラムボックスは、マルチコルコンテナのインラインベース方向に並べられ、マルチコルラインに配置されています。カラム幅は、インライン方向のカラムボックスの長さです。カラムの高さは、ブロック方向のカラムボックスの長さである。行内のすべてのカラムボックスは同じカラム幅を持ち、行内のすべてのカラムボックスは同じカラム高さを持つ。

マルチカラムコンテナ内の各マルチカラム行内では、隣接するカラムボックスはカラムギャップによって区切られており、その中にはカラムルールが含まれている場合があります。同じ複数列コンテナ内のすべての列ギャップは等しくなります。同じ複数列コンテナ内のすべての列規則が表示されている場合は、それもすべて等しくなります。

最も単純なケースでは、マルチカラムコンテナは1行のカラムのみを含み、各カラムの高さはマルチカラムコンテナのコンテンツボックスの使用済みの高さと同等になります。しかし、断片化またはスパナは、マルチカラムコンテナの内容を複数のマルチカラム行に分割することができます。

複数列コンテナがページ分割されている場合、各列の高さはページによって制約され、内容は次のページの列ボックスの新しい行に続けられます。

この効果は、スパング要素が複数列のコンテナを分割したときにも同じです：スパング要素の前の列は、その内容に合わせてバランスが取れて短くなります。スパニング要素の後の内容は、新しい行の後続の列ボックスに流れ込みます。

## 例12

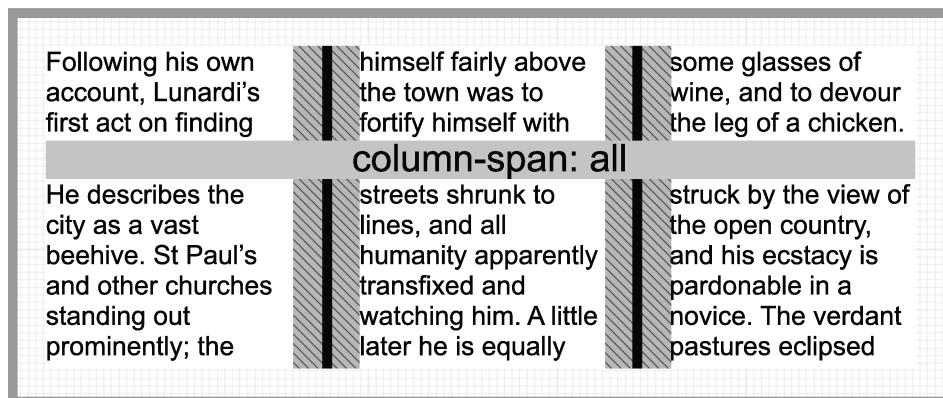


figure7 A demonstration of how the spanning element divides the multicol container.

したがって、マルチカラムコンテナは、その内容が一連のマルチカラム行とマルチカラムスパナで構成された新しい独立した書式設定コンテキストを確立する通常のブロックコンテナです。各マルチカラム行は、そのカラムボックスのためにマルチカラム書式設定コンテキストを確立するブロックレベルボックスとして動作し、各スパナは、通常通り、その表示値に応じた型で独立した書式設定コンテキストを確立するブロックレベルボックスとして動作します。

入れ子になった複数列コンテナは許可されていますが、実装固有の制限がある場合があります。

注意：カラムボックスにプロパティ/値を設定することはできません。例えば、特定のカラムボックスの背景を設定することはできませんし、カラムボックスにはパディング、マージン、ボーダーの概念がありません。将来の仕様では、追加機能が追加される可能性があります。例えば、幅の異なるカラムや背景の異なるカラムがサポートされるかもしれません。

注意：ビューポートよりもカラムの高さが大きいマルチコルコンテナは、アクセシビリティの問題を引き起こす可能性があります。

### 3. コラムの数と幅

複数カラムのコンテンツをレイアウトする際には、カラムの数と幅を見つけることが基本となります。これらのプロパティは、カラム数と幅を設定するために使用されます。

- column-count
- column-width

3番目のプロパティであるcolumnsは、column-widthとcolumn-countの両方を設定する略式プロパティです。

明示的な列の区切り、内容、高さの制約などの他の要因が、実際の列の数や幅に影響を与えることがあります。

#### 3.1. 列のサイズ:column-widthプロパティ

名前	列幅
値	auto または 長さ『0～∞』
初期値	auto
適用対象	block containers except table wrapper boxes
継承	なし
割合	該当なし
計算値	the keyword auto or 絶対値
正準順序	文法ごと

## css\_syntax

このプロパティは、マルチカラムコンテナ内のカラムの幅を記述します。

- **auto** カラムの幅が他のプロパティ(例えば、自動ではない値を持っている場合は、カラムカウント)によって決定されることを意味します。
- **<length [0,∞]>** 最適なカラム幅を記述します。実際の列幅は、(利用可能なスペースを埋めるために)広くしてもよいし、(利用可能なスペースが指定された列幅よりも小さい場合のみ)狭くしてもよい。負の値は認められません。使用される値は、最小1pxにクランプされます。

### 例13

例えば、以下のスタイルシートを考えてみましょう。

```
div {  
  width: 100px;  
  column-width: 45px;  
  column-gap: 0;  
  column-rule: none;  
}
```

幅100pxの要素の中には、幅45pxのカラムが2つ入る余地があります。空いたスペースを埋めるために、実際のカラムの幅は50pxに増加します。

### 例14

また、このスタイルシートも考えてみてください。

```
div {  
  width: 40px;  
  column-width: 45px;  
  column-gap: 0;  
  column-rule: none;  
}
```

空いているスペースは指定された列幅よりも小さくなるため、実際の列幅は小さくなります。

縦書きのテキストでもカラム幅が使えるようにするために、カラム幅とは、カラム内の行ボックスの長さを意味します。

注意：列幅を多少柔軟にする理由は、多くの画面サイズに対応できるスケーラブルなデザインを実現するためです。正確なカラム幅を設定するには、カラムギャップとマルチコルコンテナの幅（横書きテキストを想定）も指定する必要があります。

### 3.2. カラム数: `column-count`プロパティ

名前	<b>column-count</b>
値	<code>auto</code>   整数 <code>1~∞</code>
初期値	<code>auto</code>
適用対象	ボックスを含有しないテーブルを除くブロック
継承	なし
割合	該当なし
算出値	指定値
正準順序	文法ごと
アニメーション可能	計算された値による

このプロパティは、マルチコルコンテナの列数を記述します。

- `auto` カラム数が他のプロパティ(例えば、自動ではない値を持つ場合はカラム幅)によって決定されることを意味します。
- 整数`1~∞` 要素の内容が流れる最適なカラム数を記述します。`column-width`と`column-count`の両方が自動ではない値を持つ場合、整数值は最大のカラム数を記述します。

**例15**

```
body { column-count: 3 }
```

### 3.3. columnsプロパティ(『column-width』と『column-count』の短縮記法)

名前	columns
値を指定します。 <'column-width'>	
イニシャル	個別のプロパティを参照してください。
該当するもの	個別の物件を見る
継承	個別のプロパティを参照
パーセンテージ	個別物件を参照
計算値	個々のプロパティを参照
アニメーションの種類	個々のプロパティを参照してください。
正準順序	文法ごと

column-widthとcolumn-countを設定するための略式プロパティです。省略された値は初期値に設定されます。

**例16**

ここでは、columnsプロパティを使用した有効な宣言をいくつか紹介します。

```
columns: 12em;      /* column-width: 12em; column-count: auto */
columns: auto 12em; /* column-width: 12em; column-count: auto */
columns: 2;          /* column-width: auto; column-count: 2 */
columns: 2 auto;    /* column-width: auto; column-count: 2 */
columns: auto;       /* column-width: auto; column-count: auto */
columns: auto auto; /* column-width: auto; column-count: auto */
```

### 3.4. 擬似アルゴリズム(Pseudo-algorithm)

以下の擬似アルゴリズムは、『N』 column-count と『W』 column-width の使用値を決定します。擬似アルゴリズムにはもう一つの変数があります。『U』はmulti-columnコンテナの幅です。

注記：複数列コンテナの使用済み幅『U』は、要素の内容に依存することがあります、その場合は『column-count』と『column-width』プロパティの計算値にも依存します。この仕様では、『U』がどのように計算されるかは定義されていません。別のモジュール（おそらく、Basic Box Model [CSS3BOX] や Intrinsic & Extrinsic Sizing Module [CSS3-SIZING]）がこれを定義することが期待されています。

`floor(X)` 関数は、最大の整数  $Y \leq X$  を返します。

```
(01) if ((column-width = auto) and (column-count = auto)) then
(02)   exit; /* not a multicol container */
(03) if column-width = auto then
(04)   N := column-count
(05) else if column-count = auto then
(06)   N := max(1,
(07)           floor((U + column-gap)/(column-width + column-gap))
(08) else
(09)   N := min(column-count, max(1,
(10)           floor((U + column-gap)/(column-width + column-gap))
And:
(11) W := max(0, ((U + column-gap)/N - column-gap))
```

自動繰り返し列の数を見つける目的のために、UAは列のサイズを0で除算されないようにUAが指定した値に下げなければならない。このフロアは1px以下であることが提案されている。

ページ化されたメディアのような断片化されたコンテキストでは、ユーザーエージェントはこの計算を断片ごとに実行してもよい。

列数に対する使用値は、明示的な列の改行や制約された列の高さを考慮せずに計算されますが、実際の値はこれらを考慮に入れています。

## css\_syntax

### 例17

この例では、明示的な列の区切りにより、実際の列数が使用済みの列数よりも多くなっています。

```
div {  
  width: 40em;  
  columns: 20em;  
  column-gap: 0;  
}  
p {  
  break-after: column;  
}
```

```
<div>  
  <p>one  
  <p>two  
  <p>three  
</div>
```



One                    Two                    Three

figure8 計算されたカラム数は自動、使用されたカラム数は2、実際のカラム数は3です。

### 例18

実際のカラム数は、使用されているカラム数よりも少ない場合があります。この例を考えてみましょう。

```
div {  
  width: 80em;  
  height: 10em;  
  columns: 20em;
```

```
column-gap: 0;
column-fill: auto;
}
```

```
<div>foo</div>
```

計算された列数は自動、使用された列数は4、実際の列数は1です。

### 3.5. スタッキングコンテキスト

`multi-column`コンテナ内のすべてのカラムボックスは同じスタッキングコンテキストにあり、その内容の描画順序はCSS2.1で指定されている通りである。カラムボックスは新たなスタッキングコンテキストを確立しない。

## 4. 段間とルーラー(定規)

カラムギャップとカラムルールは、同じ`multicol`コンテナ内のカラム間に配置されます。カラムギャップとカラムルールの長さは、カラムの高さと同じです。カラムギャップはスペースを取ります。つまり、カラムギャップは、隣接するカラム（同じ`multicol`コンテナ内）のコンテンツを押しのけてしまいます。

カラム・ルールは、カラム・ギャップの中央に、マルチコル・コンテナの対向するコンテンツの端に終点を置いて描かれます。カラムルールはスペースを取りません。つまり、カラムルールの存在や太さが他の何かの配置を変えることはありません。カラムルールがそのギャップよりも広い場合、隣接するカラムボックスはルールと重なり、ルールはマルチコルコンテナのボックスの外にはみ出してしまる可能性があります。列規則は、マルチコルコンテナの境界線のすぐ上に描かれます。スクロール可能なマルチコルコンテナの場合、マルチコルコンテナのボーダーと背景は明らかにスクロールされませんが、ルールは列とともにスクロールする必要があることに注意してください。カラムルールが描かれるのは、両方のカラムが内容を持つ2つのカラムの間だけです。

### 4.1. `column-gap`プロパティ: 段間

`column-gap`プロパティは[CSS3-ALIGN]で定義されています。

## css\_syntax

複数列の書式設定のコンテキストでは、column-gapプロパティのnormalの値は1emである。これにより、初期値が使用されているときに列が確実に読み取れるようになります。カラム間にカラムルールがある場合は、ギャップの中央に表示されます。

## 4.2. column-rule-colorプロパティ: 列規則の色

名前	column-rule-color
値	色（列ルーラー（定規）の色を指定します。）
初期値	現在の色
適用対象	マルチカラーコンテナ
継承	なし
割合	該当なし
計算された値	計算された色
正準順序	文法ごと
アニメーションタイプ	計算値タイプによる

## 4.3. column-rule-styleプロパティ: ルーラーの形状

名称	コラムルールスタイル
値	line-style
初期値	なし
適用対象	multi-col containers
継承	なし
割合	該当なし
計算値	指定されたキーワード
正準順序	文法ごと
アニメーションの種類	discrete

- line-style none | hidden | dotted | dashed | solid | double | groove | ridge  
| inset | outset

`column-rule-style` プロパティは、要素の列間のルールのスタイルを設定する。  
`line-style` 値は、`collapsing border model` と同じように解釈されます。

`none` 値は、`column-rule-width` の計算値を 0 に強制します。

#### 4.4. column-rule-widthプロパティ: 列規則の幅

名前	column-rule-width
値	行幅
初期値	ミディアム
適用対象	multi-col containers
継承	なし
割合	該当なし
計算値	絶対長
正準順序	文法ごと
アニメーションタイプ	計算値タイプによる

このプロパティは、列間のルーラー（定規）の幅を設定します。負の値を指定することはできません。

#### 4.5. column-ruleプロパティ: 列ルーラー(定規)の短縮形

名前	column-rule
値	<code>column-rule-width</code> , <code>column-rule-style</code> , <code>column-rule-color</code>
初期値	個別のプロパティを参照してください。
該当するもの	個別の物件を見る
継承	個別のプロパティを参照
パーセンテージ	個別物件を参照
計算値	個々のプロパティを参照
アニメーションの種類: 個々のプロパティを参考してください。	
正準順序	文法ごと

## css\_syntax

このプロパティは、スタイルシート内の `column-rule-width`、`column-rule-style`、`column-rule-color` を同じ場所に設定するための略語です。省略された値は初期値に設定されます。

### 例19

この例では、列のルーラー（定規）と段間は同じ幅を持っています。したがって、それらは全く同じスペースを占めることになります。

```
body {  
    column-gap: 35px;  
    column-rule-width: 35px;  
    column-rule-style: solid;  
    column-rule-color: black;  
}
```

column-rule ←→

Following his own account, Lunardi's first act on finding himself fairly above the town was to fortify himself with some glasses of wine, and to devour

the leg of a chicken. He describes the city as a vast beehive, St. Paul's and other churches standing prominently; the streets shrunk to

lines, and all humanity apparently transfixed and watching him. A little later he is equally struck by the view of the open country, and his ecstasy is

column-gap ←→

列のルーラー（定規）と段間は同じ幅を持っています。

## 5. コラムの区切り

コンテンツが複数の列にレイアウトされている場合、ユーザエージェントは、列の区切りがどこに置かれるかを決定しなければならない。コンテンツを列に分割する問題は、コンテンツをページに分割する問題と似ており、CSS 2.1のセクション13.3.3 [CSS21]で記述されている。

列の改行をページの改行と同じプロパティで記述できるようにするために、`break-before`、`break-after`、`break-inside` の3つの新しいプロパティが導入されています。

## 5.1. the **break-before**, **break-after**, **break-inside** properties: 細分化のコントロールについて

`break-before`、`break-after`、`break-inside` は、`CSS3-BREAK` で定義されている。

## 6.1. column-span property: 段抜き

名前	column-span
値	<code>none</code> または <code>all</code>
初期値	なし
適用対象	<code>in-flow block-level elements</code>
継承	なし
割合	該当なし
計算値	指定されたキーワード
正準順序	文法ごと
アニメーションの種類	<code>discrete</code>

- `none` 要素が複数の列にまたがることはありません。
- `all` 要素は、`out-of-flow` で取り出され、同じブロック組版文脈の中で最も近い複数コラムの祖先のすべての列にまたがって広がります。要素の前に現れる通常フローの内容は、要素が現れる前の直前の複数コラム行のすべての列にまたがって自動的にバランスをとる。要素は独立した書式設定コンテキストを確立する。

『out-of-flow』とは、ボックスは、期待される位置と周囲のコンテンツとの相互作用から抽出され、親の書式設定コンテキストでのコンテンツの通常の流れの外に別のパラダイムを使用してレイアウトされている場合には、『out-of-flow』

』

## css\_syntax

となります。これは、ボックスがフロー・テイキング（『float』を使って）または絶対位置（『position』を使って）に配置されている場合に発生します。ボックスは、『out-of-flow』でなければ『in-flow』です。

注意：要素が新しい書式設定コンテキストを確立するかどうかは、要素がマルチコルの子孫であるかどうかには関係ありません。『column-span』がすべての場合は、常にそうなります。これは、複数コラムを削除した後の修正に対するデザインの堅牢性を高めるのに役立ちます。

1つ以上の列にまたがる要素はスパンニング要素と呼ばれ、それが作成するボックスはスパナと呼ばれます。

このスパナは、スパンニング要素が包含ブロックを確立するスパナ内の絶対位置にあるボックスの包含ブロックとなり、そうでない場合は包含ブロック・チェーンはマルチコル・コンテナに行きます。

スパンニング要素は、強制的なブレークを残してアウトオブフローで取り出されます。これは、スパンニング要素の描画順序 [CSS21] には影響しない。

### 例20

この例では、サンプル文書の第6文の後にh2要素が追加されています（すなわち、"the leg of a"という単語の後）。このスタイルが適用されます。

```
h2 { column-span: all; background: silver }
```

『column-span』を『all』にすることで、h2要素の前に表示されるコンテンツはすべてh2要素の前に表示されます。

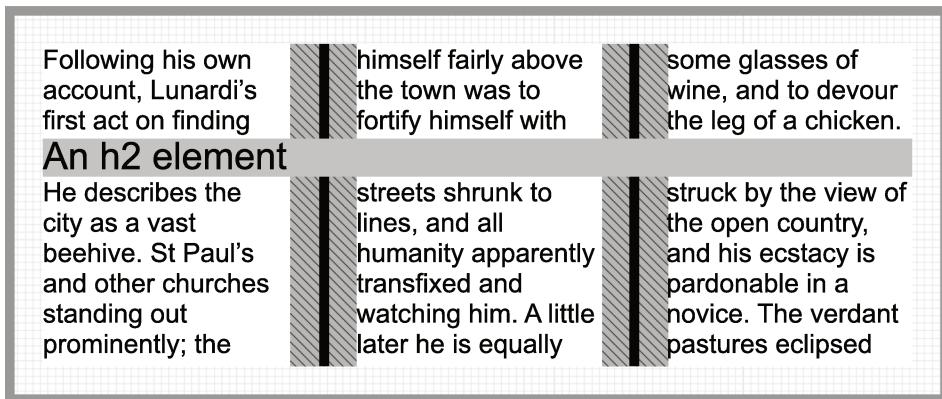


Figure10 『H2』要素を『column-span: all』で適用

スパンning要素は、同じ書式設定コンテキストの一部である限り、子孫の最初のレベルよりも下のレベルにすることができます。スパンの前のフラグメントが空の場合、何も特別なことは起こりません。

## 例21

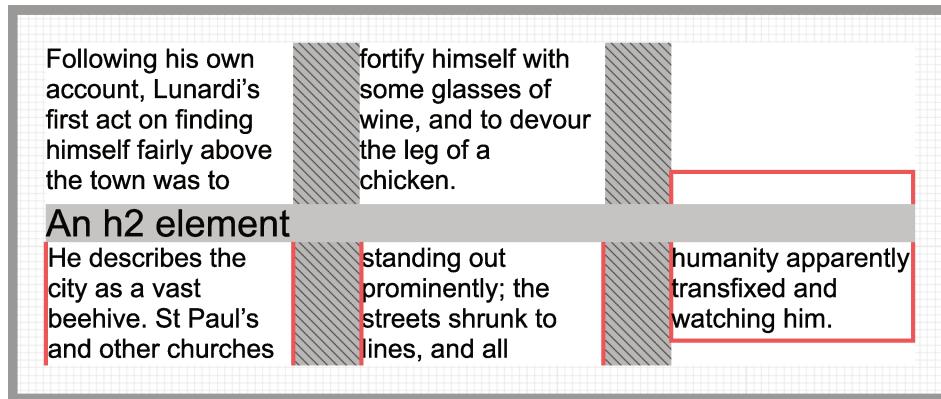
この例では、マルチコルコンテナは『article要素』です。この親の中には、本文とセクション要素があります。セクションには h2 の見出しが含まれており、これは3つの列すべてにまたがっています。 h2はセクションの最初の子です。つまり、このセクションのマージン、ボーダー（図では赤で示されています）、パディングは、空のフラグメントとしてh2の前に現れます。

```
<article>
  <p>...</p>
  <section>
    <h2>An h2 element</h2>
    <p>...</p>
  </section>
</article>
```

```
section {
  border: 2px solid red;
  margin-top: 65px;
  padding-top: 20px;
```

```
}
```

```
h2 {  
    column-span: all;  
    background: silver  
}
```



h2要素は`column-span: all`に設定されており、セクションは赤いボーダーとトップパディングとマージンを持っています。

スパニング要素は、それ以外の場合よりも多くのスペースを必要とします。スペースが限られている場合は、スパニング要素を入れるスペースを見つけることができない場合があります。このような場合、ユーザーエージェントは、このプロパティに何も指定されていないかのようにその要素を扱うことができます。