CSS3

MULTI-COLUMN LAYOUTS

The **CSS3 multi-column layout module** allows us to create multiple columns of text from one block level element.

We can use any of the **ten different properties**: column-count, column-width, column-gap, column-fill, column-rule-color, column-rule-style, column-rule-width, column-rule, column-span, columns

column-count

The **column-count property** specifies the number of columns an element should be divided into.

```
.element
{
    column-count: ;
}
```

The default value for column-count is **“auto”**. This means that the number of columns will be determined by other properties, such as the “column-width”.

```
.element
{
    -webkit-column-count: auto;
    -moz-column-count: auto;
    column-count: auto;
}
```

The value can also be specified as a **whole number**, which defines the number of columns that will be used to lay out the content.

```
.element
{
    -webkit-column-count: 2;
    -moz-column-count: 2;
    column-count: 2;
}
```

column-width

The **column-width property** sets the minimum desired column width. If column-count is not also set, then the browser will automatically make as many columns as fit in the available width.

```
.element
{
    column-width: ;
}
```

The default value for column-width is **“auto”**. This means that the column width will be determined by the browser.

```
.element
{
    -webkit-column-width: auto;
    -moz-column-width: auto;
    column-width: auto;
}
```

The value can also be specified as a **length value**, which defines the width of all columns.

```
.element
{
    -webkit-column-width: 150px;
    -moz-column-width: 150px;
    column-width: 150px;
}
```

column-gap

The **column-gap property** specifies the gap between the columns.

```
.element
{
    column-gap: ;
}
```

The default value for column-gap is **“normal”**. Specifies a normal gap between the columns.

```
.element
{
    -webkit-column-gap: normal;
    -moz-column-gap: normal;
    column-gap: normal;
}
```

The value can also be specified as a **length value**, which defines the gap between columns.

```
.element
{
    -webkit-column-gap: 1em;
    -moz-column-gap: 1em;
    column-gap: 1em;
}
```

column-fill

The **column-fill property** specifies how to fill the columns with text and whether the columns of text should be balanced or not.

```
.element
{
    column-fill: ;
}
```

The default value for column-fill is **“balanced”**. This means that browsers will try to minimise the differences between each column.

```
.element
{
    -webkit-column-fill: balanced;
    -moz-column-fill: balanced;
    column-fill: balanced;
}
```

The value can also be specified as **“auto”**, which means that columns will be filled sequentially, and they may have different lengths.

```
.element
{
    -webkit-column-fill: auto;
    -moz-column-fill: auto;
    column-fill: auto;
}
```

column-rule-color

The **column-rule-color property** specifies specifies the color of the rule between columns.

```
.element
{
    column-rule-color: ;
}
```

The value must be specified as a **recognised W3C color value** - such as color keyword, hexadecimal, RGB, RGBA, HSL, HSLA etc.

```
.element
{
    -webkit-column-rule-color: blue;
    -moz-column-rule-color: blue;
    column-rule-color: blue;
}
```

column-rule-style

The **column-rule-style property** specifies the style of the rule between columns.

```
.element
{
    column-rule-style: ;
}
```

The default value for column-fill is **“none”**. This means that no rule will be displayed.

```
.element
{
    -webkit-column-rule-style: none;
    -moz-column-rule-style: none;
    column-rule-style: none;
}
```

The value can be specified as **“hidden”**, which means that the rule will be hidden.

```
.element
{
    -webkit-column-rule-style: hidden;
    -moz-column-rule-style: hidden;
    column-rule-style: hidden;
}
```

The value can be specified as **“dotted”**, which will render a dotted rule.

```
.element
{
    -webkit-column-rule-style: dotted;
    -moz-column-rule-style: dotted;
    column-rule-style: dotted;
}
```

The value can be specified as **“dashed”**, which will render a dashed rule.

```
.element
{
    -webkit-column-rule-style: dashed;
    -moz-column-rule-style: dashed;
    column-rule-style: dashed;
}
```

The value can be specified as **“solid”**, which will render a solid rule.

```
.element
{
    -webkit-column-rule-style: solid;
    -moz-column-rule-style: solid;
    column-rule-style: solid;
}
```

The value can be specified as **“double”**, which will render a double rule.

```
.element
{
    -webkit-column-rule-style: double;
    -moz-column-rule-style: double;
    column-rule-style: double;
}
```

The value can be specified as **“groove”**, which will render 3D grooved rule. The effect depends on the width and color values.

```
.element
{
    -webkit-column-rule-style: groove;
    -moz-column-rule-style: groove;
    column-rule-style: groove;
}
```

The value can be specified as **“ridge”**, which will render a 3D ridged rule. The effect depends on the width and color values.

```
.element
{
    -webkit-column-rule-style: ridge;
    -moz-column-rule-style: ridge;
    column-rule-style: ridge;
}
```

The value can be specified as **“inset”**, which will render a 3D inset rule. The effect depends on the width and color values.

```
.element
{
    -webkit-column-rule-style: inset;
    -moz-column-rule-style: inset;
    column-rule-style: inset;
}
```

The value can be specified as **“outset”**, which will render a 3D outset rule. The effect depends on the width and color values.

```
.element
{
    -webkit-column-rule-style: outset;
    -moz-column-rule-style: outset;
    column-rule-style: outset;
}
```

column-rule-width

The **column-rule-width property** specifies the width of the rule between columns.

```
.element
{
    column-rule-width: ;
}
```

The default value for column-rule-width is **“medium”**, which creates a medium thickness rule.

```
.element
{
    -webkit-column-rule-width: medium;
    -moz-column-rule-width: medium;
    column-rule-width: medium;
}
```

The value can be specified as **“thin”**, which creates a thin rule.

```
.element
{
    -webkit-column-rule-width: thin;
    -moz-column-rule-width: thin;
    column-rule-width: thin;
}
```

The value can be specified as **“thick”**, which creates a thick rule.

```
.element
{
    -webkit-column-rule-width: thick;
    -moz-column-rule-width: thick;
    column-rule-width: thick;
}
```

The value can be specified as a **length value**, which defines the width of the rule.

```
.element
{
    -webkit-column-rule-width: 2px;
    -moz-column-rule-width: 2px;
    column-rule-width: 2px;
}
```

column-rule

The **column-rule property** is a shorthand property for setting all the column-rule-* properties.

```
.element
{
    column-rule: ;
}
```

The column-rule property sets the **width, style, and color** of the rule between columns.

```
.element
{
    -webkit-column-rule: 1px solid red;
    -moz-column-rule: 1px solid red;
    column-rule: 1px solid red;
}
```

column-span

The **column-span property** specifies how many columns an element should span across. This is often used when you want a heading to span across multiple columns

This is a heading spanning multiple columns

Lorem ipsum dolor sit amet consect etuer adipi scing elit sed diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat

```
.element
{
    column-span: ;
}
```

The default value for column-count is “1”. This means that the element will only span one column.

```
.element
{
    -webkit-column-span: 1;
    -moz-column-span: 1;
    column-span: 1;
}
```

The value can be specified as **whole “all”**, which spans the element across all columns.

```
.element
{
    -webkit-column-span: all;
    -moz-column-span: all;
    column-span: all;
}
```

columns

The **columns property** is a shorthand property for setting column-width and column-count.

```
.element
{
    columns: ;
}
```

The default value for columns is **“auto”**. This sets the column-width to “auto” and the column-count to “auto”.

```
.element
{
    -webkit-columns: auto;
    -moz-columns: auto;
    columns: auto;
}
```

The value can also be specified as a **column-width value** (a length value) only which defines the width of columns.

```
.element
{
    -webkit-columns: 100px;
    -moz-columns: 100px;
    columns: 100px;
}
```

The value can also be specified as a **column-count value** (a number value) only which defines the number of columns.

```
.element
{
    -webkit-columns: 2;
    -moz-columns: 2;
    columns: 2;
}
```

The value can also be specified with a **column-count value and a column width value**.

```
.element
{
    -webkit-columns: 2 100px;
    -moz-columns: 2 100px;
    columns: 2 100px;
}
```

Browser support

The **CSS3 multi-column layout properties** are not well supported across modern browsers and require the -moz and -webkit prefixes.

CSS3 Multiple column layout - CR

Method of flowing information in multiple columns

Global 19.89% + 75.53% = 95.42%

unprefixed: 19.89%

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								4.1	
8			43					4.3	
9		40	44					4.4	
10		41	45	8		8.4		4.4.4	
11	12	42	46	9	32	9.1	8	44	46
	13	43	47		33				
		44	48		34				
		45	49						

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley