CSS3

by Mike McGrath

Chapter 3: Styling Boxes

This chapter demonstrates how to style the boxes that contain web page content.

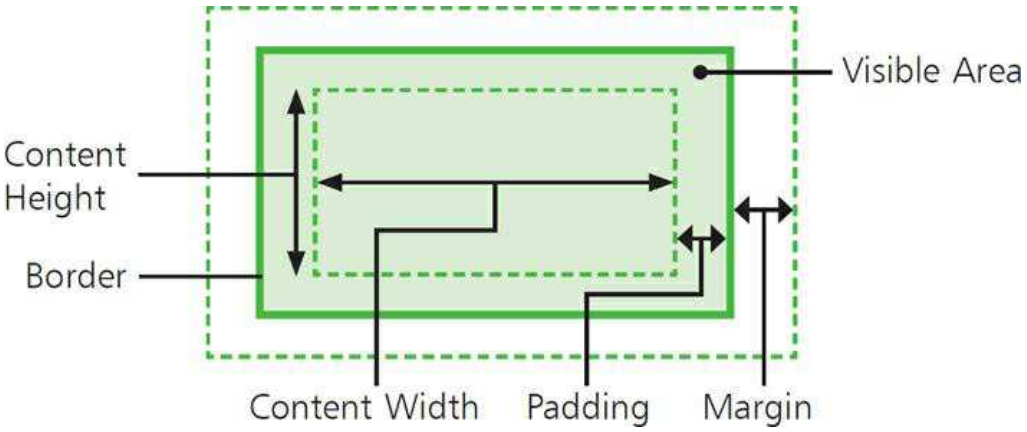
Recognizing Content Boxes

Content on a web page is displayed in a number of invisible rectangular boxes that are generated by the browser. These content boxes may be either "block level" or "inline".

Block-level content boxes normally have line breaks before and after the box, such as paragraph, heading and "div" elements.

Inline content boxes, on the other hand, do not add line breaks but are simply created within lines of text, such as span, emphasis, and hyperlink elements.

Each block-level content box comprises a core content area surrounded by optional areas for padding, border, and margins:



Style rules can specify values for the padding, border, and margin properties to control the appearance of content boxes. These all apply to block-level boxes but some properties, such as width and height, do not apply to inline boxes. Additionally, the margin and padding properties of inline boxes only apply to either side of the content – not the areas above and below the content.

Hot Tip Block-level content boxes are by default stacked on the page, one below another, whereas inline content boxes appear within a block-level box, one behind another.

When the padding, border, and margin properties all have a zero width the content box will be the same size as the content area, determined by the dimensions of the content.

Any padding, border, and margin areas that have a non-zero width are added outside the content area, so the content size remains the same but the box size increases.

The padding property extends the area around the content and inherits the background color of the content area. The border property extends the area around the content and any padding. The margin property extends the area around the content, any padding, and any border, with a transparent background.

1. Create an HTML document with four simple paragraphs, three with assigned **class** attribute values

```
<p>Content Box</p>
<p class="pad">Content Box - Padded</p>
<p class="pad bdr">Content Box - Padded + Border</p>
<p class="pad bdr mgn">Content Box -
                       Padded + Border + Margin</p>
```

box.html

2. Save the HTML document then create a linked style sheet with a style rule that sets the width and background color of the core content area

```
p { background : yellow; width : 20em }
```

box.css

3. Next add a style rule to add some padding

```
p.pad { padding : 1em }
```

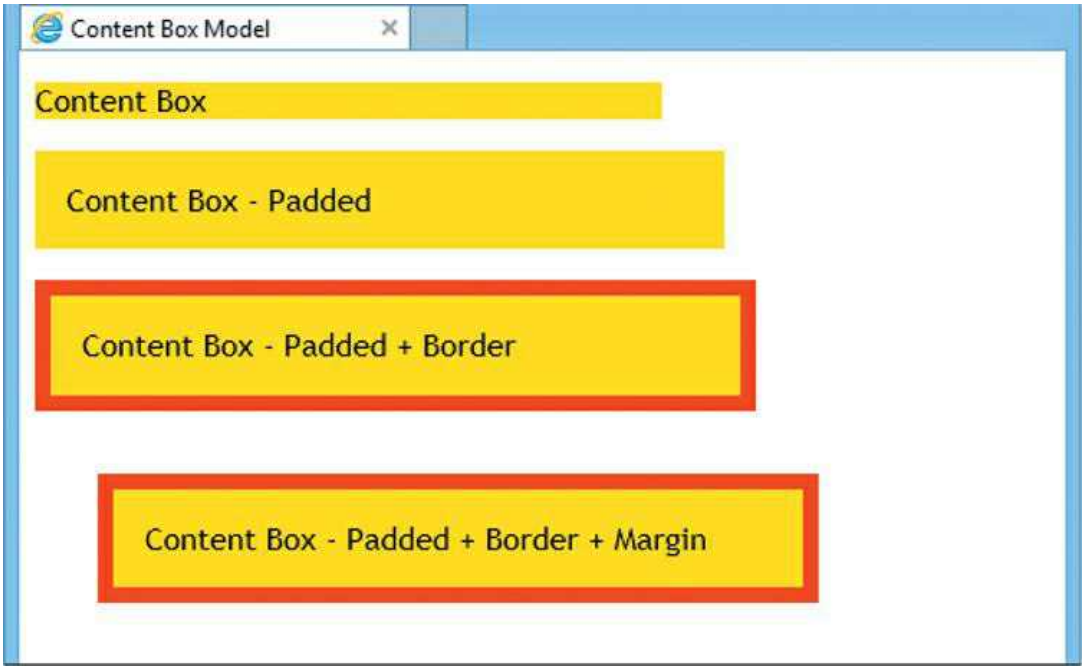
4. Now add a style rule to add a border

```
p.bdr { border : 0.5em red solid }
```

5. Finally add a style rule to add a margin

```
p.mgn { margin : 2em }
```

6. Save the style sheet alongside the HTML document then open the web page in a browser to see the content boxes with added padding, borders, and margin



Changing Display Formats

A web page relies upon the creation of block-level content boxes, to determine its general layout, and the creation of inline content boxes within the blocks to determine its precise layout.

This places great emphasis on whether an element is considered block-level or inline to determine the display format. Generally the default display format for each element will be the most appropriate. For example, it's generally desirable to display list items on individual lines in a block-level list.

Beware The run-in and inline-block styles are not universally supported, so it may be best to avoid them. They are included here for completeness.

The display format of an element can also be explicitly determined by a style rule that assigns the **block** or **inline** keywords to that element's **display** property. This means that content can be displayed in a different format without changing the HTML tags. For example, list items can be displayed on a single line with a **display : inline** declaration.

Where two block-level elements follow each other the first can be made to be displayed as if it was an inline box within the beginning of the second block by assigning its **display** property the **run-in** keyword. For example, applying a run-in style rule to a heading that is followed by a paragraph will display the heading apparently within the start of the paragraph block.

Additionally an inline content-box can have its **display** property assigned an **inline-block** value to allow it to be displayed somewhat like a block-level content box. The inline-block still appears inline, as usual, but unlike regular inline content boxes its **width** and **height** properties can be assigned values to control its size.

1. Create an HTML document containing two unordered list blocks, with one assigned a **class** attribute value

```
<ul>
<li>Item 1 </li><li>Item 2 </li><li>Item 3 </li></ul>
<ul class="line">
<li>Item 1 </li><li>Item 2 </li><li>Item 3 </li></ul>
```

2. Next add two heading blocks, each followed by a paragraph block, with one assigned a **class** attribute value

```
<h2>Heading Block</h2><p>Paragraph Block</p>
<h2 class="run">Heading Block</h2>
<p>Paragraph Block</p>
```

3. Next add a paragraph block containing three inline hyperlink content boxes

```
<p>
<a href="1.html">Page 1</a>
<a href="2.html">Page 2</a>
<a href="3.html">Page 3</a>
</p>
```

4. Save the HTML document then create a linked style sheet with a rule to display one list's items inline

```
ul.line li { display : inline; background : aqua }
```

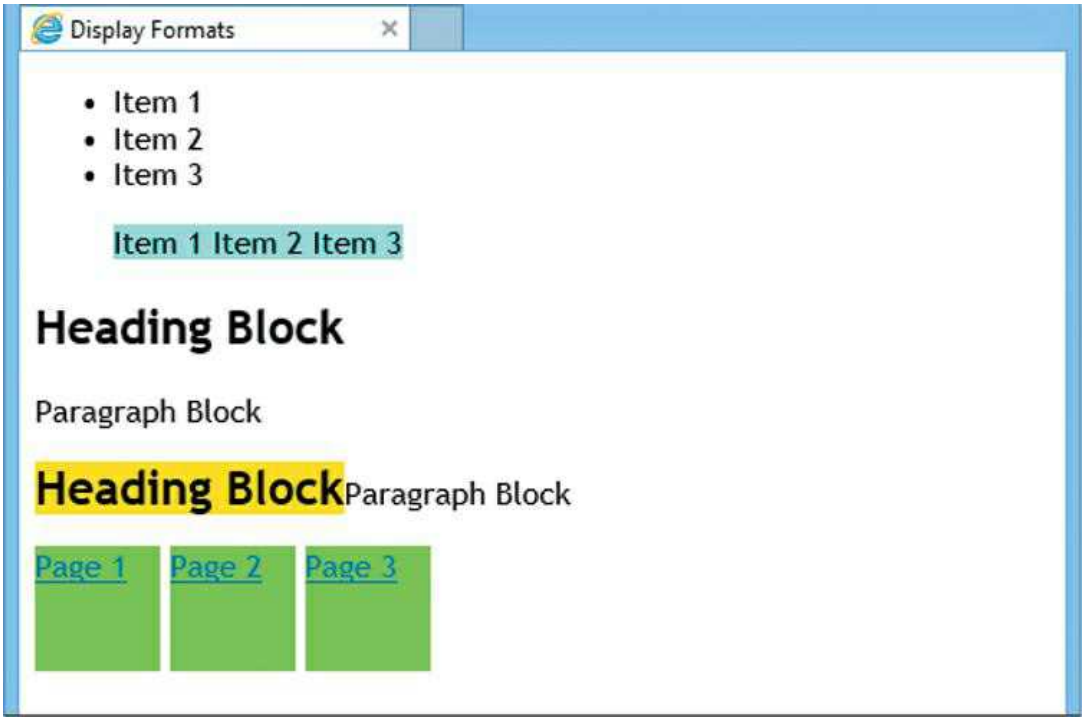
5. Add another style rule to run one heading block into the following paragraph block

```
h2.run { display : run-in; background : yellow }
```

6. Then add a style rule to change the inline hyperlink content boxes into inline blocks so their height and width can be adjusted

```
p a { display : inline-block;
          width : 4em; height : 4em; background : lime }
```

7. Save the style sheet alongside the HTML document then open the web page in a browser to see the content displayed in default and non-default display formats



Don't Forget Assigning a non-default display type to an element only changes the way it gets displayed – in the document tree inline elements are always descendants of block-level elements.

Sizing the Content Area

When assigning any non-zero value to a property the declaration must include a two-letter unit name to specify which unit of measurement to apply. The CSS specification provides the following unit names representing real world measurement:

Unit :	Description :	Example :
in(inches)	American standard unit of length measurement	**div { width : 1in }**
cm(centimeters)	Metric unit of length where 2.54 centimeters is equivalent to 1 inch	**div { height : 2.54cm }**
mm(millimeters)	Metric unit of length (one tenth of one centimeter) where 25.4 millimeters is equivalent to 1 inch	**div { height : 25.4mm }**
pt(points)	Typographical unit of font height where 72 points is equivalent to 1 inch	**div { font-size : 72pt }**
pc(picas)	Typographical unit of font height where 6 picas is equivalent to 1 inch	**div { font-size : 6pc }**

Don't Forget Zero values can be assigned using just a "0" number – without specifying a unit name.

The CSS specification also provides the following unit names representing relative values according to the viewing device:

Unit :	Description :	Example :
em(font size)	Abstract typographical unit of font size where 1em is equivalent to the height of a given font	**div { font-size : 14pt }** (1em = 14pt)
ex(font size)	Abstract typographical unit of font size where 1ex is equivalent to the height of lowercase "x" in a font (often 50% of 1em)	**div { font-size : 14pt }** (1ex = 7pt)
px(pixels)	Abstract unit representing the dots on a computer monitor where there are 1024 pixels on each line when the monitor resolution is 1024×768	**div { height : 100px };**

Hot Tip A percentage value can also specify a relative size – where a value of 50% makes the target element half the size of its containing element.

1. Create an HTML document containing 4 "div" elements

```
<div id="absolute">3in x ½in</div>
<div id="container">400px x 200px
 <div id="percent">50% x 50%</div>
```

```
 <div id="relative">20em x 2em</div>
</div>
```

size.html

2. Save the HTML document then create a linked style sheet with a rule to size an element by absolute units

```
div#absolute { width : 3in; height : 0.5in;
                        background : red; color : yellow }
```

size.css

3. Next add a rule to size an element by monitor resolution

```
div#container { width : 400px; height : 200px;
                         background : blue; color : aqua }
```

4. Now add a rule to size an element by percentage

```
div#percent { width : 50%; height : 50%;
                       background : green; color : yellow }
```

5. Then add a rule to size an element relative to font height

```
div#relative { width : 20em; height : 2em;
                         background : yellow; color : blue }
```

6. Save the style sheet alongside the HTML document then open the web page in a browser to see the element sizes

Hot Tip It is often considered good practice to use **em** units wherever possible for maximum flexibility.

Controlling Borders

Each content box can have a border comprising **border-width, border-color**, and **border-style** properties. A value can be

specified to each of these individual properties to apply a uniform border to all four sides of the content box, or a space-separated list of values can be specified to apply different borders to each side:

- When two values are listed the first is applied to the top and bottom borders
- When three values are listed the first is applied to the top border, the second is applied to the left and right borders, and the third is applied to the bottom border
- When four values are listed they are applied clockwise to the top, right, bottom, and left borders

The default **border-width** value is **medium** (a computed value), and the default **border-color** value is inherited from the element's **color** property, but the default **border-style** is **none**. This means that the border will not be visible until a value is assigned – possible **border-style** values are **solid, double, dotted, dashed, groove, ridge, inset, outset, hidden**, and **none.**

Rather than creating separate style rules for the **border-width, border-color**, and **border-style** properties it is simpler to use the CSS shorthand technique that specifies a value for each of these three properties to a **border** property as a space-separated list. This uniformly styles each side of the content box with a border of the specified width, color, and style. For example, a style rule declaration of **border : 0.5in red dotted** would apply a half-inch wide red dotted border to all four sides of the content box.

Hot Tip The **outset** border style can be used to create the appearance of a raised button – and the **inset** border style can be used to create the appearance of a depressed button.

If it is desirable to have different styles, the borders on each side of a content box can be individually styled by creating rules for the element's **border-top, border-right, border-bottom**, and **border-left** properties. The CSS shorthand technique can also be used with these properties to specify a width, color, and style for the individual side as a space-separated list. For example, a style rule declaration of **border-bottom : 0.5in red dotted** would apply a half-inch wide red dotted border to just the bottom side of the content box.

1. Create an HTML document containing four paragraphs

```
<p id="p1">Solid - Inherit - Medium</p>
<p id="p2">Top: Dotted - Orange - 0.2em
          <br>Bottom: Dashed - Green - 0.2em</p>
<p id="p3">Ridge Double - Blue - 1em</p>
<p id="p4">Outset - Yellow - 1em</p>
```

size.html

2. Save the HTML document then create a linked style sheet with a rule to add a border that inherits a color

```
#p1 { color : red; border : solid }
```

size.css

3. Next add a rule with shorthand declarations to create a border above and below the content area only

```
#p2 { border-top : 0.5em orange dotted;
      border-bottom : 0.5em green dashed }
```

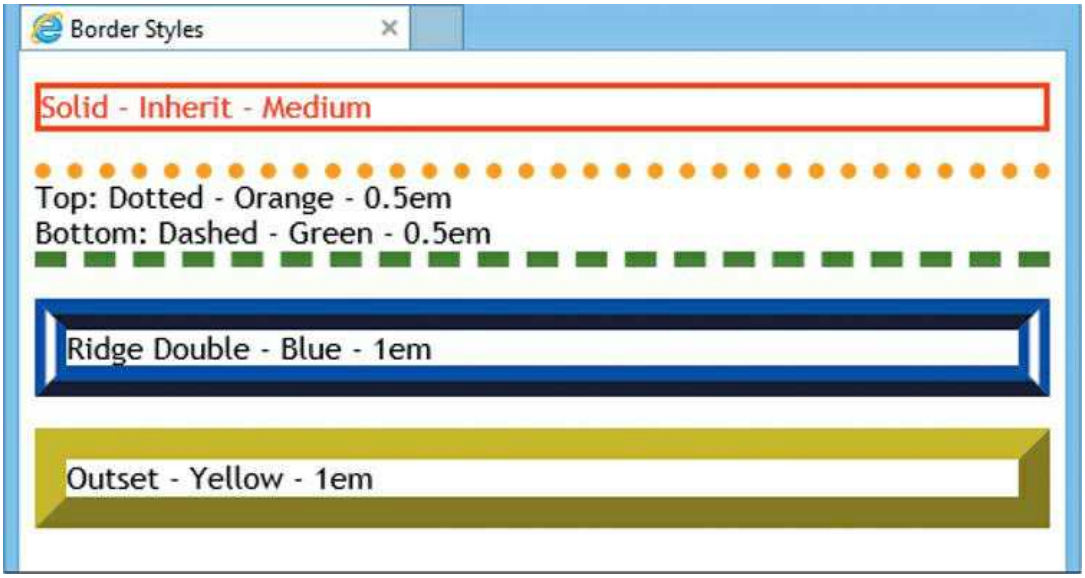
4. Now add a rule creating a border from separate properties

```
#p3 { border-style : ridge double;
          border-width : 1em;
          border-color : blue }
```

5. Then add a rule creating a border on all four sides using the recommended CSS shorthand technique

```
#p4 { border : 1em yellow outset }
```

6. Save the style sheet alongside the HTML document then open the web page in a browser to see the borders



Hot Tip Notice how the browser miters the borders diagonally where they meet – this offers some creative possibilities.

Adding Padding

Each content box can have "padding" space added around the core content area by a style rule assigning a value to the **padding** property. A single value can be specified to apply a uniform padding width to all four sides of the content area, or a space-separated list of values can be specified to apply different padding widths to each side:

- When two values are listed the first is applied to the top and bottom sides and the second is applied to the left and right
- When three values are listed the first is applied to the top side, the second is applied to the left and right sides, and the third is applied to the bottom side
- When four values are listed they are applied clockwise to the top, right, bottom, and left sides

Beware Setting padding as a percentage may produce undesirable results when the user resizes the browser window – specify unit values to avoid this.

The padding area surrounds the core content area and extends to the outer edges of the border area if a border is specified – right up to the beginning of the margin area. The element's background fills the core content area and the padding area, so that any specified background color gets automatically applied to both the core content area and the padding area.

The **padding** property can be specified as a unit value or as a percentage value. When a percentage is specified the padding width is calculated using the width and height of the containing element and the padding area size will be adjusted if the size of the containing element gets changed.

Hot Tip The padding width for each side can be set using the CSS **padding** shorthand by setting sides requiring no padding to zero – always use the shorthand.

Typically a padding area is specified when adding a border to an element to increase the space between the content and the border.

If it is desirable to have different padding widths on each side of a content box the padding can be individually styled by creating rules for the element's **padding-top, padding-right, padding-bottom**, and **padding-left** properties. For example, style rule declarations of **padding-top : 0.5in; padding-bottom : 0.5in** would apply a half-inch padding area to top and bottom sides. Alternatively, the same result can be achieved using the CSS shorthand with a declaration of **padding : 0.5in 0**

0.5in 0.

1. Create an HTML document containing three paragraphs that each include a span element and are separated by horizontal ruled lines

```
<p>Horizontally
       <span id="pad-h">Padded</span> Content.</p>
<hr>
<p>Vertically
        <span id="pad-v">Padded</span> Content.</p>
<hr>
<p>Horizontally and Vertically
      <span id="pad-hv">Padded</span> Content.</p>
```

padding.html

2. Save the HTML document then create a linked style sheet with rules to color the paragraph and span elements

```
p { background : aqua }
span { background : yellow; border : 0.3em red dashed }
```

padding.css

3. Next add a style rule to add padding to the left and right sides of the first span content box

```
span#pad-h { padding : 0 3em 0 3em }
```

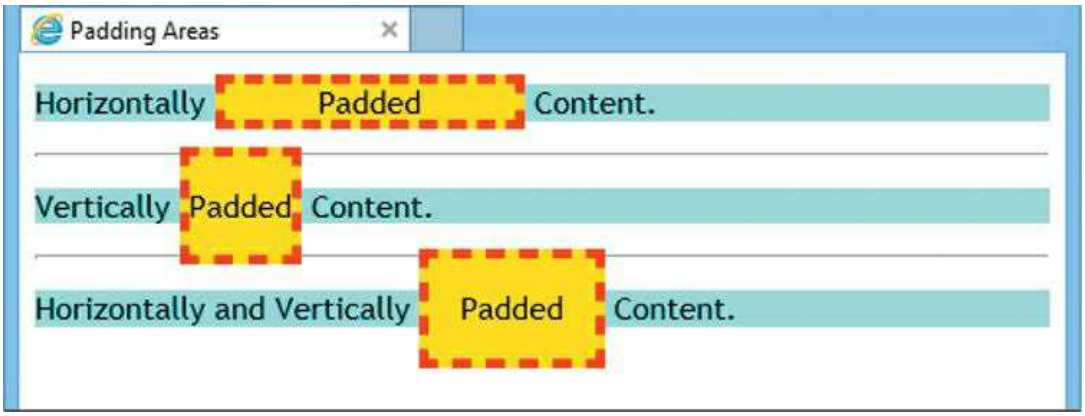
4. Now add a style rule to add padding to the top and bottom sides of the second span content box

```
span#pad-v { padding : 1em 0 1em 0 }
```

5. Then add a style rule to add uniform padding to all sides of the third span element

```
span#pad-hv { padding : 1em }
```

6. Save the style sheet alongside the HTML file then open the web page in a browser to see the added padding



Don't Forget Notice how the background color fills the content area and padding area – extending up to the outer edge of the border area.

Setting Margins

Each content box can have outer "margin" space added around the entire content, padding, and border areas by a style rule assigning a value to the **margin** property. A single value can be specified to apply a uniform margin width to all four sides of the content box, or a space-separated list of values can be specified to apply different margin widths to each side:

- When two values are listed the first is applied to the top and bottom sides and the second is applied to the left and right
- When three values are listed the first is applied to the top side, the second is applied to the left and right sides, and the third is applied to the bottom side
- When four values are listed they are applied clockwise to the top, right, bottom, and left sides

Hot Tip The ability to automatically compute the margin size is essential for centering content – as described on page 58.

The **margin** property has a default value of zero but in reality the browser applies its own intrinsic default values to allow spacing between elements. For example, heading elements always allow a margin area before a following paragraph element.

The **margin** property can be specified as a unit value, or as a percentage value, or with the **auto** keyword to have the browser compute a suitable margin.

Margins do not inherit any background and are always transparent – they merely separate elements by a specified distance.

Hot Tip The margin width for each side can be set using the CSS **margin** shorthand by setting sides requiring no margin to zero – always use the shorthand.

If it is desirable to have different margin widths on each side of a content box the margin can be individually styled by creating rules for the element's **margin-top, margin-right, margin-bottom**, and **margin-left** properties. For example, style rule declarations of **margin-top : 0.5in; margin-bottom : 0.5in** would apply a half-inch margin area to top and bottom sides. Alternatively, the same result can be achieved using the CSS shorthand with a declaration of **margin : 0.5in 0 0.5in 0.**

1. Create an HTML document containing three "div" elements that each contain a heading element and a paragraph element

```
<div><h2>Heading - Default Margin</h2>
<p>Paragraph - Default Margin</p></div>
<div><h2 class="zero">Heading - No Margin</h2>
<p class="zero">Paragraph - No Margin</p></div>
<div><h2 class="left">Heading - Left Margin</h2>
<p class="left">Paragraph - Left Margin</p></div>
```

2. Save the HTML document then create a linked style sheet with rules to color the elements

```
div { border : 1px solid black }
h2 { background : lime }
p { background : orange }
```

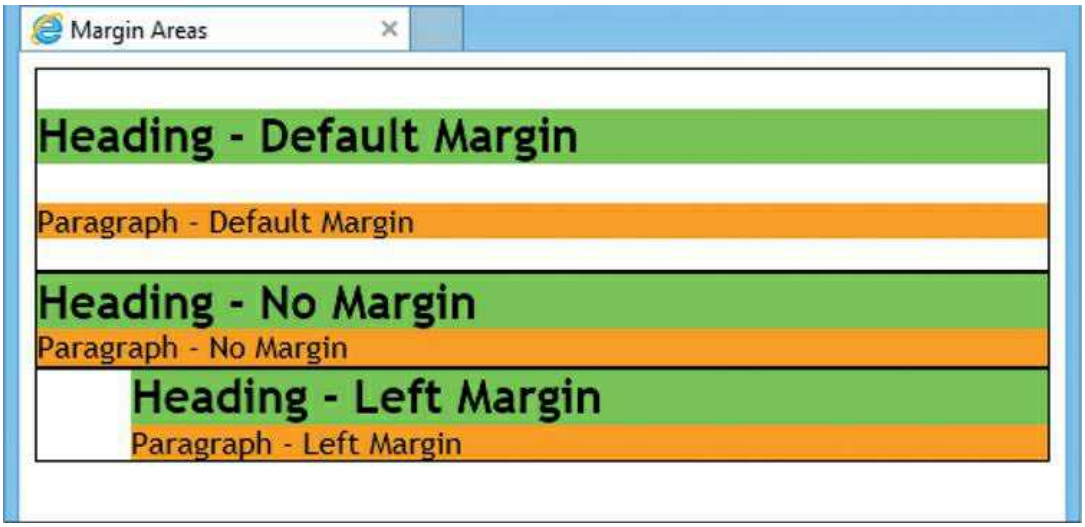
3. Next add a style rule to remove all margins from the second heading and paragraph element content boxes

```
.zero { margin : 0 }
```

4. Now add a style rule to add a left margin only to the third heading and paragraph element content boxes

```
.left { margin : 0 0 0 0.5in }
```

5. Save the style sheet alongside the HTML file then open the web page in a browser to see the margin styles



Painting Colors

The listed examples so far in this book have employed color keywords to represent the 17 CSS "standard" colors as below:

While the standard named colors are convenient to use they provide only a limited choice. Fortunately CSS allows authors to create their own custom colors.

Each color can be defined by the amount of red, green, and blue that needs to be mixed together to form that particular color. For example, the standard olive color comprises 50% red, 50% green and 0% blue. This can be expressed in RGB notation form as **rgb(50%, 50%, 0%)** for assignment to a CSS property like this:

```
h1 { color : rgb( 50%, 50%, 0% ) }      /* Olive headings */
```

Specifying different amounts of red, green, and blue, in this way the author can assign to a property any custom color desired.

As an alternative to specifying each color component as a percentage the author may choose an integer value in the range 0-255 for each component. In this format the RGB notation for the standard olive color looks like this:

```
h1 { color : rgb( 128, 128, 0 ) }       /* Olive headings */
```

Don't Forget Web browsers often recognize many more color names (often up to 140) but those shown here are the standard color names defined in the CSS specifications.

A second alternative to specifying each color component as a percentage allows the author to choose a hexadecimal value in the range 00-FF for each component. In this format the assignment for the standard olive color looks like this:

```
h1 { color : #808000 }       /* Olive headings */
```

The table opposite lists the ways in which each standard color can be assigned to a property using these various methods.

Color :	Percentage :	Integer :	Hex :

Color :	Percentage :	Integer :	Hex :
Black	rgb(0%,0%,0%)	rgb(0,0,0)	#000000
Red	rgb(100%,0%,0%)	rgb(255,0,0)	#FF0000
Orange	rgb(100%,65%,0%)	rgb(255,165,0)	#FFA500
Yellow	rgb(100%,100%,0%)	rgb(255,255,0)	#FFFF00
Fuchsia	rgb(100%,0%,100%)	rgb(255,0,255)	#FF00FF
Lime	rgb(0%,100%,0%)	rgb(0,255,0)	#00FF00
Aqua	rgb(0%,100%,100%)	rgb(0,255,255)	#00FFFF
Blue	rgb(0%,0%,100%)	rgb(0,0,255)	#0000FF
White	rgb(100%,100%,100%)	rgb(255,255,255)	#FFFFFF
Maroon	rgb(50%,0%,0%)	rgb(128,0,0)	#800000
Olive	rgb(50%,50%,0%)	rgb(128,128,0)	#808000
Purple	rgb(50%,0%,50%)	rgb(128,0,128)	#800080
Green	rgb(0%,50%,0%)	rgb(0,128,0)	#008000
Teal	rgb(0%,50%,50%)	rgb(0,128,128)	#008080
Navy	rgb(0%,0%,50%)	rgb(0,0,128)	#000080
Gray	rgb(50%,50%,50%)	rgb(128,128,128)	#808080
Silver	rgb(75%,75%,75%)	rgb(192,192,192)	#C0C0C0

Hexadecimal color values comprising three pairs of matching digits can alternatively be expressed using CSS shorthand notation that represents each pair of digits as a single digit. For example, the hexadecimal value for the standard red color is **#FF0000** but can be specified in shorthand as **#F00**. Similarly black **#000000** can be written as **#000** and white **#FFFFFF** as **#FFF.**

Hot Tip In hex notation, you can specify "web safe" consistent colors by using only digits that are multiples of 3 for the red, green and blue.

The author is free to choose which color notation method to use according to their preference. Declarations in the style rules below show how each method might be used to specify a custom color:

```
h1 { color : rgb( 85%, 15%, 0% ) }      /* A custom red */
h2 { color : rgb( 0, 192, 12 ) }        /* A custom green */
h3 { color : #042BDF }                  /* A custom blue */
h4 { color : #DE2 }       /* A custom yellow (#DDEE22) */
```

Repeating Backgrounds

Each content box can have a background color specified by assigning a valid color value to its **background** property. Alternatively an image can be specified for the background by assigning a ***url(filename)*** value, where ***filename*** is the path to an image file – this should not be enclosed within quotes.

The **background-color** and **background-image** properties can be used to specify both a background color and a background image. Any specified background image will normally be positioned at the top left corner of the content box. The default behavior of the browser's **background-repeat** property is set to **repeat** so the image will be tiled, row-by-row, across the content box.

Hot Tip Use the **background** property shorthand rather than the various individual properties – to keep the style sheet more concise.

The tiling behavior can be modified by assigning different values to the **background-repeat** property where values of **repeat-x** restricts the tiling pattern to one horizontal row and **repeat-y** restricts the tiling pattern to one vertical column. Tiling can also be prevented by assigning a **no-repeat** value so that a single copy of the image appears at the top left corner of the content box.

Background images are placed on a layer above the background's color layer so specifying an image with transparent areas will allow the background color to shine through.

Using CSS shorthand the color, image, and repeat values can simply be assigned as a space-separated list to the **background** property. These may appear in any order and default values are assumed for any value not specified in the list. For example, the **background-repeat** property is assumed to have a **repeat** value unless another value is specified.

1. Create an HTML document with four paragraphs that each contain a span element

```
<p><span>Repeat - Default</span></p>

<p id="x"><span>Repeat-X</span></p>

<p id="y"><span>Repeat-Y</span></p>

<p id="no"><span>No Repeat</span></p>
```

2. Save the HTML document then create a linked style sheet containing a rule to style the span elements

```
span { background : yellow; margin : 0 0 0 10em }
```

3. Next add a style rule to specify the background color, background image, and height of each paragraph

```
p { background : url(tile.png) fuchsia repeat;
    height : 3.5em }
```

4. Now add a style rule to modify the tiling behavior of the second paragraph to one row

```
p#x { background-repeat : repeat-x }
```

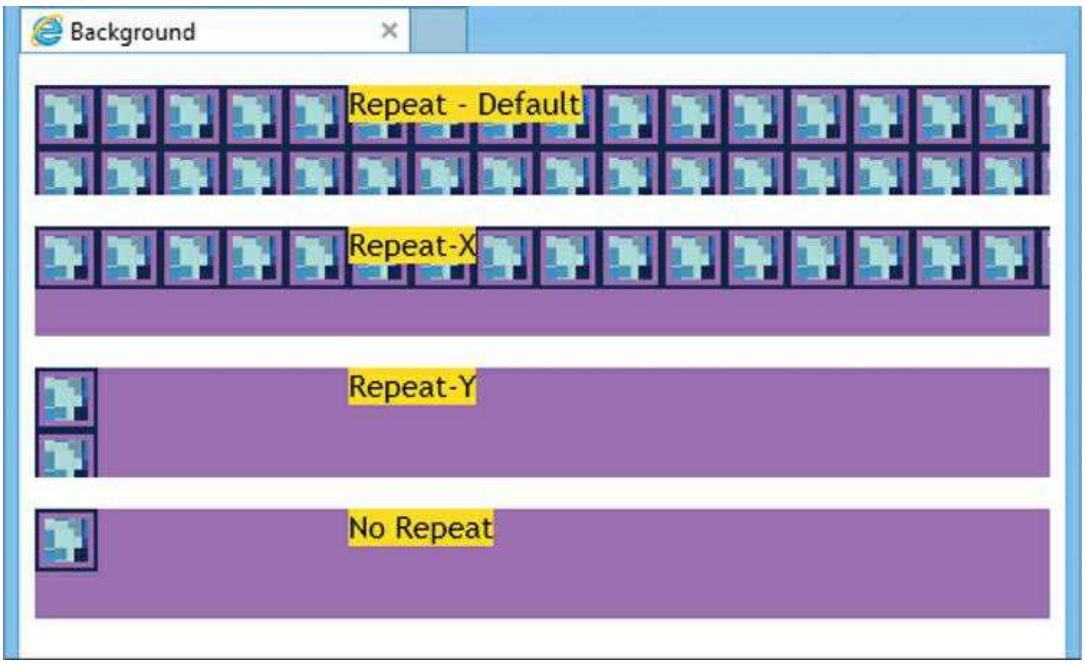
5. Then add a style rule to modify the tiling behavior of the third paragraph to one column

```
p#y { background-repeat : repeat-y }
```

6. Finally add a style rule to modify the tiling behavior of the fourth paragraph to just one copy of the image

```
p#no { background-repeat : no-repeat }
```

7. Save the style sheet alongside the HTML document then open the web page in a browser to see the background color and image applied



Don't Forget The **repeat** value could be omitted from the rule in step 3 – as that is the default value.

Positioning Backgrounds

A background image is, by default, positioned at the top left corner of the content box but this can be modified by specifying a different value to the **background-position** property. This can accept keywords of either **left, center, right, top**, and **bottom**. Two of these keywords may be used to specify the position, separated by a space, such as **top center**. Alternatively just one of these keywords can be specified and the second value will be assumed to be **center.**

The **background-position** property can also be assigned percentage values to specify the position with greater precision, taken from the dimensions of the containing element. The keywords have the equivalent to these percentages:

X-Axis :			Y-Axis :
			top (0%)
left(0%)	center(50%)	right (100%)	center (50%)
			bottom(100%)

When specifying position with percentages the first value sets the X-axis position and the second value sets the Y-axis position. If only one percentage is specified the second is assumed to be 50%. Interestingly, when computing the background image position the browser first identifies a point within the image at the specified coordinates, then places that point at the same coordinates within the content box. For example, with values of **50% 50%** the browser first identifies a point at the exact center of the image then places that point at the exact center of the content box.

Hot Tip The **background-position** property can also accept unit values to specify an offset position of the top left corner of the image from the top left corner of the content box. But this is only useful where the size of the content box cannot change – use keywords or percentages for greater flexibility.

The **background-attachment** property has a **scroll** value by default so the background image scrolls with the page but a style rule can specify a **fixed** value so it remains at specified coordinates relative to the viewport when the page gets scrolled.

Both **background-position** and **background-attachment** property values can be specified to the CSS **background** property, along with the **background-color, background-image**, and **background-repeat** values described on the previous page.

1. Create an HTML document with three paragraphs that each contain a span element

```
<p><span>Top Left - Default</span></p>
<p id="p1"><span>Bottom Left</span></p>
<p id="p2"><span>20% 50%</span></p>
```

position.html

2. Save the HTML document then create a linked style sheet containing rules to style the spans and paragraphs

```
span { background : yellow; margin : 0 0 0 10em }
p { background : url(tile.png) fuchsia no-repeat;
                                          height : 3.5em }
```

position.css

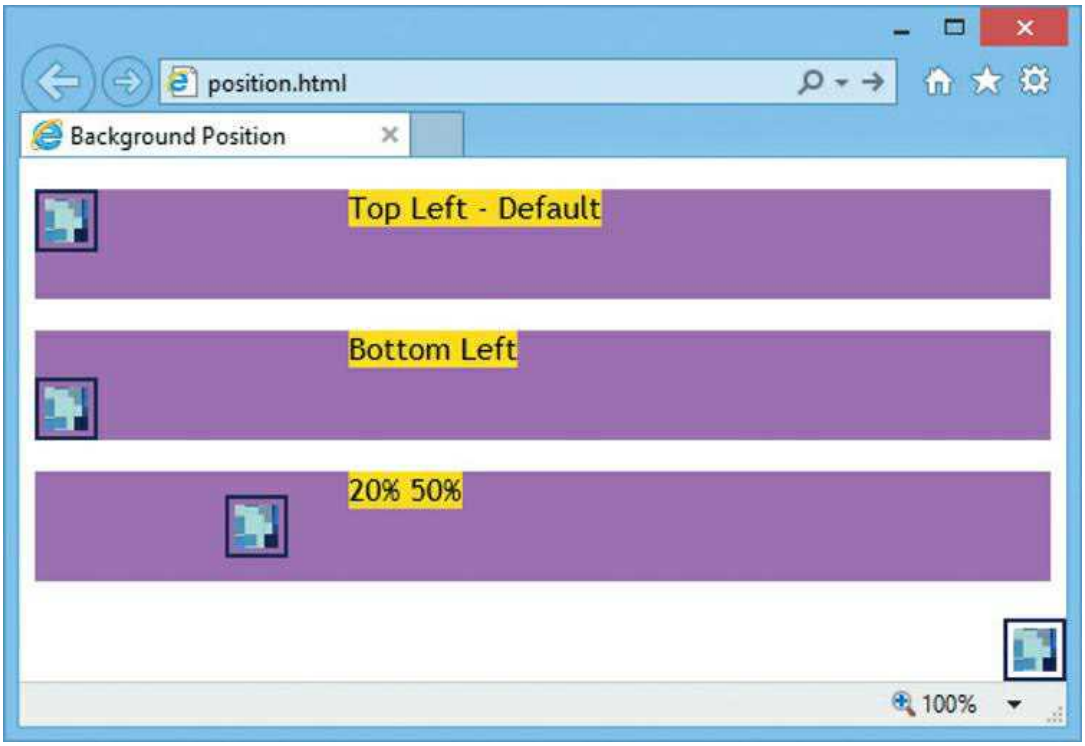
3. Now add rules to position paragraph background images using keywords and percentages

```
p#p1 { background-position : bottom left }
p#p2 { background-position : 20% 50% }
```

4. Finally add a rule to fix a background image in the page that will remain positioned when the page gets scrolled

```
body { background : url(tile.png) no-repeat
                                    bottom right fixed }
```

5. Save the style sheet alongside the HTML document then open the web page in a browser to see the positioned background images



Summary

- Block-level content boxes add line breaks before and after the box – but inline content boxes appear within lines of text

- The core content area of a content box can be surrounded by **padding, border**, and **margin** areas
- A content box's **display** property can be used to change its display format from block-level to inline, and vice versa
- Content box dimensions can be specified to **width** and **height** properties as absolute values of **in, cm, mm, pt**, and **pc** units, or as relative values of **em, ex**, and **px** measurements
- The **border** property can specify the individual **border-width, border-color**, and **border-style** properties using CSS shorthand
- Border styles may be **solid, double, dotted, dashed, groove, ridge, inset, outset, hidden** or **none**
- The **padding** property can specify the individual **padding-top, padding-right, padding-bottom**, and **padding-left** properties using CSS shorthand
- The **margin** property can specify the individual **margin-top, margin-right, margin-bottom**, and **margin-left** properties using CSS shorthand
- Colors can be specified using the standard color keywords, or as a three-part **rgb()** value with percentage or numerical parts, or as a single hexadecimal value
- The **background** property can specify the individual **background-color, background-image, background-repeat, background-position**, and **background-attachment** properties using CSS shorthand
- Background images can be specified as a **url()** value in which the path to the image file is stated within the parentheses