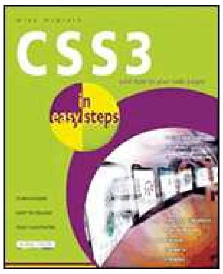


CSS3

by Mike McGrath

Chapter 5: Formatting Text

This chapter demonstrates how to suggest specific font styles for text content.

Suggesting a Font

A CSS style rule can suggest a specific font to be used by the browser for the display of text content in a selected element by specifying the font name to its **font-family** property. The browser will use the specified font if it is available on the user's system – otherwise it will display the text using its default font.

The default font may not be the best choice for the author's purpose so CSS additionally allows the **font-family** property to suggest a generic font family from those in the table below:

Font Family :	**Description :**	**Example :**
serif	Proportional fonts where characters have different widths to suit their size, and with serif decorations at the end of the character strokes	**Times New Roman**
sans-serif	Proportional fonts without serif decorations	**Arial**
monospace	Non-proportional fonts where characters are of fixed width, similar to type-written text	**Courier**
cursive	Fonts that attempt to emulate human hand-written text	**Comic Sans**
fantasy	Decorative fonts with highly graphic appearance	**Castellar**

Hot Tip Develop the habit of enclosing all named fonts within quotes.

The browser will first try to apply the named font but in the event that it is unavailable will select a font from those available that most closely matches the characteristics of the generic preference. In this way the appearance of the text should at least approximate the author's intention, even without specific font-matching.

In a style rule **font-family** declaration the preferred font name should appear before the generic font family preference separated by a comma. Multiple named fonts can be specified as a comma-separated list – all before the generic font family preference. Font names that include spaces must be enclosed within quotes or they will not be recognized by the browser.

1. Create an HTML document containing a paragraph with several spanned sections of text

```
<p>The <span class="serif">City of New York</span>
was introduced to professional football on the same day
that the city was introduced to the
<span class="fantasy">New York Giants</span>.
It was a clear sunny
<span class="mono">October afternoon in 1925</span>
when the Giants took the field to play against the
<span class="cursive">Frankford Yellow Jackets</span>.
</p>
```

2. Save the HTML document then create a linked style sheet containing a rule suggesting a default font for the entire paragraph

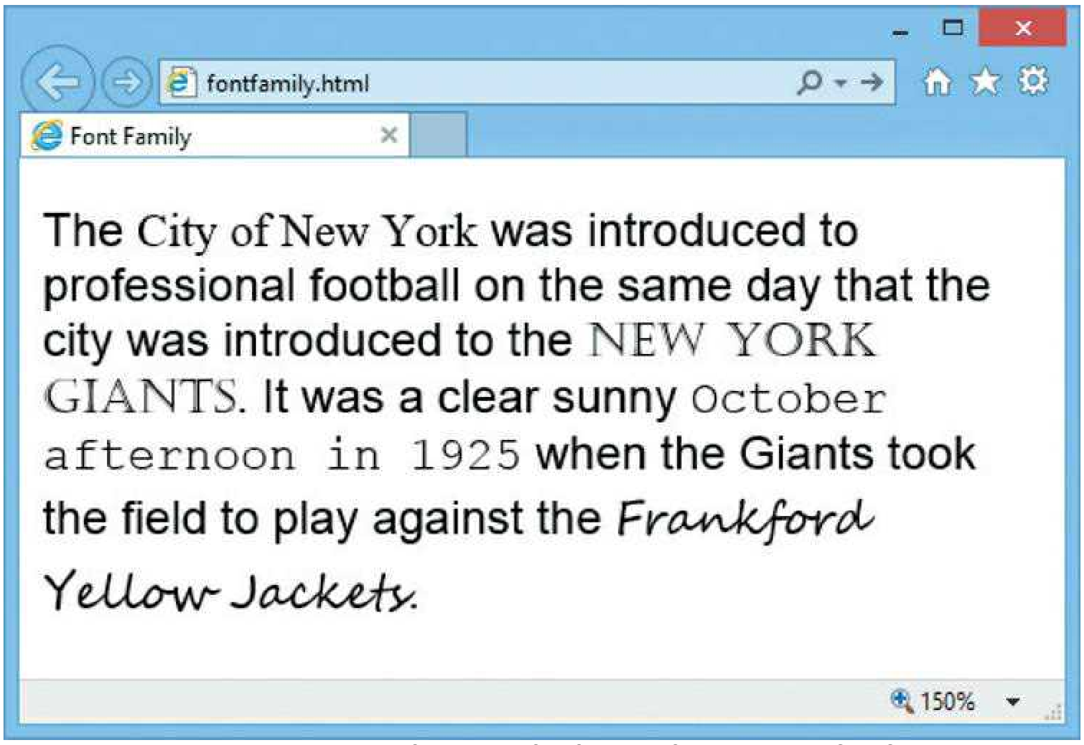
```
p { font-family : "Arial Narrow", sans-serif }
```

fontfamily.css

3. Next add style rules suggesting fonts for the spanned text

```
span.serif { font-family : "Times New Roman", serif }
span.fantasy { font-family : "Castellar", fantasy }
span.mono { font-family : "Courier", monospace }
span.cursive { font-family : "Segoe Script", cursive }
```

4. Save the style sheet alongside the HTML document then open the web page in a browser to see the sections of text appear in the named fonts or generic family fonts



The City of New York was introduced to professional football on the same day that the city was introduced to the NEW YORK GIANTS. It was a clear sunny October afternoon in 1925 when the Giants took the field to play against the Frankford Yellow Jackets.

Don't Forget It is good practice to specify a generic font family preference in every **font-family** declaration.

Specifying Font Size

CSS provides a number of ways to specify the size of text in a style rule declaration by assigning values to the **font-size** property. The most obvious is as an absolute size using any of the absolute units listed on page 42. For example, as a **12pt** size like this:

```
#text-1 { font-size : 12pt }
```

Additionally CSS provides the keywords **xx-small, x-small, small, medium, large, x-large**, and **xx-large** to specify an absolute size.

Keyword :	Equivalent :
xx-large	24pt
x-large	17pt
large	13.5pt
medium	12pt
small	10.5pt

Keyword :	Equivalent :
x-small	7.5pt
xx-small	7pt

Here the **medium** size is determined by the browser's default font size then the rest are computed relative to that size.

Where the default size is equivalent to **12pt** the computed values might look something like those in the table on the left.

So a style rule might specify a font size one scale higher than the default font size like this:

```
#text-2 { font-size : large }
```

Declarations may also specify a font size one scale higher or lower using the **larger** and **smaller** keywords like this:

```
#text-3 { font-size : larger }
```

Hot Tip Use relative values rather than absolute values to specify font sizes for maximum flexibility.

Alternatively font size can be specified as a relative size using any of the relative units listed on page 42. For example, double the inherited size (or default size if none is inherited) like this:

```
#text-4 { font-size : 2em }
```

Similarly, font size can be specified as a relative size stated as a percentage. For example, as five times the inherited or default size with a style rule declaration like this:

```
#text-5 { font-size : 500% }
```

1. Create an HTML document containing a paragraph with several spanned sections of text

   ```
   <p>This 12pt high text can become
   <span class = "lg">larger</span> or
   <span class = "sm">smaller</span> <br>It can get
   <span class = "xxl">extremely large</span> or
   <span class = "xxs">extremely small</span> <br>
   and it may even grow to be
   <span class = "huge">huge</span>
   </p>
   ```

   fontsize.html

2. Save the HTML document then create a linked style sheet containing a rule specifying a default font size for the entire paragraph

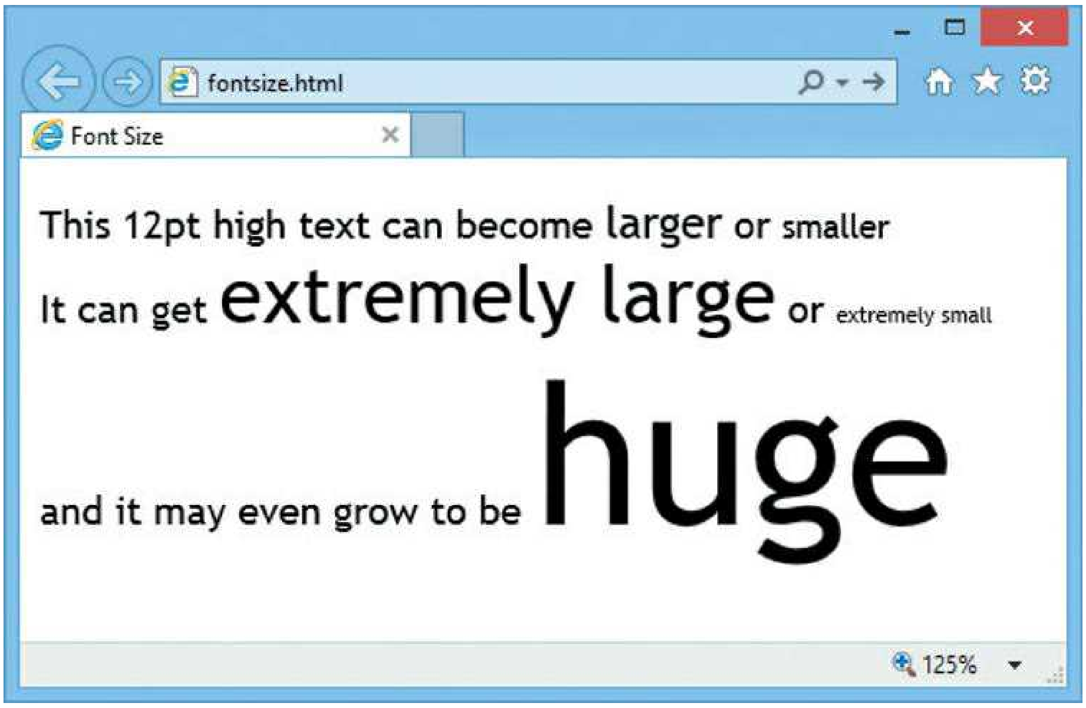
   ```
   p { font-size : 12pt }
   ```

   fontsize.css

3. Next add style rules specifying font sizes for the spanned

   ```
   span.lg { font-size : larger }
   span.sm { font-size : smaller }
   span.xxl { font-size : xx-large }
   span.xxs.{ font-size : xx-small }
   span.huge { font-size : 500% }
   ```

4. Save the style sheet alongside the HTML document then open the web page in a browser to see the sections of text appear in the specified font sizes



This 12pt high text can become larger or smaller

It can get extremely large or extremely small

and it may even grow to be huge

Adjusting Font Weight

The thickness or "weight" of text can easily be adjusted using the CSS **font-weight** property and the **bold** and **normal** keywords.

Specifying a **bold** value to a selected element's **font-weight** property causes normally-weighted text to appear in a heavier font and specifying a **normal** value causes heavily-weighted text to appear in a lighter font. In actuality the browser uses two different fonts to achieve this effect – for **normal** text it uses a regularly-weighted font (for example "Verdana") but it switches to the heavier-weighted variant of that font if one is available (such as "Verdana Bold") for **bold** text.

Don't Forget The **font-weight** numeric values do not need to specify a unit type because they are effectively keywords.

As an alternative to the bold and normal keywords a font weight can be specified as a numeric value from 100-900, by hundreds. A **font-weight** value of **400** is equivalent to **normal** weight and a value of **700** is equivalent to **bold** weight.

It is intended that this numeric weighting system should allow for font variants other than the **normal** and **bold** ones to be allotted to intermediate values. For example, a font lighter than the **normal** font (say, "Verdana Light") could equate to the weight value **300**. Similarly, a font heavier than the **bold** font (say, "Verdana Bold Black") could equate to the numeric weight value **800.**

In practice, however, the numeric system typically uses the **normal** font for values of **100, 200, 300, 400, 500** and the **bold** font for values of **600, 700, 800, 900.**

CSS also provides the keywords **bolder** and **lighter** for the purpose of moving up or down the font-weight ladder by single steps – but where the browser only recognizes two fonts, one for normal text and the other for bold text, these simply have the same effect as the **normal** and **bold** keywords.

1. Create an HTML document with two paragraphs containing spanned sections of text

```
<p id = "para-1">This normal weight text can become
<span class = "bold">bold</span> or
<span class = "more">bolder</span>
</p>

<p id = "para-2">This bold weight text can get
<span class = "norm">light</span> or
<span class = "less">lighter</span>
</p>
```

2. Now add a heading containing a spanned section of text

```
<h2>...here's a heading with
<span class = "less">lighter</span> text</h2>
```

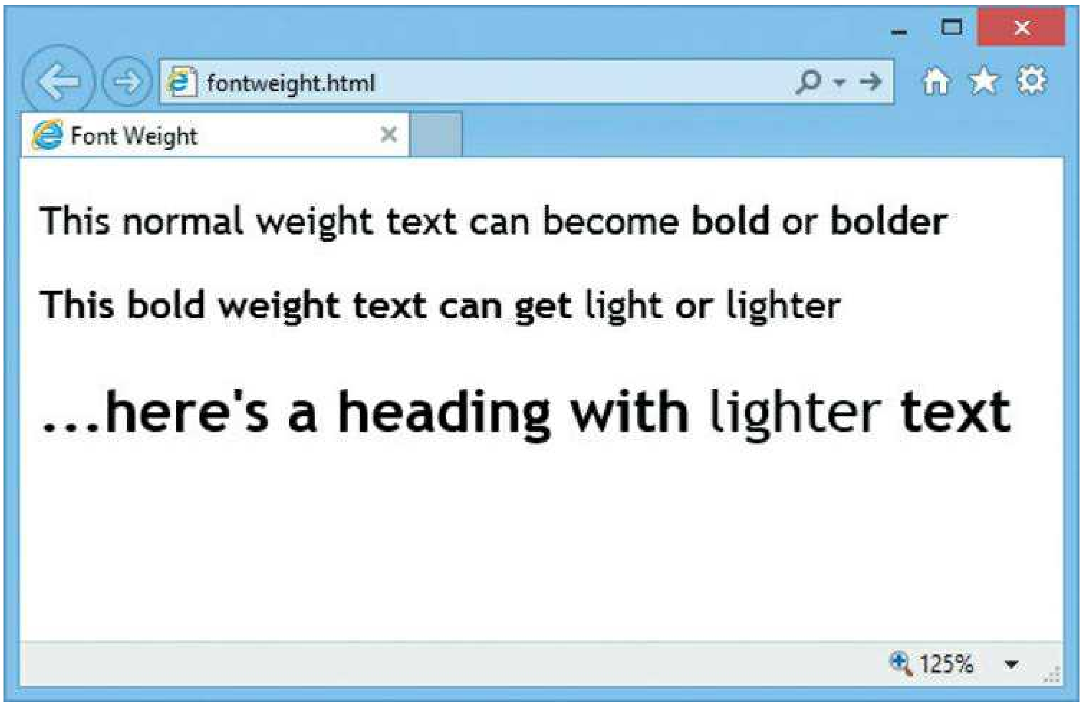
3. Save the HTML document then create a linked style sheet containing rules specifying a default font weight for each paragraph

```
p#para-1 { font-weight : 400 }
p#para-2 { font-weight : 700 }
```

4. Next add style rules specifying font weights for the spanned sections of text

```
span.bold { font-weight: bold }
span.more { font-weight: bolder }
span.norm { font-weight: normal }
span.less { font-weight: lighter }
```

5. Save the style sheet alongside the HTML document then open the web page in a browser to see the sections of text appear in the specified font weights



This normal weight text can become **bold** or **bolder**

This bold weight text can get light or lighter

...here's a heading with lighter **text**

Varying Font Styles

Slanting Text

A CSS **font-style** property can request the browser to use a slanting variant of the current font by specifying the **italic** or **oblique** keywords – these are subtly different. When the **italic** keyword is specified the browser seeks an italicized variant of the current font in its font database. This is an actual font set, similar to the current upright font but graphically different to produce slanting versions of each upright character. When the **oblique** keyword is specified the browser seeks an oblique variant of the current font in its font database. This may be an actual font set, a slanting version of the current upright font, or alternatively it may be a generated version in which the browser has computed a slanting version of the upright font. Either may be mapped to the **oblique** keyword in the browser font database and called upon by the CSS **font-style** property. In reality using either **italic** or **oblique** keywords typically produces the same italicized text appearance, and in each case upright text can be resumed by specifying the **normal** keyword to the element's **font-style** property.

Hot Tip Specify the **small-caps** value to the **font-variant** property of heading elements to make document headings more interesting.

Small Capitals

A CSS property called **font-variant** can specify a **small-caps** value to allow text characters to be displayed in a popular format using uppercase characters of two different sizes. Uppercase text in the selected element will appear as large capital characters but lowercase text will appear as smaller capitals. The browser may achieve this effect using a smaller capital from the font set, or by generating a computed smaller version. Once again regular text can be resumed by specifying the **normal** keyword as the **font-variant.**

1. Create an HTML document with two paragraphs and a heading containing spanned sections of text

```
<p id = "para-1">This normal style text can become
<span class = "ital">italicized</span> or
<span class = "oblq">oblique</span> </p>
<p id = "para-2">This italic style text can become
<span class = "reg">normal</span> </p>
<h2>...And Here's a
<span class="reg-caps">Heading</span>
with Small Caps</h2>
```

fontstyle.html

2. Save the HTML document then create a linked style sheet containing rules specifying default font styles for each paragraph

```
p#para-1 { font-style : normal }
p#para-2 { font-style : italic }
```

fontstyle.css

3. Next add style rules specifying font styles for the spanned sections of text in the paragraphs

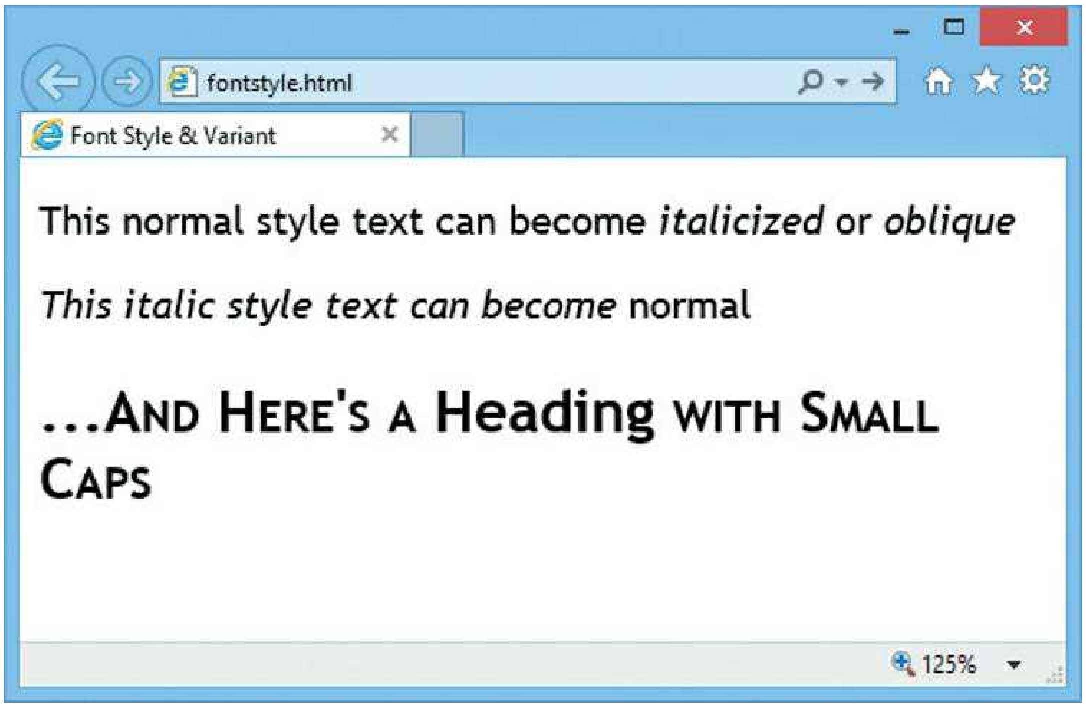
```
span.ital { font-style: italic }
span.oblq { font-style: oblique }
span.reg { font-style: normal }
```

4. Now add style rules specifying font variants for the heading and its spanned text

```
h2 { font-variant: small-caps }
span.reg-caps { font-variant : normal }
```

5. Save the style sheet alongside the HTML document then open the web page in a browser to see the sections of text

appear in the specified font styles and variants



This normal style text can become *italicized* or *oblique*

This italic style text can become normal

...And Here's a Heading with Small Caps

Using the Font Shorthand

Usefully CSS provides a **font** property to which various font preferences can be specified in a combined single rule stating:

```
font-style | font-variant | font-weight | font-size | font-family
```

Appropriate values can be assigned to each part of the combined **font** shorthand property. For example, like this:

```
p { font : italic small-caps bold medium "Times", serif }
```

Beware Remember that a **normal** value is applied for each part of a combined **font** rule unless explicitly specified – and this will override the current parent element value.

The first three values for the **font-style, font-variant**, and **font-weight** properties may appear in any order. Optionally values for each one of these properties may be completely omitted and a **normal** value will be automatically assumed.

It is important to recognize that values not explicitly specified will still have a **normal** value applied – no value is inherited from the containing element, and this can produce some unexpected results. For example, a style rule selecting a span element within a containing paragraph element styled by the rule above might look like this:

```
span { font : large cursive }
```

The values explicitly specified in this rule will be applied to the **font-size** and **font-family** properties of the span element, and a **normal** value will be applied to its **font-style, font-variant**, and **font-weight** properties – so text within the span element does not inherit the **italic, small-caps**, or **bold** values from the paragraph.

One further possibility available with a combined **font** rule is the option to specify a **line-height** (the spacing between each line) by adding a forward slash and unit value after the **font-weight** value. This is useful to establish a common standard line spacing where various font sizes appear.

font.html

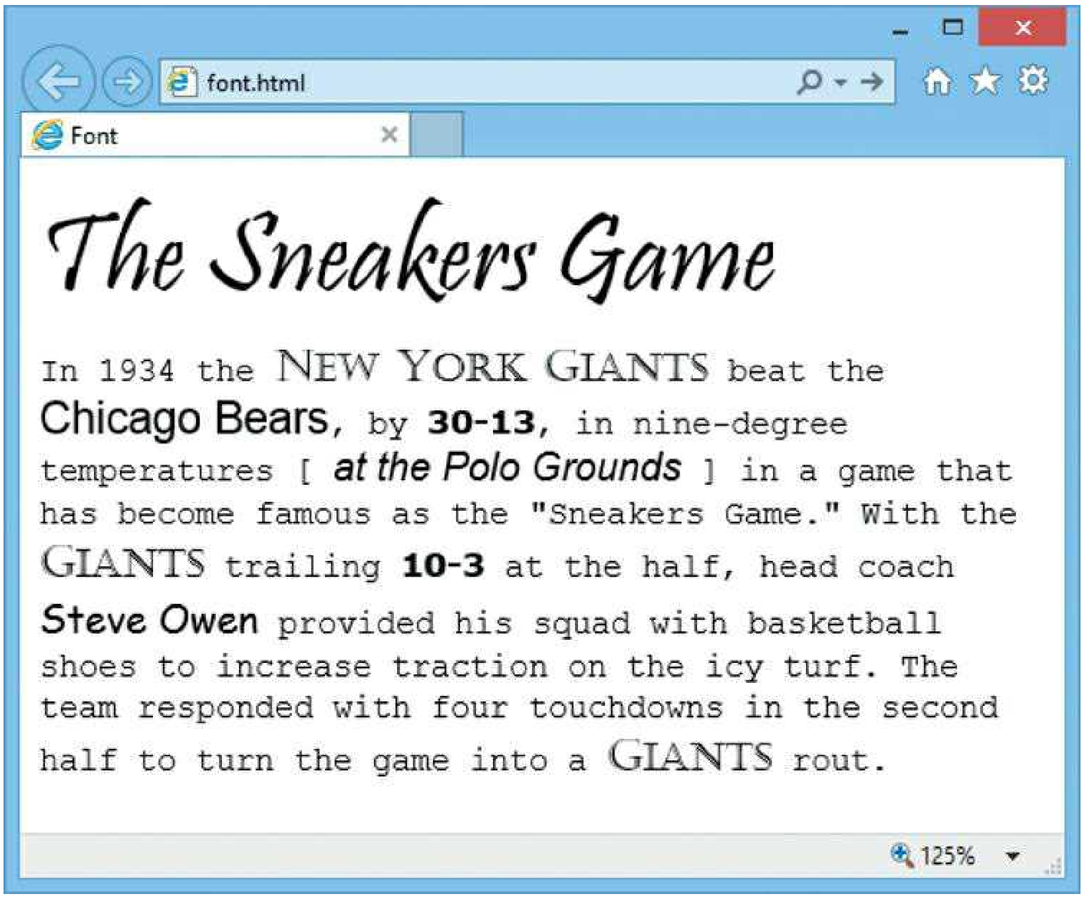
1. Create an HTML document containing a paragraph with several spanned sections of text

```
<span class = "header">The Sneakers Game</span><br>
In 1934 the
<span class = "giant">New York Giants</span> beat the
<span class="bears">Chicago Bears</span>, by
<span class = "score">30-13</span>,
in nine-degree temperatures
[ <span class="stadium">at the Polo Grounds</span> ]
in a game that has become famous as the "Sneakers
Game." With the <span class ="giant">Giants</span>
trailing <span class = "score">10-3</span> at the half,
head coach <span class = "coach">Steve Owen</span>
provided his squad with basketball shoes to increase
traction on the icy turf. The team responded with four
touchdowns in the second half to turn the game into a
<span class = "giant">Giants</span> rout. </p>
```

font.css

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

```
p.general { font : normal small/1.3em "Courier", monospace }
span.header { font : 350% "Pristina", cursive }
span.giant { font : small-caps large "Castellar", fantasy }
span.bears { font : large "Arial", sans-serif }
span.score { font : bold small "Verdana", sans-serif }
span.stadium { font : italic medium "Arial", sans-serif }
span.coach { font : medium "Comic Sans MS", cursive }
```



The Sneakers Game

In 1934 the NEW YORK GIANTS beat the Chicago Bears, by 30-13, in nine-degree temperatures [at the Polo Grounds] in a game that has become famous as the "Sneakers Game." With the GIANTS trailing 10-3 at the half, head coach Steve Owen provided his squad with basketball shoes to increase traction on the icy turf. The team responded with four touchdowns in the second half to turn the game into a GIANTS rout.

3. Save the style sheet alongside the HTML document then open the web page in a browser to see the font styles

Hot Tip Always use the **font** shorthand property rather than the individual **font-style, font-variant, font-weight, font-size**, and **font-family** properties.

Aligning Text

English language text in a paragraph is normally horizontally aligned to the left edge of the paragraph and this is the default behavior to display text in a paragraph element's content box.

Additionally CSS provides a **text-align** property that can explicitly specify how text should be aligned horizontally within the paragraph element's content box using the keywords **left, center, right**, or **justify**. As expected the **left** value aligns each line to the paragraph's left edge, the **right** value aligns each line to the paragraph's right edge, and the **center** value aligns each line centrally between both edges.

Don't Forget The **text-align** property only controls alignment of text within a content box – it is not used to center content boxes. See page 58 for details on how to center content boxes.

Perhaps more interestingly the **justify** value aligns each full line to both left and right edges of the content box and adjusts the spacing between characters and words to make each line the same length.

In displaying lines of text the browser automatically computes the line height to suit the content size – typically this will be the height of the font × 1.2. The browser then displays the text centered vertically in invisible "line boxes" of the computed height.

The CSS **vertical-align** property can explicitly specify how text should be aligned vertically using the keywords **baseline, sub**, and **super**. The **baseline** value specifies central vertical alignment, the default behavior. The **sub** and **super** values increase the boundaries of the outer container in which the line box exists and shift the text down or up respectively to display subscript or superscript.

Content can also be shifted up or down by specifying positive or negative unit values, or percentage values, to the **vertical-align** property. Alternatively the **top, middle**, and **bottom** keywords can specify vertical alignment with top-most, middle, or bottom-most content.

Hot Tip Usually subscript and superscript is much smaller than the text – create the vertical shift by specifying **sub** or **super** values then apply a **font** rule to reduce the shifted text's size.

Two other keywords of **text-top** and **text-bottom** can be specified to the **vertical-align** property in order to vertically align other inline content boxes, such as those of image elements, to the top or bottom edge of a line box.

1. Create an HTML document containing three paragraphs

```
<p>Enjoy the sunsets, the restaurants, the fishing, the
diving... the lifestyle of the Florida Keys!</p>

<p id="equalize">Enjoy the sunsets, the restaurants, the
fishing, the diving... the lifestyle of the Florida Keys!</p>

<p>Line <span class="up">Superscript</span>
<span class="down">Subscript</span>
<span class="top">Top</span></p>
```

align.html

2. Save the HTML document then create a linked style sheet containing a rule to specify the font and colors

```
p, span { font : medium monospace;
             background : yellow; border : 1px solid red }
```

align.css

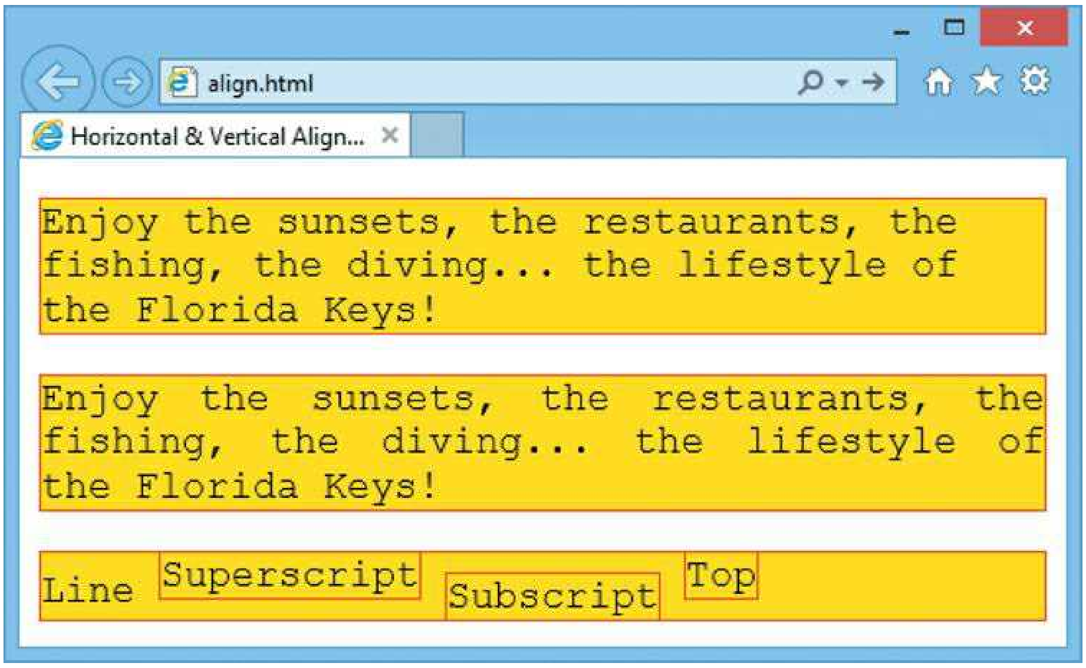
3. Next add a rule to horizontally justify the text within the second paragraph's content box

```
p#equalize { text-align : justify }
```

4. Now add rules to adjust the vertical alignment of spanned text within the third paragraph

```
span.up { vertical-align : super }
span.down { vertical-align : sub }
span.top { vertical-align : top }
```

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



Enjoy the sunsets, the restaurants, the fishing, the diving... the lifestyle of the Florida Keys!

Enjoy the sunsets, the restaurants, the fishing, the diving... the lifestyle of the Florida Keys!

Line Superscript Subscript Top

Indenting & Spacing Text

One of the most common features of printed text is the indentation of the first line of each paragraph to improve readability. This can easily be accomplished for text in HTML paragraphs using the **text-indent** property to specify an indentation size, such as **5em.**

Alternatively the indentation value may be specified as a percentage where the browser will indent an amount relative to the total line length. For example, given a paragraph element within a "div" container element of **500px** width, specifying a **text-indent** value of **10%** would indent the start of the first line by **50px** (500 × 10% = 50).

Hot Tip The **word-spacing** and **letter-spacing** properties can both accept negative values – to produce some interesting results.

It is also possible to specify negative values for the **text-indent** property but this can produce inconsistent results so is best avoided.

The amount of space between each word can be adjusted from the **normal** default spacing by explicitly specifying a value to the CSS **word-spacing** property. Note that the specified value is added to the default spacing to increase the space. For example, specifying a unit value of **5em** increases the space to **normal + 5em**, not a spacing of **5em** overall.

Similarly the amount of space between each letter can be adjusted from the **normal** default spacing by explicitly specifying a value to the CSS **letter-spacing** property. This also adds the specified value onto the default spacing to determine the total space. For example, specifying a unit value of **5em** increases the space to **normal + 5em**, not a spacing of **5em** overall.

Both **word-spacing** and **letter-spacing** properties accept the **normal** keyword to resume normal spacing. Also they may both be overridden by the **text-align** property, described on the previous page, that has precedence in determining the appearance of the entire line.

1. Create an HTML document with two paragraphs containing spanned text

```
<p>The Geologic Story at the
<span class="spread">Grand Canyon</span>
attracts the attention of the world for many reasons, but
perhaps its greatest significance lies in the geologic record
preserved and exposed here.</p><p>The rocks at
<span class="spread">Grand Canyon</span>
are not inherently unique but the
<span class="space">variety of rocks clearly exposed
present a complex geologic story.</span> </p>
```

space.html

2. Save the HTML document then create a linked style sheet containing a rule to indent the start of each paragraph

```
p { text-indent : 5em }
```

space.css

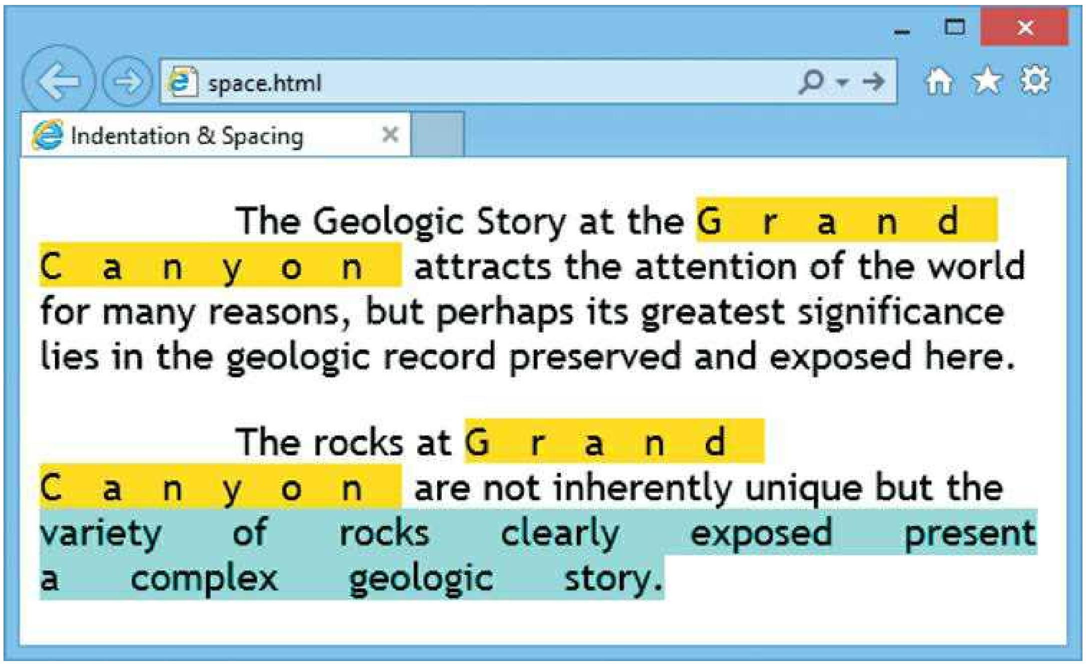
3. Next add a style rule to increase the letter spacing and set a background color on two spanned sections of text

```
span.spread { letter-spacing : 1em; background : yellow }
```

4. Now add a style rule to increase the word spacing and set a background color on the other spanned text

```
span.space { word-spacing : 1.5em; background : aqua }
```

5. Save the style sheet alongside the HTML document then open the web page in a browser to see the indentations and spacing



The Geologic Story at the G r a n d C a n y o n attracts the attention of the world for many reasons, but perhaps its greatest significance lies in the geologic record preserved and exposed here.

The rocks at G r a n d C a n y o n are not inherently unique but the variety of rocks clearly exposed present a complex geologic story.

Decorating Text

Style rules can add decorative lines to text content using the CSS **text-decoration** property with keywords **underline, overline**, and **line-through**. These behave as expected adding a line below, a line above, and a line through the text respectively.

The **blink** keyword may also be specified to the **text-decoration** property to cause the browser to repeatedly hide then display the text, blinking it on and off.

Don't Forget Some users may not recognize hyperlinks if their default underline is removed.

More usefully, the CSS **none** keyword can be specified to the **text-decoration** property to prevent unwanted decorations appearing – this is particularly popular for displaying hyperlinks without their usual default underline.

Multiple keywords can be specified to the **text-decoration** property as a space-separated list to apply multiple decorations to the text.

An additional way to enhance text with CSS is available using the **text-transform** property to specify capitalization in the selected element with the keywords **uppercase, lowercase**, or **capitalize.**

1. Create an HTML document with a paragraph containing spanned text and another paragraph containing hyperlinks, separated by a ruled line

```
<p id="para-1">You know that it's
<span class = "under caps">important</span>
when<br>it is
<span class = "under">underlined</span>
<br>and that it's been
<span class = "thru caps">cancelled</span>
when<br>it has been
<span class = "thru">struck through</span>
<br>but you also must remember to<br>
<span class = "rails upper">read between the lines</span>
<br>
<span id = "sig" class = "lower"> - MIKE MCGRATH</span>
</p>
<hr>
<p>
<a href = "http://w3c.org">Regular link</a> |
<a class = "plain" href = "http://w3c.org">Plain link</a>
</p>
```

decor.html

2. Save the HTML document then create a linked style sheet with rules to specify fonts and colors

```
#para-1 { font : medium "Courier", monospace;
                                  background : #CCCCFF }
#sig { font : xx-large "Lucida Handwriting", cursive;
                                              color : purple }
```

decor.css

3. Next add style rules to decorate spanned text with lines

```
span.under { text-decoration : underline }
span.thru { text-decoration : line-through }
span.rails { text-decoration : overline underline blink }
```

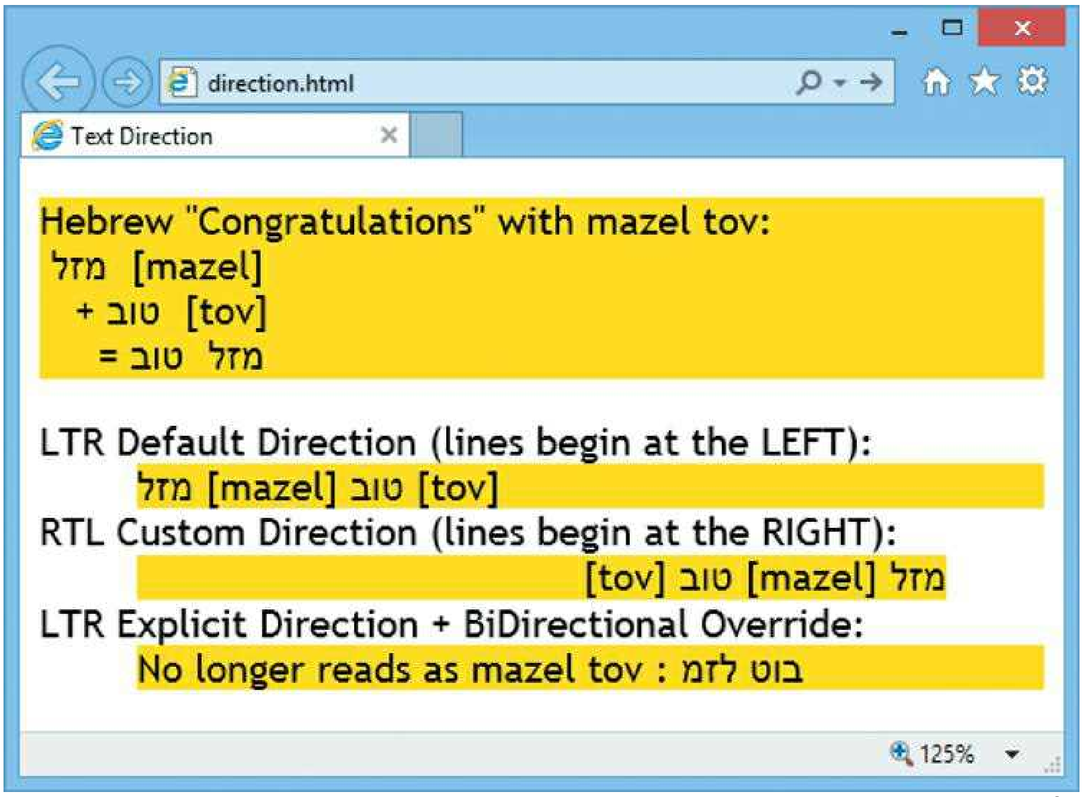
4. Now add style rules to transform the case of spanned text

```
span.lower { text-transform : lowercase }
span.upper { text-transform : uppercase }
span.caps { text-transform : capitalize }
```

5. Finally add a style rule to remove the default underline from a hyperlink

```
a.plain { text-decoration : none }
```

6. Save the style sheet alongside the HTML document then open the web page in a browser to see the text decorations and case transformations



Hebrew "Congratulations" with mazel tov:
מזל [mazel]
+ טוב [tov]
= מזל טוב

LTR Default Direction (lines begin at the LEFT):
מזל [mazel] טוב [tov]
RTL Custom Direction (lines begin at the RIGHT):
מזל [mazel] טוב [tov]
LTR Explicit Direction + BiDirectional Override:
No longer reads as mazel tov : בוט לזמ

Beware Avoid using the **blink** value as blinking text is despised by many users, and is not supported by default in some browsers.

Governing Space & Direction

The default treatment of whitespace within text content is to collapse multiple spaces into a single space, but this can be controlled with the CSS **white-space** property. Specifying the **pre** keyword preserves all spaces as they appear in the original text, including any line breaks. Conversely, the automatic wrapping of text in a block can be prevented by specifying the **no-wrap** keyword. Additionally the **pre-wrap** keyword can be specified to preserve spaces while still allowing text to wrap normally, or the **pre-line** keyword can be specified to collapse multiple spaces while preserving line breaks.

Hot Tip You can discover more about Unicode online at **www.unicode.org** and more on character entities at **www.w3.org.**

The default left-to-right direction of text lines can be changed to right-to-left by specifying the **rtl** keyword to the CSS **direction** property and the normal direction resumed with the **ltr** keyword.

Interestingly, when the line direction is changed with the **rtl** keyword the words appear from right-to-left but the order of English language characters is preserved so that each word still reads correctly left-to-right.

This intelligent feature also allows text to be presented in different directions on a single line. For example, to incorporate words in languages that are read right-to-left such as Hebrew and Arabic. The browser examines the Unicode value of each character using a complex BiDirectional algorithm to determine which direction each word should be displayed – those characters from right-to-left languages are automatically displayed in that direction, even if written logically from left-to-right in the HTML source code. The automatic BiDirectional algorithm can be turned off however by specifying the **bidi-override** keyword to a **unicode-bidi** property.

1. Create an HTML document with a paragraph containing Hebrew character entities and stepped whitespace

```
<p>Hebrew "Congratulations" with mazel tov:
&#1502;&#1494;&#1500;  [mazel]
 + &#1496;&#1493;&#1489;  [tov]
  = &#1502;&#1494;&#1500;  &#1496;&#1493;&#1489;
</p>
```

direction.html

2. Next begin a definition list with the same entities

```
<dl>
<dt>LTR Default Direction (lines begin at the LEFT):</dt>
<dd class = "ltr">&#1502;&#1494;&#1500; [mazel]
&#1496;&#1493;&#1489; [tov]</dd>
```

3. Now add two more definitions to complete the list, again featuring the same Hebrew character entities

```
<dt>RTL Custom Direction (lines begin at the RIGHT):</dt>
<dd class = "rtl">&#1502;&#1494;&#1500; [mazel]
&#1496;&#1493;&#1489; [tov]</dd>
<dt>LTR Explicit Direction + BiDirectional Override:</dt>
<dd class = "bidi-off ltr">No longer reads as mazel tov :
&#1502;&#1494;&#1500; &#1496;&#1493;&#1489;</dd>
</dl>
```

4. Save the HTML document then create a linked style sheet with style rules to color element backgrounds and preserve whitespace in paragraphs

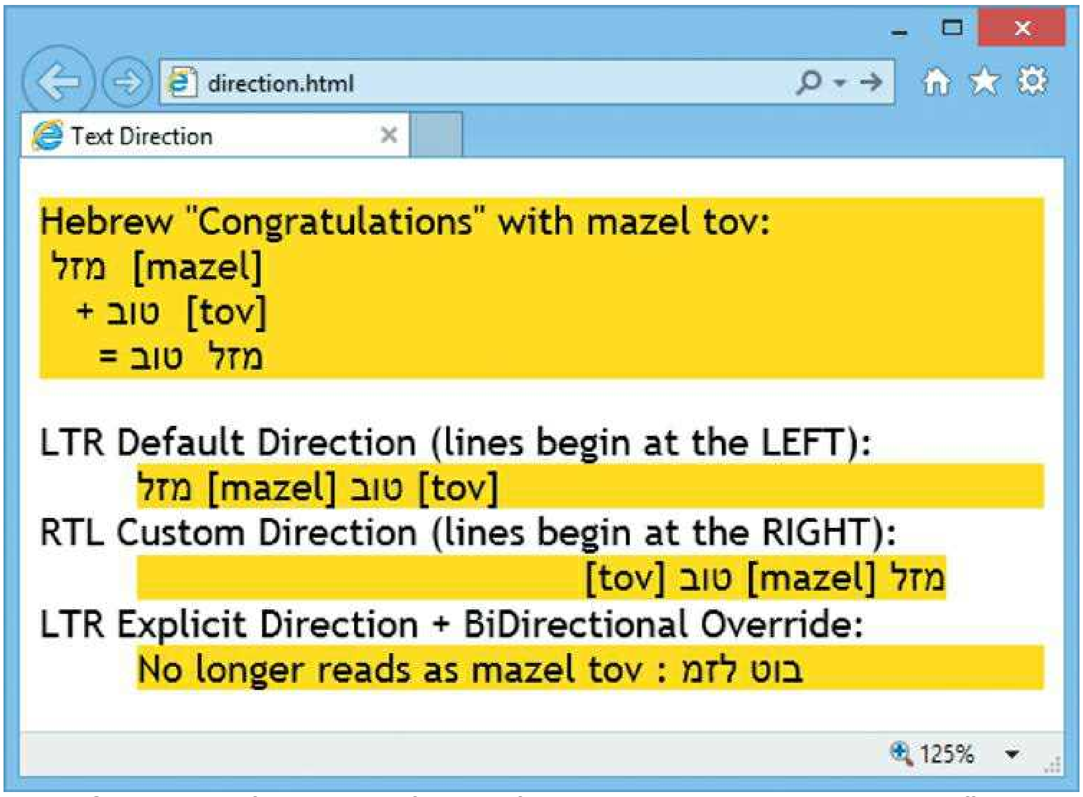
```
p,dd { background : yellow }
p { whitespace : pre }
```

direction.css

5. Now add style rules to set the text directions of each definition in the list

```
dd.ltr { direction : ltr }
dd.rtl { direction : rtl }
dd.bidi-off { unicode-bidi : bidi-override }
```

6. Save the style sheet alongside the HTML document then open the web page in a browser to see the preserved whitespace and various text directions



Hebrew "Congratulations" with mazel tov:
מזל [mazel]
+ טוב [tov]
= מזל טוב

LTR Default Direction (lines begin at the LEFT):
מזל [mazel] טוב [tov]
RTL Custom Direction (lines begin at the RIGHT):
[tov] טוב [mazel] מזל
LTR Explicit Direction + BiDirectional Override:
No longer reads as mazel tov : בוט לזמ

Beware Generally the default treatment of right-to-left language characters achieves the desired effect. In practice overriding the Unicode BiDirectional algorithm is seldom needed.

Summary

- The **font-family** property can suggest specific fonts by name and also specify a generic font family as **serif, sans-serif, monospace, cursive**, or **fantasy**
- Font size can be specified using keywords such as **large**, absolute sizes such as **12pt**, or relative sizes such as **larger**
- Thickness of text can be specified to the **font-weight** property using keywords such as **bold**, or numeric values such as **700**
- Slanting text can be created by specifying the **italic** or **oblique** keywords to the **font-style** property
- Specifying a **small-caps** value to the **font-variant** property causes lowercase characters to appear as small capital letters
- The **font** shorthand property can be used to specify values for the **font-style, font-variant, font-weight, font-size**, and **font-family** properties rather than individual rules
- The **font-weight** property may also specify a **line-height** by adding a forward slash and unit size after the weight value
- Horizontal position of text within a content box can be specified to the **text-align** property by keywords such as **center**
- Vertical inline position of text can be specified to the **vertical-align** property using keywords such as **super**
- The **text-indent** property allows the start of each paragraph to be indented by a specified distance
- Text spacing can be adjusted by the **word-spacing** and **letter-spacing** properties
- Lines can be added to text by the **text-decoration** property and the case can be specified to the **text-transform** property
- Specifying **pre** to the **white-space** property preserves spacing
- The **direction** property can control text direction but can be overridden by the **unicode-bidi** property