border, you also know how to specify the length of a margin and a padding and an image. This reuse of syntax makes working with different properties much easier.

Table 12-1
Sample Property Names and Values

Name	*Value*
display	none
font-style	italic
margin-top	0.5in
font-size	12pt
border-style	solid
color	#CC0033
background-color	white
background-image	url(http://www.idgbooks.com/images/paper.gif)
list-style-image	url(/images/redbullet.png)
line-height	120%

Length values

In CSS, a length is a scalar measure used for width, height, font-size, word and letter spacing, text indentation, line height, margins, padding, border widths, and many other properties. Lengths may be specified in three ways:

1. Absolute units
2. Relative units
3. Percentages

Absolute Units of Length

Absolute units of length are something of a misnomer because there's really no such thing as an absolute unit of length on a computer screen. Changing a monitor resolution from 640 to 480 to 1600 by 1200 changes the length of everything on your screen, inches and centimeters included. Nonetheless, CSS supports five "absolute" units of length that at least don't change from one font to the next. These are listed in Table 12-2.

Table 12-2
Absolute Units of Length

	Inch (in)	*Centimeters (cm)*	*Millimeters (mm)*	*Points (pt)*	*Picas (pc)*
Inch	1.0	2.54	25.4	72	6
Centimeters	0.3937	1	10	28.3464	4.7244
Millimeters	0.03937	0.1	1.0	2.83464	0.47244
Points	0.01389	0.0352806	0.352806	1.0	0.83333
Picas	0.16667	0.4233	4.233	12	1.0

Lengths are given as a number followed by the abbreviation for one of these units:

Inches	in
Centimeters	cm
Millimeters	mm
Points	pt
Picas	pc

The number may have a decimal point (for example, `margin-top: 0.3in`). Some properties allow negative values like -0.5in, but not all do; and even those that do often place limits on how negative a length can be. It's best to avoid negative lengths for maximum cross-browser compatibility.

Relative Units of Length

CSS also supports three relative units for lengths. These are:

1. em: the width of the letter *m* in the current font
2. ex: the height of the letter *x* in the current font
3. px: the size of a pixel (assumes square pixels; all common modern displays use square pixels though some older PC monitors, mostly now consigned to the rubbage bin, do not)

For example, this rule sets the left and right borders of the `PULLQUOTE` element to twice the width of the letter *m* in the current font and the top and bottom borders to one and a half times the height of the letter *x* in the current font:

```
PULLQUOTE { border-right-width: 2em; border-left-width: 2em;
            border-top-width: 1.5ex; border-bottom-width: 1.5ex }
```

The normal purpose of using ems and ex's is to set a width that's appropriate for a given font, without necessarily knowing how big the font is. For instance in the above rule, the font size is not known so the exact width of the borders is not known either. It can be determined at display time by comparison with the *m* and the *x* in the current font. Larger font sizes will have correspondingly larger ems and ex's.

Lengths in pixels are relative to the height and width of a (presumably square) pixel on the monitor. Widths and heights of images are often given in pixels.

Pixel measurements are generally not a good idea. First, the size of a pixel varies widely with resolution. Most power users set their monitors at much too high a resolution, which makes the pixels far too small for legibility.

Secondly, within the next ten years, 200 and even 300 dpi monitors will become common, finally breaking away from the rough 72-pixels-per-inch (give or take 28 pixels) de facto standard that's prevailed since the first Macintosh in 1984. Documents that specify measurements in nonscreen-based units like ems, ex's, points, picas, and inches will be able to make the transition. However, documents that use pixel level specification will become illegibly small when viewed on high-resolution monitors.

Percentage Units of Length

Finally, lengths can be specified as a percentage of something. Generally, this is a percentage of the current value of a property. For instance, if the font-size of a `STANZA` element is 12 points, and the font-size of the `VERSE` the `STANZA` contains is set to 150 percent, then the font-size of the `VERSE` will be 18 points.

URL Values

Three CSS properties can have URL values: `background-image`, `list-style-image`, and the shorthand property `list-style`. Furthermore, as you've already seen, the `@import` rule uses URL values. Literal URLs are placed inside `url()`. All forms of relative and absolute URLs are allowed. For example:

```
DOC { background-image: url (http://www.mysite.com/bg.gif) }
LETTER { background-image: url(/images/paper.gif) }
SOFTWARE { background-image: url(../images/screenshot.gif)}
GAME { background-image: url(currentposition.gif)}
```

You can enclose the URL in single or double quotes, though nothing is gained by doing so. For example:

```
DOC { background-image: url("http://www.mysite.com/bg.gif") }
LETTER { background-image: url("/images/paper.gif") }
SOFTWARE { background-image: url('../images/screenshot.gif')}
GAME { background-image: url('currentposition.gif')}
```

Parentheses, commas, whitespace characters, single quotes (') and double quotes (") appearing in a URL must be escaped with a backslash: '\(', '\)', '\,'. Any parentheses, apostrophes, whitespace, or quotation marks that appear inside the URL (uncommon except perhaps for the space character) should be replaced by URL standard % escapes. That is:

space	%20
,	%2C
'	%27
"	%22
(	%2B
)	%2C

CSS defines its own backslash escapes for these characters (`\(`, `\)`, `\,`, `\'`, and `\"`), but these only add an additional layer of confusion.

Color Values

One of the most widely adopted uses of CSS over traditional HTML is its ability to apply foreground and background colors to almost any element on a page. Properties that take on color values include `color`, `background-color`, and `border-color`.

CSS provides four ways to specify color: by name, by hexadecimal components, by integer components, and by percentages. Defining color by name is the simplest. CSS understands these 16 color names adopted from the Windows VGA palette:

- aqua
- black
- blue
- fuchsia
- gray
- green
- lime
- maroon
- navy
- olive
- purple
- red
- silver
- teal
- white
- yellow

Of course, the typical color monitor can display several million more colors. These can be specified by providing values for the red, green, and blue (RGB) components of the color. Since CSS assumes a 24-bit color model, each of these primary colors is assigned 8 bits. An 8-bit unsigned integer is a number between 0 and 255. This number may be given in either decimal RGB or hexadecimal RGB. Alternately, it

may be given as a percentage RGB between 0% (0) and 100% (255). Table 12-3 lists some of the possible colors and their decimal, hexadecimal, and percentage RGBs.

Table 12-3
CSS Sample Colors

Color	*Decimal RGB*	*Hexadecimal RGB*	*Percentage RGB*
Pure red	rgb(255,0,0)	#FF0000	rgb(100%, 0%, 0%)
Pure blue	rgb(0,0,255)	#0000FF	rgb(0%, 0%, 0%)
Pure green	rgb(0,255,0)	#00FF00	rgb(0%, 100%, 0%)
White	rgb(255,255,255)	#FFFFFF	rgb(100%, 100%, 100%)
Black	rgb(0,0,0)	#000000	rgb(0%, 0%, 0%)
Light violet	rgb(255,204,255)	#FFCCFF	rgb(100%, 80%, 100%)
Medium gray	rgb(153,153,153)	#999999	rgb(60%, 60%, 60%)
Brown	rgb(153,102,51)	#996633	rgb(60%, 40%, 20%)
Pink	rgb(255,204,204)	#FFCCCC	rgb(100%, 80%, 80%)
Orange	rgb(255,204,204)	#FFCC00	rgb(100%, 80%, 80%)

Many people still use 256 color monitors. Furthermore, some colors are distinctly different on Macs and PCs. The most reliable colors are the 16 named colors.

The next most reliable colors are those formed using only the hexadecimal components 00, 33, 66, 99, CC, and FF (0, 51, 102, 153, 204, 255 in decimal RGBs; 0%, 20%, 40%, 60%, 80%, 100% in percentage units). For instance, 33FFCC is a "browser-safe" color because the red component is made from two threes, the green from two F's, and the blue from two C's.

If you specify a hexadecimal RGB color using only three digits, CSS duplicates them; for example, #FC0 is really #FFCC00 and #963 is really #996633.

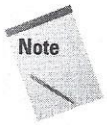
Keyword Values

Keywords are the most variable of the four kinds of values a CSS property may take on. They are not generally the same from property to property, but similar properties generally support similar keywords. For instance, the value of `border-left-style` can be any one of the keywords `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, or `outset`. The `border-right-style`, `border-top-style`, `border-bottom-style`, and `border-style` properties can also assume one of the values `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, or `outset`. The individual keywords will be discussed in the sections about the individual properties.

Block, Inline, and List Item Elements

From the perspective of CSS Level 1 all elements are either block-level elements, inline elements, list items, or invisible. (CSS Level 2 adds a few more possibilities.) The type of a given element is set by its `display` property. This property has four possible values given by keywords:

```
block
inline
list-item
none
```



In CSS Level 1, the default value of the `display` property is `block` which means that the item appears in its own box and is separated from other elements in some fashion. However, in CSS Level 2 the default has changed to `inline` which means that the contents of the element are simply placed sequentially in the text after the previous element. Most Web browsers use the CSS 2 default (`inline`) rather the CSS 1 default (`block`).

In HTML, `EM`, `STRONG`, `B`, `I`, and `A` are all inline elements. As another example, you can think of `EM`, `STRONG`, `B`, `I`, and `A` in this paragraph as inline code elements. They aren't separated out from the rest of the text.

Block-level elements are separated from other block-level elements, generally by breaking the line. In HTML `P`, `BLOCKQUOTE`, `H1` through `H6`, and `HR` are all examples of block-level elements. The paragraphs you see on this page are all block-level elements. Block-level elements may contain inline elements and other block-level elements, but inline elements should only contain other inline elements, not block-level elements.

List item elements are block-level elements with a list-item marker preceding them. In HTML, `LI` is a list-item element. List items are discussed further in the following section.

Finally, elements with their `display` property set to `none` are invisible and not rendered on the screen. Nor do they affect the position of other visible elements on the page. In HTML, `TITLE`, `META`, and `HEAD` would have a `display` property of `none`. In XML, `display: none` is often useful for meta-information in elements.

Consider Listing 12-7, a synopsis of William Shakespeare's *Twelfth Night*. It contains the following elements:

```
SYNOPSIS        ACT_NUMBER
TITLE           SCENE_NUMBER
ACT             LOCATION
SCENE           CHARACTER
```

You can do a fair job of formatting this data using only display properties. SYNOPSIS, TITLE, ACT, and SCENE are all block-level elements. ACT_NUMBER, SCENE_NUMBER, LOCATION, and CHARACTER can remain inline elements. Listing 12-8 is a very simple style sheet that accomplishes this.

Listing 12-7: A synopsis of Shakespeare's *Twelfth Night* in XML

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="12-8.css"?>
<SYNOPSIS>
  <TITLE>Twelfth Night</TITLE>

  <ACT>
    <ACT_NUMBER>Act 1</ACT_NUMBER>
    <SCENE>
      <SCENE_NUMBER>Scene 1</SCENE_NUMBER>
      <LOCATION><CHARACTER>Duke Orsino</CHARACTER>'s palace
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 2</SCENE_NUMBER>
      <LOCATION>The sea-coast</LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 3</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s house
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 4</SCENE_NUMBER>
      <LOCATION><CHARACTER>Duke Orsino</CHARACTER>'s palace.
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 5</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s house
      </LOCATION>
    </SCENE>
  </ACT>

  <ACT>
    <ACT_NUMBER>Act 2</ACT_NUMBER>
    <SCENE>
      <SCENE_NUMBER>Scene 1</SCENE_NUMBER>
      <LOCATION>The sea-coast</LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 2</SCENE_NUMBER>
      <LOCATION>A street</LOCATION>
    </SCENE>
```

Continued

Listing 12-7 *(continued)*

```
    <SCENE>
      <SCENE_NUMBER>Scene 3</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s house
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 4</SCENE_NUMBER>
      <LOCATION><CHARACTER>Duke Orsino</CHARACTER>'s palace.
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 5</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s garden
      </LOCATION>
    </SCENE>
  </ACT>

  <ACT>
    <ACT_NUMBER>Act 3</ACT_NUMBER>
    <SCENE>
      <SCENE_NUMBER>Scene 1</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s garden
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 2</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s house
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 3</SCENE_NUMBER>
      <LOCATION>A street</LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 4</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s garden
      </LOCATION>
    </SCENE>
  </ACT>

  <ACT>
    <ACT_NUMBER>Act 4</ACT_NUMBER>
    <SCENE>
      <SCENE_NUMBER>Scene 1</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s front yard
      </LOCATION>
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 2</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s house
      </LOCATION>
```

```
    </SCENE>
    <SCENE>
      <SCENE_NUMBER>Scene 3</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s garden
      </LOCATION>
    </SCENE>
  </ACT>

  <ACT>
    <ACT_NUMBER>Act 5</ACT_NUMBER>
    <SCENE>
      <SCENE_NUMBER>Scene 1</SCENE_NUMBER>
      <LOCATION><CHARACTER>Olivia</CHARACTER>'s front yard
      </LOCATION>
    </SCENE>
  </ACT>

</SYNOPSIS>
```

Listing 12-8: A very simple style sheet for the synopsis of a play

```
SYNOPSIS, TITLE, ACT, SCENE { display: block }
```

Figure 12-4 shows the synopsis of *Twelfth Night* loaded into Mozilla with the style sheet of Listing 12-8. Notice that in Listing 12-8 it is not necessary to explicitly specify that `ACT_NUMBER`, `SCENE_NUMBER`, `LOCATION`, and `CHARACTER` are all inline elements. This is the default unless otherwise specified. The `display` property is not inherited by children. Thus, just because `SCENE` is a block-level element does not mean that its children `SCENE_NUMBER` and `LOCATION` are also block-level elements.

List Items

If you choose the `list-item` value for the `display` property, there are three additional properties you can set. These properties affect how list items are displayed. These are:

1. `list-style-type`
2. `list-style-image`
3. `list-style-position`

There's also a shorthand `list-style` property that lets you set all three in a single rule.

Internet Explorer 5.0 and Mozilla 5.0 milestone 3 do not yet support `display: list-item`. Mozilla treats list items as simple block-level elements while Internet Explorer does even worse by treating them as inline elements.

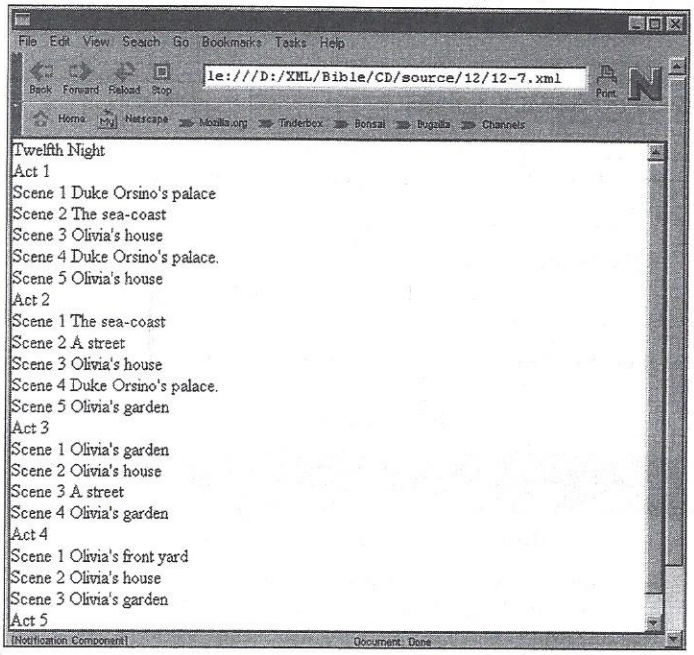


Figure 12-4: The synopsis of *Twelfth Night* as displayed in Mozilla 5.0

The list-style-type Property

The `list-style-type` property determines the nature of the bullet character in front of each list item. The possibilities are:

```
disc
circle
square
decimal
lower-roman
upper-roman
lower-alpha
upper-alpha
none
```

The default is `disc`. For example, the style sheet in Listing 12-9, which applies to the synopsis in Listing 12-7 defines `ACT` and `SCENE` as list items. However, `ACT` is given no bullet, and `SCENE` is given a square bullet.

Listing 12-9: A style sheet for a play synopsis that uses list items

```
SYNOPSIS, TITLE { display: block }
ACT { display: list-item; list-style-type: none }
SCENE { display: list-item; list-style-type: square }
```

The list-style-image Property

Alternately, you can use a bitmapped image of your choice loaded from a file as the bullet. To do this you set the `list-style-image` property to the URL of the image. If both `list-style-image` and `list-style-type` are set, the list item will be preceded by both the image and the bullet character. This is rare, however. Listing 12-10 uses a ♥ stored in the file heart.gif as the bullet before each scene (*Twelfth Night* is a romantic comedy after all).

Listing 12-10: A style sheet for a play synopsis that uses the list-style-image property

```
SYNOPSIS, TITLE { display: block }
ACT { display: list-item; list-style-type: none }
SCENE { display: list-item;
      list-style-image: url(heart.gif); list-style-type: none }
```

The list-style-position Property

The `list-style-position` property specifies whether the bullet is drawn inside or outside the text of the list item. The legal values are `inside` and `outside`. The default is `outside`. The difference is only obvious when the text wraps onto more than one line. This is inside:

- If music be the food of love, play on/Give me excess of it, that, surfeiting,/The appetite may sicken, and so die./That strain again! it had a dying fall:

This is outside:

- If music be the food of love, play on/Give me excess of it, that, surfeiting,/The appetite may sicken, and so die./That strain again! it had a dying fall:

The list-style Shorthand Property

Finally, the `list-style` property is a short hand that allows you to set all three of the above-described properties at once. For example, this rule says that a `SCENE` is displayed inside with a heart image and no bullet:

```
SCENE { display: list-item;
        list-style: none inside url(heart.gif) }
```

The whitespace Property

The `white-space` property determines how significant whitespace (spaces, tabs, line breaks) within an element is. The allowable values are:

`normal`

`pre`

`nowrap`

The default value, `normal`, simply means that runs of whitespace are condensed to a single space and words are wrapped to fit on the screen or page. This is the normal treatment of whitespace in both HTML and XML.

The `pre` value acts like the `PRE` (preformatted) element in HTML. All whitespace in the input document is considered significant and faithfully reproduced on the output device. It may be accompanied by a shift to a monospaced font. This would be useful for much computer source code or some poetry. Listing 12-11 is a poem, *The Altar*, by George Herbert in which spacing is important. In this poem, the lines form the shape of the poem's subject.

Listing 12-11: *The Altar* in XML

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="12-12.css"?>
<POEM>

<TITLE>The Altar</TITLE>
<POET>George Herbert</POET>

<VERSE>   A broken ALTAR, Lord, thy servant rears,</VERSE>
<VERSE>   Made of a heart, and cemented with tears:</VERSE>
<VERSE>    Whose parts are as thy hand did frame;</VERSE>
<VERSE>    No workman's tool hath touched the same.</VERSE>
<VERSE>    No workman's tool hath touched the same.</VERSE>
<VERSE>              A     HEART     alone</VERSE>
<VERSE>              Is   such   a   stone,</VERSE>
<VERSE>              As    nothing     but</VERSE>
<VERSE>              Thy  power  doth  cut.</VERSE>
<VERSE>              Wherefore  each  part</VERSE>
```

```
<VERSE>              Of  my   hard   heart</VERSE>
<VERSE>              Meets in  this  frame,</VERSE>
<VERSE>              To  praise  thy  name:</VERSE>
<VERSE>    That  if  I  chance  to  hold  my  peace,</VERSE>
<VERSE>    These stones to praise thee may not cease.</VERSE>
<VERSE>   O  let  thy  blessed   SACRIFICE   be   mine,</VERSE>
<VERSE>   And  sanctify   this   ALTAR   to  be  thine.</VERSE>

</POEM>
```

Listing 12-12 is a style sheet that uses `white-space: pre` to preserve this form. Figure 12-5 shows the result in Mozilla.

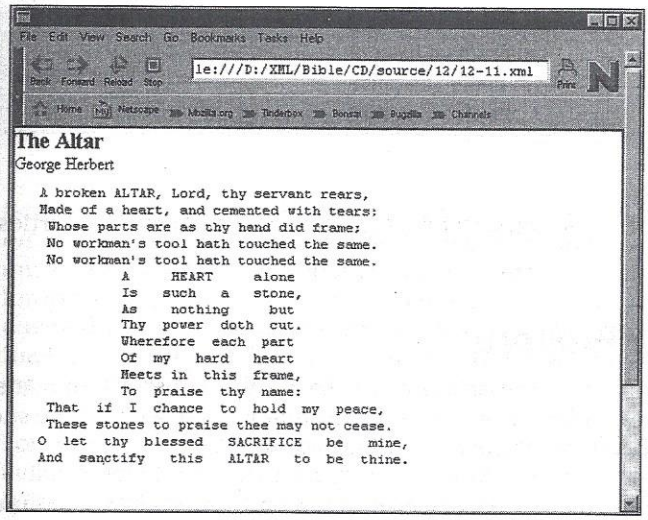


Figure 12-5: *The Altar* by George Herbert with white-space: pre

Internet Explorer 5.0 does not support `white-space: pre`.

Listing 12-12: A style sheet for whitespace-sensitive poetry

```
POEM   { display: block }
TITLE  { display: block; font-size: 16pt; font-weight: bold }
POET   { display: block; margin-bottom: 10px }
STANZA { display: block; margin-bottom: 10px }
VERSE  { display: block;
         white-space: pre; font-family: monospace }
```

Finally, the `nowrap` value is a compromise that breaks lines exactly where there's an explicit break in the source text, but condenses other runs of space to a single space. This might be useful when you're trying to faithfully reproduce the line breaks in a classical manuscript or some other poetry where the line breaks are significant but the space between words isn't.

Internet Explorer 5.0 and earlier do not properly support `nowrap`.

Font Properties

CSS Level 1 supports five basic font properties. These are:

1. `font-family`
2. `font-style`
3. `font-variant`
4. `font-weight`
5. `font-size`

Furthermore, there's a `font` shorthand property that can set all five properties at once.

The font-family Property

The value of the `font-family` property is a comma-separated list of font names such as Helvetica, Times, Palatino, etc. Font names that include whitespace such as "Times New Roman" should be enclosed in double quotes.

Names may also be one of the five generic names `serif`, `sans-serif`, `cursive`, `fantasy`, and `monospace`. The browser replaces these names with a font of the requested type installed on the local system. Table 12-4 demonstrates these fonts.

Table 12-4
Generic Fonts

Name	*Typical Family*	*Distinguishing Characteristic*	*Example*
Serif	Times, Times New Roman, Palatino	Curlicues on the edges of letters make serif text easier to read in small body type.	The quick brown fox jumped over the lazy dog.

Name	*Typical Family*	*Distinguishing Characteristic*	*Example*
sans-serif	Geneva, Helvetica, Verdana	Block type, often used in headlines.	The quick brown fox jumped over the lazy dog.
Monospace	Courier, Courier New, Monaco, American Typewriter	A typewriter like font in which each character has exactly the same width, commonly used for source code and email.	`The quick brown fox jumped over the lazy dog.`
Cursive	ZapfChancery	Script font, a simulation of handwriting.	*The quick brown fox jumped over the lazy dog.*
Fantasy	Western, Critter	Text with special effects; e.g. letters on fire, letters formed by tumbling acrobats, letters made from animals, etc.	THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

Because there isn't a guarantee that any given font will be available or appropriate on a particular client system (10-point Times is practically illegible on a Macintosh, much less a Palm Pilot), you'll generally provide a comma-separated list of choices for the font in the order of preference. The last choice in the list should always be one of the generic names. However, even if you don't specify a generic name and the fonts you *do* specify aren't available, the browser will pick something. It just may not be anything like what you wanted.

For example, here are two rules that make the `TITLE` element Helvetica with a fallback position of any sans serif font; and the rest of the elements Times with fallback positions of Times New Roman, and any serif font.

```
TITLE    { font-family: Helvetica, sans-serif }
SYNOPSIS { font-family: Times, "Times New Roman", serif }
```

Figure 12-6 shows the synopsis loaded into Mozilla 5.0 after these two rules are added to the style sheet of Listing 12-8. Not a great deal has changed since Figure 12-4 Times is commonly the default font. The most obvious difference is that the title is now in Helvetica.

The `font-family` property is inherited by child elements. Thus by setting `SYNOPSIS`'s `font-family` to Times, all the child elements are also set to Times except for `TITLE` whose own `font-family` property overrides the one it inherits.

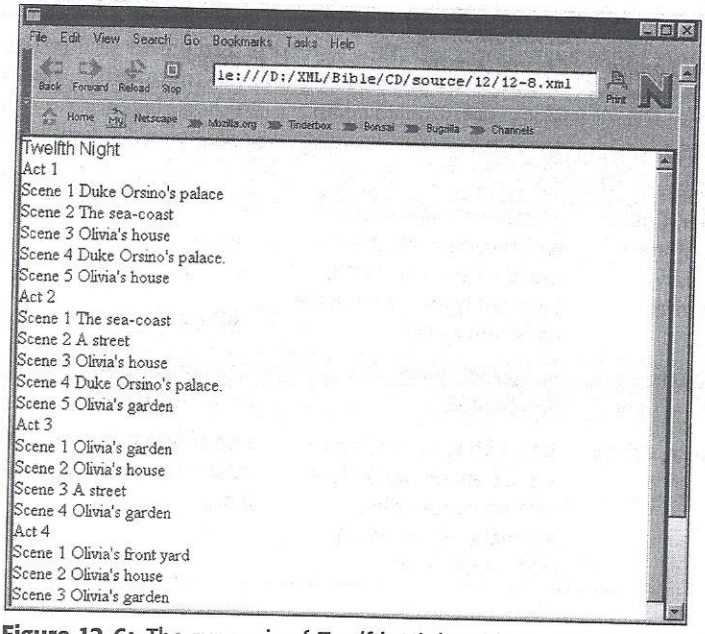


Figure 12-6: The synopsis of *Twelfth Night* with the title in Helvetica

The font-style Property

The `font-style` property has three values: `normal`, `italic`, and `oblique`. The regular text you're reading now is normal. The typical rendering of the HTML `EM` element is *italicized*. Oblique text is very similar to italicized text. However, oblique text is most commonly created by a computer following a simple algorithm to slant normal text by a fixed amount. Italicized text generally uses a font hand designed to look good in its slanted form.

This rule italicizes the `SCENE_NUMBER`:

```
SCENE_NUMBER { font-style: italic}
```

Figure 12-7 shows the synopsis loaded into Internet Explorer 5.0 after this rule is added to the style sheet for the synopsis.

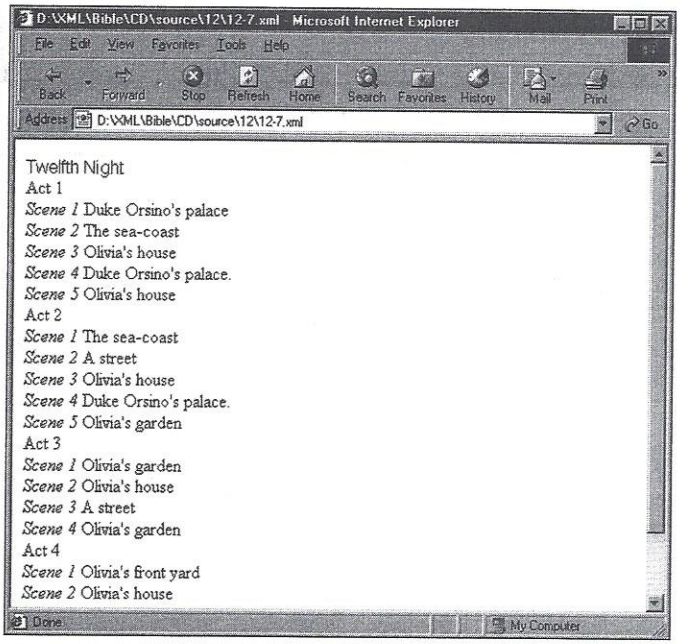


Figure 12-7: The synopsis of *Twelfth Night* with italic scene numbers

The font-variant Property

The `font-variant` property has two possible values in CSS Level 1, `normal` and `small-caps`. The default is normal. Setting `font-variant` to `small-caps` replaces lowercase letters with capital letters in a smaller font size than the main body text.

You can get a very nice effect by combining the `font-variant` property with the `first-letter` pseudo-element. For example, define the `ACT_NUMBER` element to have the `font-variant: small-caps`. Next define the first letter of `ACT_NUMBER` to have `font-variant: normal`. This produces act numbers that look like this:

ACT 1

Here are the rules:

```
ACT_NUMBER              { font-variant: small-caps}
ACT_NUMBER:first-letter { font-variant: normal}
```

The second rule overrides the first, but only for the first letter of the act number.

The font-weight Property

The `font-weight` property determines how dark (bold) or light (narrow) the text appears. There are 13 possible values of this property:

```
normal
bold
bolder
lighter
100
200
300
400
500
600
700
800
900
```

Weights range from 100 (the lightest) to 900 (the darkest). Intermediate, non-century values like 850 are not allowed. Normal weight is 400. Bold is 700. The `bolder` value makes an element bolder than its parent. The `lighter` value makes an element less bold than its parent. However, there's no guarantee that a particular font has as many as nine separate levels of boldness.

Here's a simple rule that makes the `TITLE` and `ACT_NUMBER` elements bold.

```
TITLE, ACT_NUMBER { font-weight: bold}
```

Figure 12-8 shows the results in the Mozilla viewer after this rule is added to the style sheet for Listing 12-7.

The font-size Property

The `font-size` property determines the height and the width of a typical character in the font. Larger sizes take up more space on the screen. The size may be specified as a keyword, a value relative to the font size of the parent, a percentage of the size of the parent element's font size, or an absolute number.

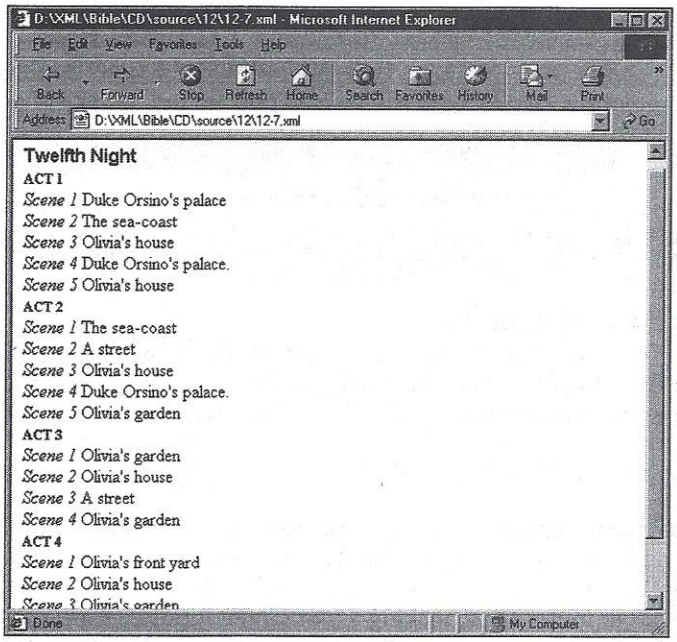


Figure 12-8: The synopsis of *Twelfth Night* with bold title and act numbers

Keyword

Absolute size keywords are:

```
xx-small
x-small
small
medium
large
x-large
xx-large
```

These keywords are the preferred way to set font sizes because they are still relative to the base font size of the page. For instance, if the user has adjusted their default font size to 20 points because they're very near-sighted, all other values here will scale accordingly.

In CSS1, each size is 1.5 times larger than the next smallest size. The default is `medium`, so if a browser's default is 12 points, then `large` type will be 18 points, `x-large` type will be 27 points, and `xx-large` type will be 40.5 points. By contrast, `small` type will be 8 points; `x-small` type will be 5.33 points, and `xx-small` will be an almost certainly illegible 3.56 points.

Here's the simple rule that makes the `TITLE` extra large:

```
TITLE   { font-size: x-large }
```

Value Relative to Parent's Font Size

You can also specify the size relative to the parent element as either `larger` or `smaller`. For instance, in the following, the `SCENE_NUMBER` will have a size that is smaller than the font size of its parent `SCENE`.

```
SCENE_NUMBER { font-size: smaller }
```

There's no definitive rule for exactly how much smaller a smaller font will be or how much larger a larger font will be. Generally, the browser will attempt to move from medium to small, from small to x-small and so forth. The same is true (in the other direction) for larger fonts. Thus, making a font larger should increase its size by about 50 percent, and making a font smaller should decrease its size by about 33 percent, but browsers are free to fudge these values in order to match the available font sizes.

Percentage of Parent Element's Font Size

If these options aren't precise enough, you can make finer adjustments by using a percentage of the parent element's font size. For example, this rule says that the font used for a `SCENE_NUMBER` is 50% of the size of the font for the `SCENE`.

```
SCENE_NUMBER { font-size: 50% }
```

Absolute Length Value

Finally, you can give a font size as an absolute length. Although you can use pixels, centimeters, or inches, the most common unit when measuring fonts is points. For example, this rule sets the default `font-size` for the `SYNOPSIS` element and its children to 14 points.

```
SYNOPSIS { font-size: 14pt }
```

I strongly urge you not to use absolute units to describe font sizes. It's extremely difficult (I'd argue impossible) to pick a font size that's legible across all the different platforms on which your page might be viewed, ranging from PDAs to the Sony Jumbotron in Times Square. Even when restricting themselves to standard personal computers, most designers usually pick a font that's too small. Any text that's intended to be read on the screen should be at least 12 points, possibly more.

Figure 12-9 shows the results in Mozilla after these rules are added to the style sheet for Listing 12-7. The text of the scenes is not really bolder. It's just bigger. In any case, it's a lot easier to read.

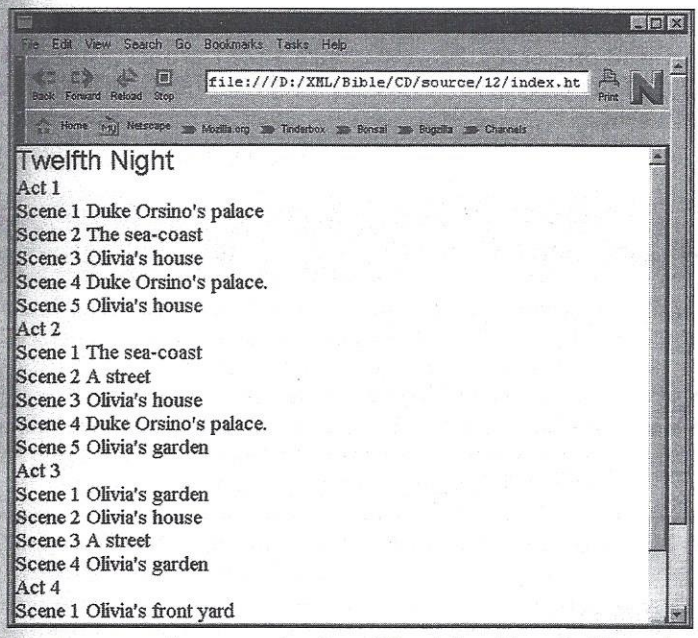


Figure 12-09: The synopsis of *Twelfth Night* with varied font sizes

The font Shorthand Property

The `font` property is a shorthand property that allows the font style, variant, weight, size, and family to be set with one rule. For example, here are two rules for the `TITLE` and `SCENE_NUMBER` elements that combine the six separate rules of the previous section:

```
TITLE { font: bold x-large Helvetica, sans-serif }
SCENE_NUMBER { font: italic smaller serif }
```

Values must be given in the following order:

1. One each of style, variant, and weight, in any order, any of which may be omitted
2. Size, which may not be omitted
3. Optionally a forward slash (/) and a line height
4. Family, which may not be omitted

If this sounds complicated and hard to remember, that's because it is. I certainly can't remember the exact details for the order of these properties without looking it up. I prefer to just use the individual properties one at a time. It's questionable whether shorthand properties like this really save any time.

Listing 12-13 is the style sheet for the synopsis with all the rules devised so far, using the `font` shorthand properties. However, since a `font` property is exactly equivalent to the sum of the individual properties it represents, there's no change to the rendered document.

Listing 12-13: A style sheet for the synopsis with font shorthand

```
SYNOPSIS, TITLE, ACT, SCENE { display: block }
SCENENUMBER { font: italic smaller serif }
TITLE { font: bold x-large Helvetica, sans-serif }
SYNOPSIS { font: 14pt Times, "Times New Roman", serif }
ACTNUMBER { font-variant: small-caps}
ACTNUMBER:first-letter { font-variant: normal}
ACTNUMBER { font-weight: bold}
```

The Color Property

CSS allows you to assign colors to almost any element on a page with the `color` property. The value of the color property may be one of 16 named color keywords, or an RGB triple in decimal, hexadecimal, or percentages. For instance, the following rules specify that all elements on the page are colored black except the `SCENE_NUMBER`, which is colored blue:

```
SYNOPSIS { color: black }
SCENE_NUMBER { color: blue}
```

The `color` property is inherited by children. Thus, all elements in the synopsis except for the `SCENE_NUMBER` elements will be colored black.

The following rules are all equivalent to the above two. I recommend using named colors when possible, and browser-safe colors when not.

```
SYNOPSIS { color: #000000 }
SCENE_NUMBER { color: #0000FF}
SYNOPSIS { color: rgb(0, 0, 0) }
SCENE_NUMBER { color: rgb(0, 0, 255)}
SYNOPSIS { color: rgb(0%, 0%, 0%) }
SCENE_NUMBER { color: rgb(0%, 0%, 100%)}
```

Background Properties

The background of an element can be set to a color or an image. If it's set to an image, the image can be positioned differently relative to the content of the element. This is accomplished with the following five basic properties:

1. `background-color`
2. `background-image`
3. `background-repeat`
4. `background-attachment`
5. `background-position`

Finally, there's a `background` shorthand property that allows you to set some or all of these five properties in one rule.

Fancy backgrounds are vastly overused on the Web today. Anything other than a very light background color only makes your page harder to read and annoys users. I list these properties here for completeness' sake, but I strongly recommend that you use them sparingly, if at all.

None of the background properties is inherited. Each child element must specify the background it wants. However, it may appear as if background properties are inherited because the default is for the background to be transparent. The background of whatever element is drawn below an element will show through. Most of the time this is the background of the parent element.

The background-color Property

The `background-color` property may be set to the same values as the `color` property. However, rather than changing the color of the element's contents, it changes the color of the element's background on top of which the contents are drawn. For example, to draw a `SIGN` element with yellow text on a blue background, you would use this rule:

```
SIGN { color: yellow; background-color: blue}
```

You can also set the `background-color` to the keyword `transparent` (the default) which simply means that the background takes on the color or image of whatever the element is laying on top of, generally the parent element.

The background-image Property

The `background-image` property is either `none` (the default) or a URL (generally relative) where a bitmapped image file can be found. If it's a URL, then the browser will load the image and use it as the background, much like the `BACKGROUND`

attribute of the BODY element in HTML. For example, here's how you attach the file party.gif (shown in Figure 12-10) as the background for an INVITATION element.

```
INVITATION { background-image: url(party.gif) }
```

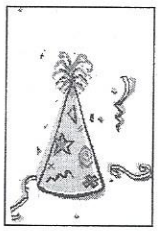


Figure 12-10: The original, untiled, uncropped background image for the party invitation in Listing 12-14

The image referenced by the background-image property is drawn underneath the specified element, *not* underneath the browser pane like the BACKGROUND attribute of HTML's BODY element.

Note

If the background image is associated with the root element, early betas of Mozilla 5.0 attach the background image to the entire document pane rather than to only the element itself. For all non-root elements, however, the background image applies only to the element it's applied to. The CSS Level 1 specification is not clear regarding whether or not this is acceptable behavior.

Background images will generally not be the exact same size as the contents of the page. If the image is larger than the element's box, the image will be cropped. If the image is smaller than the element's box, it will be tiled vertically and horizontally. Figure 12-11 shows a background image that has tiled exactly far enough to cover the underlying content. Note that the tiling takes place across the element, *not* across the browser window. The XML file for this picture is in Listing 12-14.

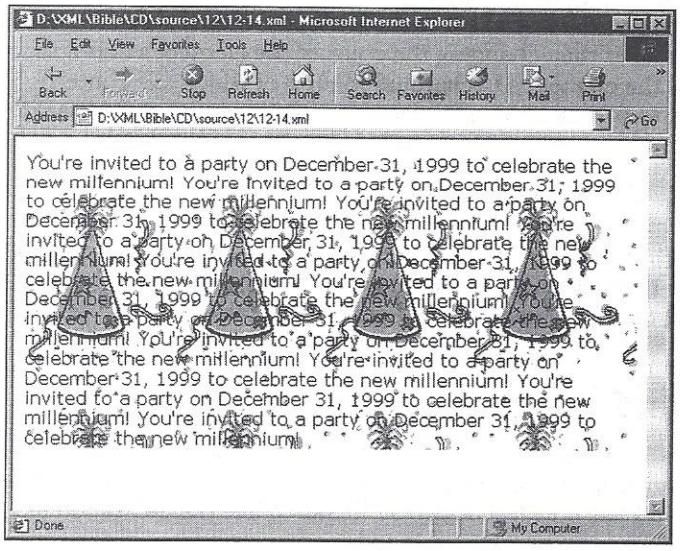


Figure 12-11: A tiled background image

Listing 12-14: A party invitation in XML

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="party.css"?>
<INVITATION>
You're invited to a party on December 31, 1999 to celebrate the
new millennium! You're invited to a party on December 31, 1999
to celebrate the new millennium! You're invited to a party on
December 31, 1999 to celebrate the new millennium! You're
invited to a party on December 31, 1999 to celebrate the new
millennium! You're invited to a party on December 31, 1999 to
celebrate the new millennium! You're invited to a party on
December 31, 1999 to celebrate the new millennium! You're
invited to a party on December 31, 1999 to celebrate the new
millennium! You're invited to a party on December 31, 1999 to
celebrate the new millennium! You're invited to a party on
December 31, 1999 to celebrate the new millennium! You're
invited to a party on December 31, 1999 to celebrate the new
millennium! You're invited to a party on December 31, 1999 to
celebrate the new millennium!
</INVITATION>
```

The background-repeat Property

The `background-repeat` property adjusts how background images are tiled across the screen. You can specify that background images are not tiled or are only tiled horizontally or vertically. Possible values for this property are:

```
repeat
repeat-x
repeat-y
no-repeat
```

For example, to only show a single party hat on the invitation you would set the `background-repeat` of the `INVITATION` element to `no-repeat`. Figure 12-12 shows the result. For example:

```
INVITATION { background-image:  url(party.gif);
             background-repeat: no-repeat }
```

To tile across but not down the page, set `background-repeat` to `repeat-x`, as shown below. Figure 12-13 shows the background image tiled across but not down.

```
INVITATION { background-image:  url(party.gif);
             background-repeat: repeat-x }
```

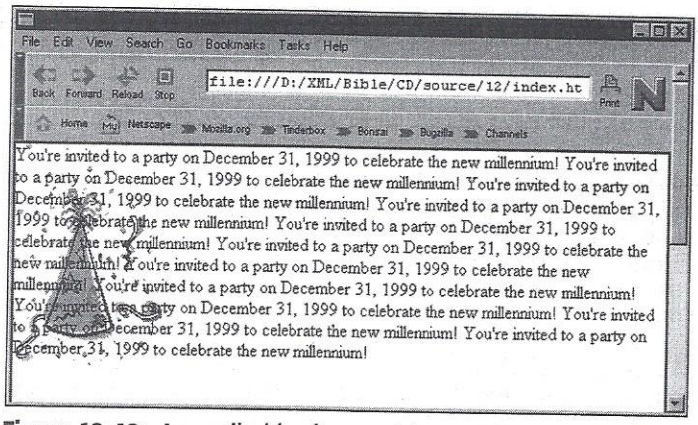


Figure 12-12: An untiled background image

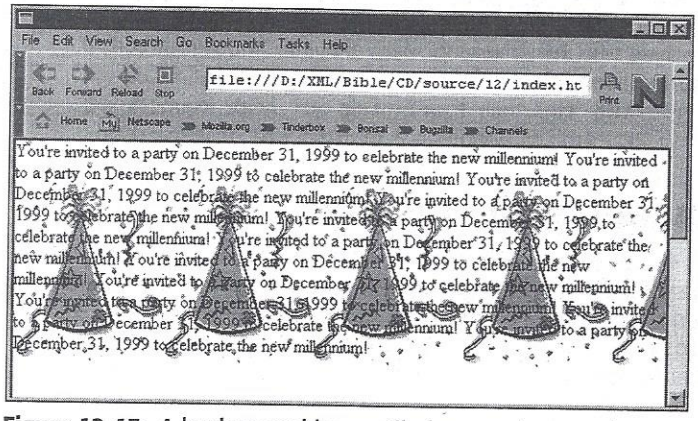


Figure 12-13: A background image tiled across, but not down

To tile down but not across the page, set `background-repeat` to `repeat-y`, as shown below. Figure 12-14 shows the result.

```
INVITATION { background-image:  url(party.gif);
             background-repeat: repeat-y }
```

The background-attachment Property

In HTML, the background image is attached to the document. When the document is scrolled, the background image scrolls with it. With the `background-attachment` property, you can specify that the background be attached to the window or pane instead. Possible values are `scroll` and `fixed`. The default is `scroll`; that is, the background is attached to the document rather than the window.

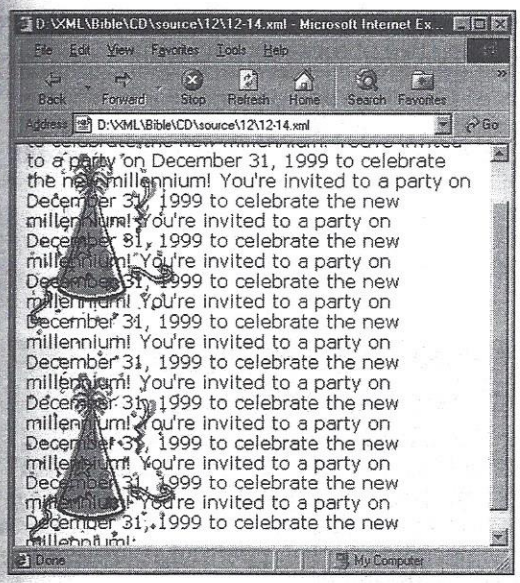


Figure 12-14: A background image tiled down but not across

However, with `background-attachment` set to `fixed`, the document scrolls but the background image doesn't. This might be useful in conjunction with an image that's big enough for a typical browser window but not big enough to be a backdrop for a large document when you don't want to tile the image. You would code this:

```
DOCUMENT { background-attachment: fixed;
           background-repeat: no-repeat }
```

Neither IE5 nor Mozilla supports fixed background images. This feature may be added in later releases. (The CSS1 spec does not require browsers to support fixed backgrounds.)

The background-position Property

By default, the upper-left corner of a background image is aligned with the upper-left corner of the element it's attached to. (See Figure 12-12 for an example.) Most of the time this is exactly what you want. However, for those rare times when you want something else, the `background-position` property allows you to move the background relative to the element.

You can specify the offset using percentages of the width and height of the parent element, using absolute lengths, or using two of these six keywords:

```
top
center
bottom
left
center
right
```

Percentages of Parent Element's Width and Height

Percentages enable you to pin different parts of the background to the corresponding part of the element. The *x* coordinate is given as a percentage ranging from 0% (left-hand side) to 100% (right-hand side). The *y* coordinate is given as a percentage ranging from 0% (top) to 100% (bottom). For example, this rule places the upper-right corner of the image in the upper-right corner of the `INVITATION` element. Figure 12-15 shows the result.

```
INVITATION { background-image: url(party.gif);
             background-repeat: no-repeat;
             background-position: 100% 0% }
```

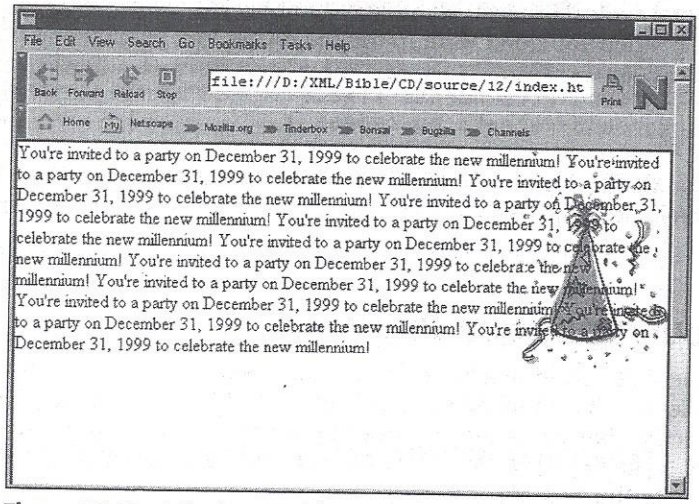


Figure 12-15: A background image aligned with the upper-right corner of the content

Absolute Lengths

Absolute lengths position the upper-left corner of the background in an absolute position in the element. The next rule places the upper-left corner of the background image party.gif one centimeter to the right and two centimeters below the upper-left corner of the element. Figure 12-16 shows the result.

```
INVITATION { background-image: url(party.gif);
             background-repeat: no-repeat;
             background-position: 1cm 2cm }
```

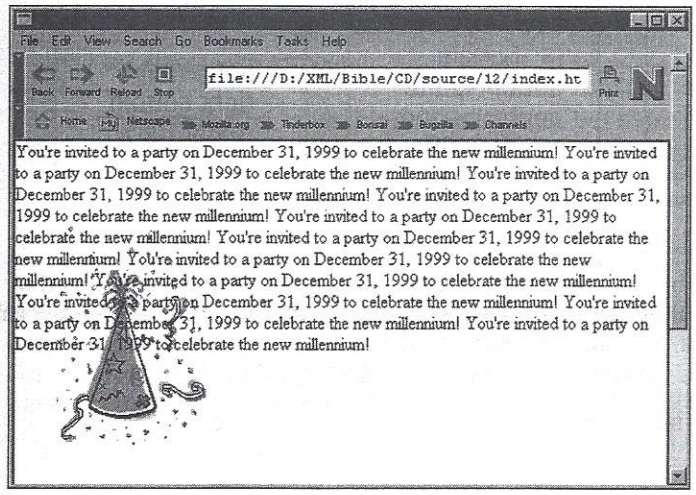


Figure 12-16: A background image 1.0 cm to the right and 2.0 cm below the left-hand corner of the element

Keywords

The `top left` and `left top` keywords are the same as 0% 0%. The `top`, `top center`, and `center top` are the same as 50% 0%. The `right top` and `top right` keywords are the same as 100% 0%. The `left`, `left center`, and `center left` keywords are the same as 0% 50%. The `center` and `center center` keywords are the same as 50% 50%. The `right`, `right center`, and `center right` keywords are the same as 100% 50%. The `bottom left` and `left bottom` keywords are the same as 0% 100%. The `bottom`, `bottom center`, and `center bottom` mean the same as 50% 100%. The `bottom right` and `right bottom` keywords are the same as 100% 100%. Figure 12-17 associates these with individual positions on an element box.

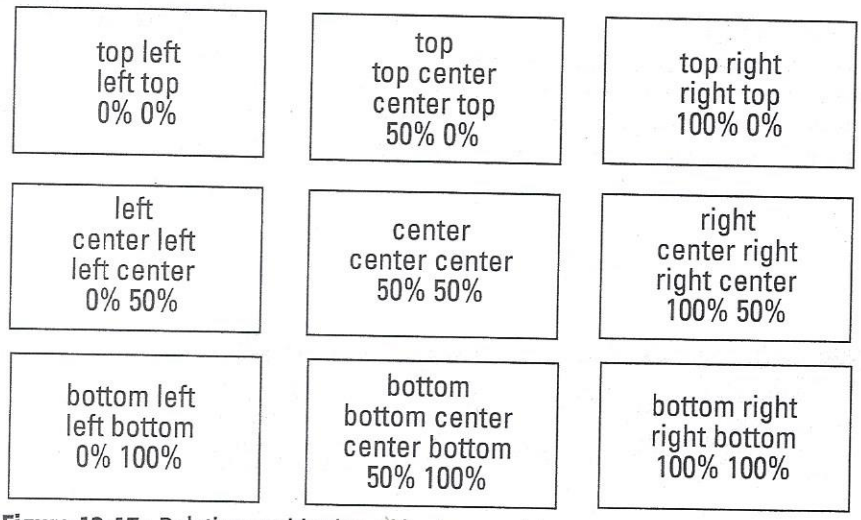


Figure 12-17: Relative positioning of background images

For instance, for our running invitation example, the best effect is achieved by pinning the centers together, as shown in Figure 12-18. Here's the necessary rule:

```
INVITATION { background-image: url(party.gif);
             background-repeat: no-repeat;
             background-position: center center }
```

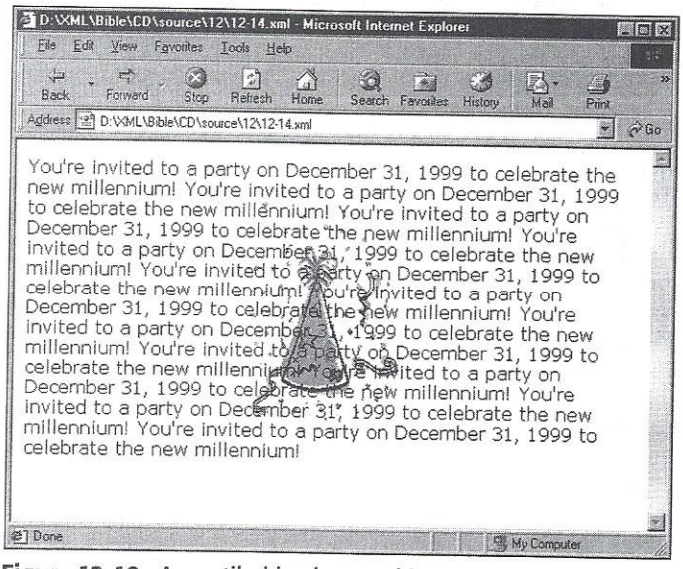


Figure 12-18: An untiled background image pinned to the center of the INVITATION element

If the background-attachment property has the value fixed, then the image is placed relative to the windowpane instead of the element.

The Background Shorthand Property

The background property is shorthand for setting the background-color, background-image, background-repeat, background-attachment, and background-position properties in a single rule. For example, to set background-color to white, background-image to party.gif, background-repeat to no-repeat, and background-attachment to fixed in the INVITATION element, you can use this rule:

```
INVITATION { background: url(party.gif) white no-repeat fixed }
```

This means exactly the same thing as this longer but more legible rule:

```
INVITATION { background-image: url(party.gif);
             background-color: white;
             background-repeat: no-repeat;
             background-attachment: fixed }
```

When using the background shorthand property, values for any or all of the five properties may be given in any order. However, none may occur more than once. For example, the upper-right corner alignment rule used for Figure 12-16 could have been written like this instead:

```
INVITATION { background: url(party.gif) no-repeat 100% 0% }
```

Text Properties

There are eight properties affecting the appearance of text, irrespective of font. These are:

1. word-spacing
2. letter-spacing
3. text-decoration
4. vertical-align
5. text-transform
6. text-align
7. text-indent
8. line-height

The word-spacing Property

The `word-spacing` property expands the text by adding additional space between words. A negative value removes space between words. The only reason I can think of to alter the word spacing on a Web page is if you are a student laboring under tight page-count limits who wants to make a paper look bigger or smaller than it is.

Desktop publishers love adjusting these kinds of properties. The problem is that all the rules they've learned about how and when to adjust spacing are based on ink on paper, and really don't work when transferred to the medium of electrons on phosphorus (a typical CRT monitor). You're almost always better off letting the browser make decisions about word and letter spacing for you.

If, on the other hand, your target medium *is* ink on paper, then there's a little more to be gained by adjusting these properties. The main difference is that with ink on paper you control the delivery medium. You know exactly how big the fonts are, how wide and high the display is, how many dots per inch are being used, and so forth. On the Web, you simply don't have enough information about the output medium available to control everything at this level of detail.

To change this from the default value of `normal`, you set a length for the property. For example,

```
INVITATION { word-spacing: 1em }
```

Browsers are not required to respect this property, especially if it interferes with other properties like `align: justified`. Internet Explorer 5.0 does not support `word-spacing`, but Mozilla does, as shown in Figure 12-19.

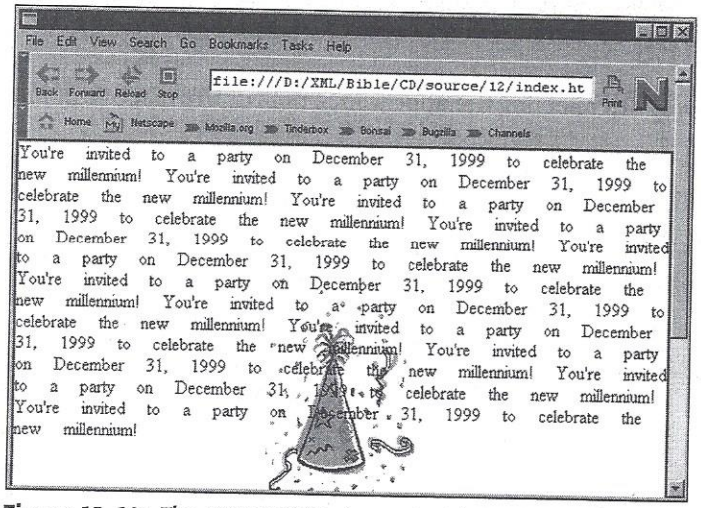


Figure 12-19: The INVITATION element with one em of word spacing

The letter-spacing Property

The `letter-spacing` property enables you to expand the text by adding additional space between letters. You can make the value negative to remove space between letters. Again, the only reason I can think of to do this on a Web page is to make a paper look bigger or smaller than it really is to meet a length requirement.

To change this from the default value of `normal`, you set a length for the property. For example:

```
INVITATION { letter-spacing: 0.3em }
```

Since justification works by adjusting the amount of space between letters, changing the letter spacing manually prevents the browser from justifying text.

Browsers are not required to respect this property, especially if it interferes with other properties like `align: justified`. However both Internet Explorer and Mozilla do, as shown in Figure 12-20.

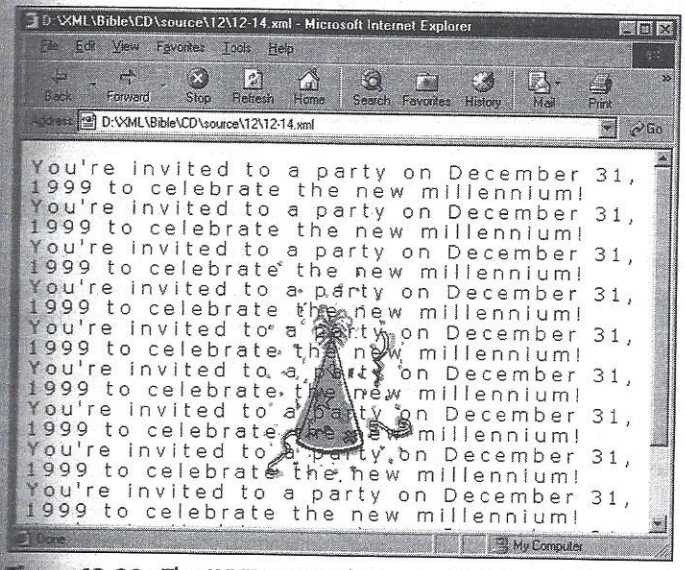


Figure 12-20: The INVITATION element with 0.3em letter spacing

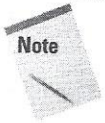
The text-decoration Property

The text-decoration property may have one of the following five values:

```
none
underline
```

```
overline
line-through
blink
```

Except for `none`, the default, these values are not mutually exclusive. You may for example, specify that a paragraph is underlined, overlined, struck through, and blinking. (I do not, however, recommend that you do this.)



Browsers are not required to support blinking text. This is a good thing.

For example, the next rule specifies that `CHARACTER` elements are underlined. Figure 12-21 shows the result of applying this rule to the synopsis of *Twelfth Night* in Listing 12-7.

```
CHARACTER { text-decoration: underline }
```

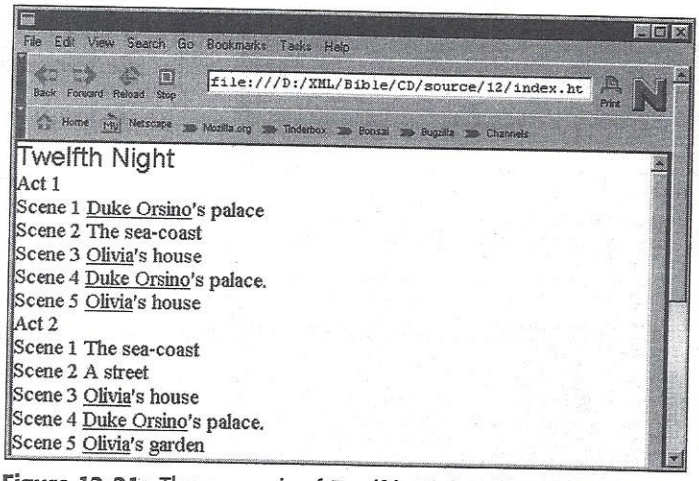


Figure 12-21: The synopsis of *Twelfth Night* with underlined characters

The vertical-align Property

The `vertical-align` property specifies how an inline element is positioned relative to the baseline of the text. Valid values are:

```
baseline
sub
super
top
```

```
text-top
middle
bottom
text-bottom
```

You can also use a percentage of the line height of the element. The default is `baseline` which lines up the baseline of the element with the baseline of its parent.

The `sub` value makes the element a subscript. The `super` value makes the element a superscript. The `text-top` value aligns the top of the element with the top of the parent element's font. The `middle` value aligns the vertical midpoint of the element with the baseline of the parent plus half the *x*-height. The `text-bottom` value aligns the bottom of the element with the bottom of the parent element's font.

The `top` value aligns the top of the element with the tallest letter or element on the line. The `bottom` value aligns the bottom of the element with the bottom of the lowest letter or element on the line. The exact alignment changes as the height of the tallest or lowest letter changes.

For example, the rule for a footnote number might look like this one that superscripts the number and decreases its size by 20 percent.

```
FOOTNOTE_NUMBER { vertical-align: super; font-size: 80% }
```

The text-transform Property

The `text-transform` property lets you to specify that the text should be rendered in all uppercase, all lowercase, or with initial letters capitalized. This is useful in headlines, for example. The valid values are:

```
capitalize
uppercase
lowercase
none
```

Capitalization Makes Only The First Letter Of Every Word Uppercase Like This Sentence. PLACING THE SENTENCE IN UPPERCASE, HOWEVER, MAKES EVERY LETTER IN THE SENTENCE UPPERCASE. The following rule converts the `TITLE` element in the *Twelfth Night* synopsis to uppercase. Figure 12-22 shows the synopsis after this rule has been applied.

```
TITLE { text-transform: uppercase }
```

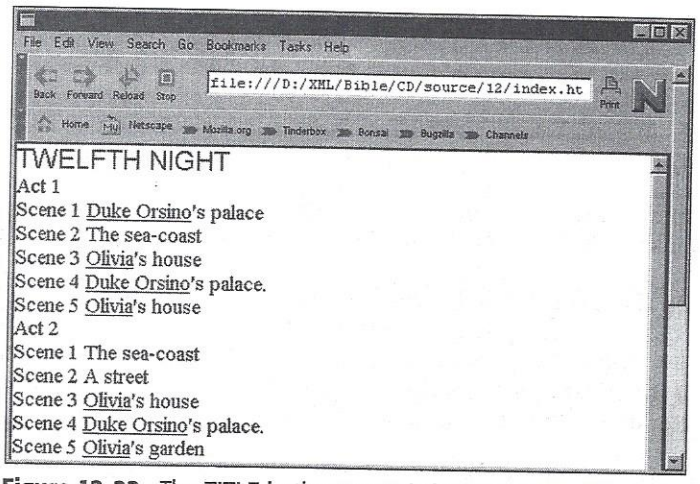


Figure 12-22: The TITLE in the synopsis is now uppercase.

The `text-transform` property is somewhat language-dependent since many languages—Chinese, for example—don't have any concept of an upper- and a lowercase.

The text-align Property

The `text-align` property applies only to block-level elements. It specifies whether the text in the block is to be aligned with the left-hand side, the right-hand side, centered, or justified. The valid values are:

```
left
right
center
justify
```

The following rules center the `TITLE` element in the *Twelfth Night* synopsis and justifies everything else. Figure 12-23 shows the synopsis after these rules have been applied.

```
TITLE { text-align: center }
SYNOPSIS { text-align: justify }
```

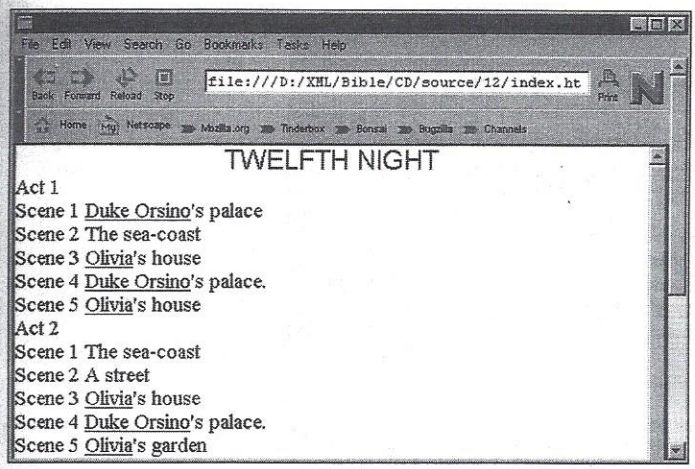


Figure 12-23: The TITLE in the synopsis is centered and the rest of the text is justified.

The text-indent Property

The `text-indent` property, which applies only to block-level elements, specifies the indentation of the first line of a block with respect to the remaining lines of the block and is given as either an absolute length or a percentage of the width of the parent element. The value may be negative which produces a hanging indent.

To indent all the lines of an element, rather than just the first, you use the box properties discussed in the next section to set an extra left margin on the element.

For example, the following rule indents the scenes in the synopsis by half an inch. Figure 12-24 shows the synopsis after this rule has been applied.

```
SCENE { text-indent: 0.5in }
```

The line-height Property

The `line-height` property specifies the distance between the baselines of successive lines. It can be given as an absolute number, an absolute length, or a percentage of the font size. For instance, the following rule double-spaces the `SYNOPSIS` element. Figure 12-25 shows the *Twelfth Night* synopsis after this rule has been applied.

```
SYNOPSIS { line-height: 200% }
```

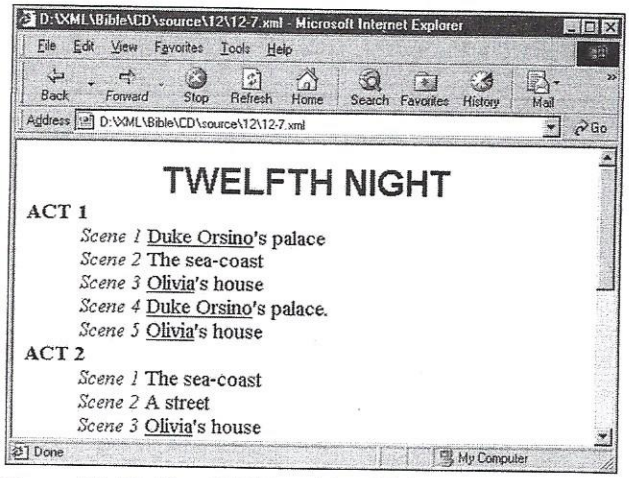


Figure 12-24: The SCENE and its children in the synopsis are all indented half an inch.

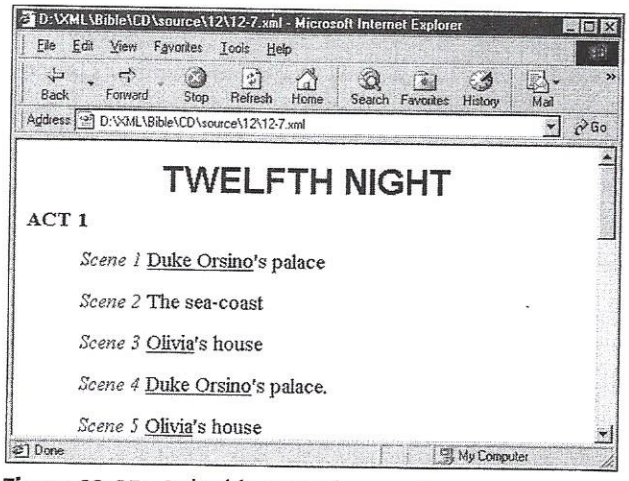


Figure 12-25: A double-spaced synopsis

Double-spacing isn't particularly attractive though so I'll remove it. In the next section, some extra margins are added around individual elements to get a nicer effect. Listing 12-15 summarizes the additions made in this section to the synopsis style sheet (minus the double-spacing).

Listing 12-15: The synopsis style sheet with text properties

```
SYNOPSIS, TITLE, ACT, SCENE { display: block }
SCENE_NUMBER { font: italic smaller serif }
TITLE { font: bold x-large Helvetica, sans-serif }
SYNOPSIS { font: 14pt Times, "Times New Roman", serif }
ACT_NUMBER              { font-variant: small-caps}
ACT_NUMBER:first-letter { font-variant: normal}
ACT_NUMBER { font-weight: bold}
CHARACTER { text-decoration: underline }
TITLE { text-transform: uppercase }
TITLE { text-align: center }
SYNOPSIS { text-align: justify }
SCENE { text-indent: 0.5in }
```

Box Properties

CSS describes a two-dimensional canvas on which output is drawn. The elements drawn on this canvas are encased in imaginary rectangles called boxes. These boxes are always oriented parallel to the edges of the canvas. Box properties enable you to specify the width, height, margins, padding, borders, sizes, and positions of the individual boxes. Figure 12-26 shows how these properties relate to each other.

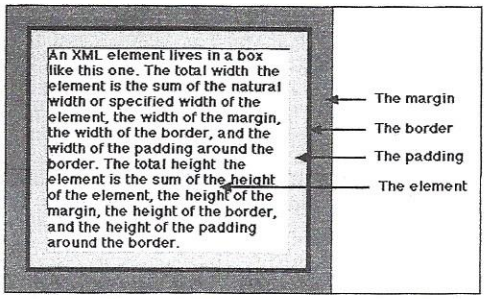


Figure 12-26: A CSS box with margin, border, and padding

Margin Properties

Margin properties specify the amount of space added to the box outside its border. This may be set separately for the top, bottom, right and left margins using the `margin-top`, `margin-bottom`, `margin-right`, and `margin-left` properties. Each margin may be given as an absolute length or as a percentage of the size of the parent element's width. For example, you can add a little extra space between each `ACT` element and the preceding element by setting `ACT`'s `margin-top` property to 1ex as the following rule and Figure 12-27 demonstrate.

```
ACT { margin-top: 1ex }
```

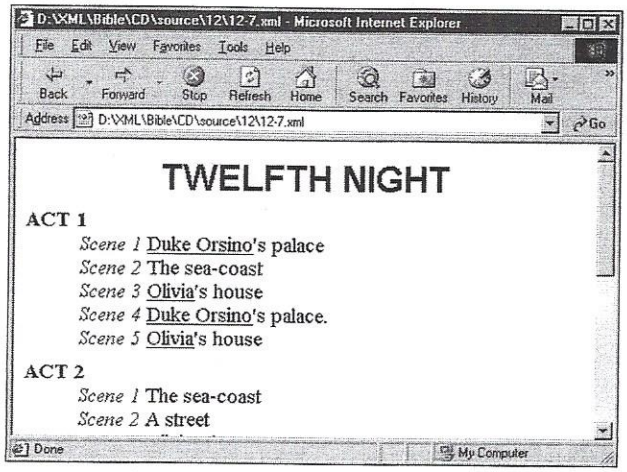


Figure 12-27: The top margin of the ACT element is larger

You can also set all four margins at once using the shorthand `margin` property. For example, you can add extra whitespace around the entire *Twelfth Night* document by setting the margin property for the root-level element (`SYNOPSIS` in this example) as shown in the next rule. Figure 12-28 demonstrates.

```
SYNOPSIS { margin: 1cm 1cm 1cm 1cm }
```

In fact, this is the same as using a single value for margin, which CSS interprets as being applicable to all four sides.

```
SYNOPSIS { margin: 1cm }
```

Given two `margin` values, the first applies to top and bottom, the second to right and left. Given three `margin` values, the first applies to the top, the second to the right and left, and the third to the bottom. It's probably easier to just use the separate `margin-top`, `margin-bottom`, `margin-right`, and `margin-left` properties.

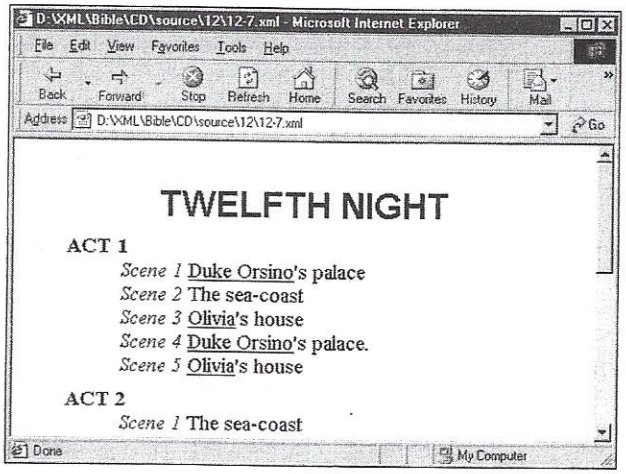


Figure 12-28: One centimeter of whitespace around the entire synopsis

Border Properties

Most boxes won't have borders. They are imaginary boxes that affect the layout of their contents, but are probably not seen as boxes by the readers. However, you can make a box visible by drawing lines around it using the border properties. Border properties let you to specify the style, width, and color of the border.

Border Style

By default, no border is drawn around boxes regardless of the width and color of the border. To make a border visible you must change the `border-style` property of the box from its default value of `none` to one of these values:

```
dotted
dashed
solid
double
groove
ridge
inset
outset
```

The `border-style` property can have between one and four values. As with the `margin` property, a single value applies to all four borders. Two values set top and bottom borders to the first style, right and left borders to the second style. Three

values set the top, right and left, and bottom border styles in that order. Four values set each border in the order top, right, bottom, and left. For example, the next rule surrounds the entire SYNOPSIS with a solid border. Figure 12-29 shows the result in Internet Explorer 5.0. In this case, the border has the secondary effect of making the margin more obvious. (Remember, the margin is outside the border.)

```
SYNOPSIS { border-style: solid }
```

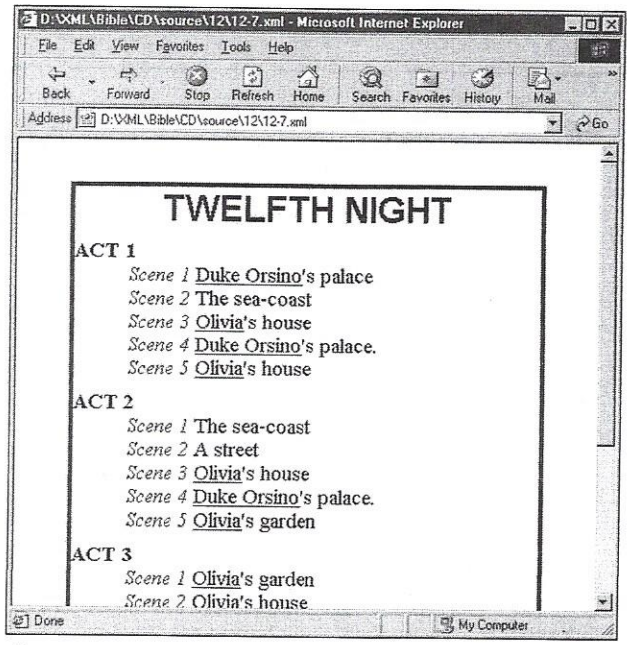


Figure 12-29: A border around the synopsis

Internet Explorer 5.0 can only display solid borders. The other styles are all drawn as simple, solid borders.

Border Width

There are four border-width properties for specifying the width of the borderline along the top, bottom, right, and left edges of the box. These are:

1. `border-top-width`
2. `border-right-width`
3. `border-bottom-width`
4. `border-left-width`

Each may be specified as an absolute length or as one of three keywords: `thin`, `medium`, or `thick`. Border widths cannot be negative, but can be zero.

For example, to enclose the `SYNOPSIS` element in a one-pixel wide solid border (the thinnest border any computer monitor can display), you could use the next rule to set these four properties:

```
SYNOPSIS { border-style:        solid;
           border-top-width:    1px;
           border-right-width:  1px;
           border-bottom-width: 1px;
           border-left-width:   1px }
```

If want to set all or several borders to the same width, it's most convenient to use the `border-width` shorthand property. This property can have between one and four values. One value sets all four border widths. Two values set top and bottom borders to the first value, right and left borders to the second value. Three values set the top, right, and left, and bottom widths in that order. Four values set each border in the order top, right, bottom, and left. For example, the following is equivalent to the previous rule:

```
SYNOPSIS { border-style: solid; border-width: 1px }
```

Border Color

The `border-color` property sets the color of between one and four borders. A single value sets all four border colors. Two values set top and bottom borders to the first color, right and left borders to the second color. Three values set the top, right and left, and bottom border colors in that order. Four values set each border in the order top, right, bottom, and left. Valid values are any recognized color name or RGB triplet. For example, to enclose the `SYNOPSIS` element in a one-pixel wide, solid red border, you'd use this rule:

```
SYNOPSIS { border-style: solid;
           border-width: 1px;
           border-color: red }
```

Since this book is printed in black and white, I'll spare you the picture.

Shorthand Border Properties

Five shorthand border properties let you set the width, style, and color of a border with one rule. These are:

1. `border-top`
2. `border-right`
3. `border-bottom`
4. `border-left`
5. `border`