CSS Property Reference

Mastering Cascading Style Sheets involves knowing how to use a large number of CSS properties that control the appearance of text, images, tables, and forms. To help you in your quest, this appendix gives you a summary of the properties and values you'll use to create your own styles. This list covers nearly all of the CSS 2.1 standard propertiesthe ones that most Web browsers support.

Note: This appendix leaves out properties that no (or hardly any) browsers recognize. Otherwise, the following descriptions mention the browsers with which each property works. For full details straight from the horse's mouth, visit the World Wide Web Consortium's CSS 2.1 specification at www.w3.org/TR/CSS21/.

A.1. CSS Values

Every CSS property has a corresponding value. The color property, which formats font color, requires a color value to specify which color you want to use. The property color: #FFF; creates white text. Different properties require different types of values, but they come in four basic categories: colors, lengths and sizes, keywords, and URLs.

Colors

You can assign colors to many different properties, including those for font, background, and borders. CSS provides several different ways to specify color.

A.1.1.1. Keywords

A color keyword is simply the name of the color, like white or black. There are currently 17 recognized color keywords: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, and yellow. Some browsers accept more keywords, and CSS 3 promises to offer many more in the future (http://www.w3.org/TR/css3-color/).

A.1.1.2. RGB values

Computer monitors create colors using a mixture of red, green, and blue light. These RGB values can create (nearly) the full spectrum of color. Almost every design, illustration, and graphics program lets you specify colors using RGB, so it's easy to transfer a color from one of those programs to a CSS property. CSS represents RGB values in several ways:

- Hex values. The method most commonly used on the Web for identifying color, hex color values consist of three two-character numbers in the hexadecimal (that is, base 16) system. #FF0033 represents an RGB value composed of red (FF, which equals 255 in normal, base 10 numbers), green (00), and blue (33). The # tells CSS to expect hex numbers ahead, and it's required. If you leave off the #, a Web browser won't display the correct color.

Tip: If all three two-digit values have repeated digits, you can shorten the hex value by using just the first number of each pair. For example #361 means the same thing as #336611.

- RGB percentages. You can also specify a color using percentage values, like this: rgb(100%, 0%, 33%). You can get these numbers from image editing and design programs that can define colors using percentages (which is most of them).
- Decimal values. Finally, you can use decimal RGB values to specify a color. The format is similar to the percentage option, but you use a number from 0 to 255 to indicate each color: rgb(255, 0, 33).

It doesn't matter which method you usethey all work. For consistency's sake, you should pick one way of specifying RGB values and stick with it. The Windows and Mac operating systems both have color pickers which let you find the perfect color from a palette of millions, and then show you the RGB value. Alternatively, you can use this free online color picker: www.ficml.org/jemimap/style/color/wheel.html.

Tip: Many Mac programs such as TextEdit, let you open the color picker by pressing -Shift-C.

Lengths and Sizes

CSS provides many different ways to measure the size of type, the width of a box, or the thickness of a borderline. To indicate type size, you can use inches, picas, points, centimeters, millimeters, em-heights, ex-heights, pixels, and percentages. However, even though there are a lot of options, most don't apply to the world of onscreen display, for reasons discussed in Section 6.2. You really need to think about these three only pixels, ems, and percentages.

A.1.2.1. Pixels

A pixel is a single dot on a computer screen. Pixels give you a consistent method of identifying lengths and font sizes from computer to computer: 72 pixels on one monitor is 72 pixels on another monitor. That doesn't mean the actual, real-world length is the same for everyone, though. Since people set their monitors to different resolutions800 x 600, 1024 x 768, 1600 x 1200, or whatever72 pixels may take up 1 inch on one monitor, but only half an inch for someone else. Nevertheless, pixels give you the most consistent control over presentation.

Note: There's just one drawback to using pixels: folks using Internet Explorer 6 or earlier can't resize any type that's sized using pixels. If your text is too small for someone's eyes, the visitor won't be able to enlarge it to make it more readable. (See Section 6.2.2 for more on pixel measurements.)

A.1.2.2. Ems

Originally from the typographic world, an em is a unit that represents the height of the capital letter M for a particular font. In Web pages, one em is the height of the Web browser's base text size, which is usually 16 pixels. However, anyone can change that base size setting, so 1em may be 16 pixels for one person, but 24 pixels in someone else's browser. In other words, ems are a relative unit of measurement.

In addition to the browser's initial font size setting, ems can inherit size information from containing tags. A type size of .9em would make text about 14 pixels tall on most browsers with a 16 pixel base size. But if you have a <p> tag with a font size of .9ems, and then a <strong> tag with a font size of .9ems inside that <p> tag, that <strong> tag's em size isn't 14 pixelsit's 12 pixels (16 x .9 x .9). So keep inheritance in mind when you use em values.

A.1.2.3. Percentages

CSS uses percentages for many different purposes, like sizing text, determining the width or height of an element, and specifying the placement of an image in the background of a style, to name a few. Now, what you're taking a percentage of varies from property to property. For font sizes, the percentage is calculated based on the text's inherited value. Say the general font size for a paragraph is 16 pixels tall. If you created a style for one special paragraph and set its font size to 200 percent, that text is displayed at 32 pixels tall. When applied to width, however, percentages are calculated based on the width of the page, or on another parent element with a set width. You specify a percentage with a number followed by the percent sign: 100%.

Keywords

Instead of color or size, many properties have their own specific values that affect how the properties display and are represented by keywords. The text-align property, which aligns text on screen, can take one of four keywords: right, left, center, and justify. Since keywords vary from property to property, read the property descriptions that follow to learn the keyword appropriate to each property.

One keyword, however, is shared by all properties inherit. This keyword lets you force a style to inherit a value from a parent element. You can use the inherit keyword on any property. This keyword gives you the power to make styles inherit properties that aren't normally inherited from parent tags. For instance, say you use the text-decoration property to underline a paragraph. Other tags, such as <em> and <strong>, inside the <p> tag don't inherit this value, but you can force them to do so with the inherit keyword:

```
em, strong {
  text-decoration: inherit;
}
```

That way, the em and strong tags display the same text-decoration value as their parent <p> tagunderline, in this case. So the <em> and <strong> elements of the paragraph each get underlined as does the entire paragraph so you'd end up with double underlines under emphasized text (a good reason why that property isn't inherited normally). If you change the <p> tag's text-decoration value to overline instead of underline, the <em> and <strong> tags inherit that value and display overlines, too.

Note: Underline/overline isn't a very useful example, mainly because inherit isn't a very useful value. But this wouldn't be a Missing Manual if it didn't give you all the facts.

URLs

URL values let you point to another file on the Web. For example, the background-image property accepts a URLthe path to the file on the Webas its value, which lets you assign a graphic file as a background for a page element. This technique is handy for adding a tiling image in the background of a page or for using your own graphic for bulleted lists (see Section 8.2).

In CSS, you specify an URL like this: url(images/tile.gif). A style that adds an image called tile.gif to the background of the page would look like this:

```
body { background-image: url(images/tile.gif); }
```

Unlike HTML, in CSS, quotes around the URL are optional, so url("images/tile.gif"), url('images/tile.gif'), and url(images/tile.gif) are equivalent.

Note: The URL itself is just like the HTML href attribute used for links, meaning you can use an absolute URL like http://www.missingmanuals.com/images/tile.gif, a root-relative path like /images/tile.gif, or a document-relative URL like ../../images/tile.gif. See Section 8.3 for the full story on these kinds of paths.

A.2. Text Properties

The following properties affect how text is formatted on a Web page. Since most of the properties in this category are inherited, you don't necessarily have to apply them to tags specifically intended for text (like the <p> tag). You can apply these properties to the <body> tag, so that other tags inherit and use the same settings. This technique is a quick way to create an overall font, color, and so on for a page or section.

color (inherited)

Sets the color of text. Since it's inherited, if you set the color of the <body> tag to red, for example, all text inside of the bodyand all other tags inside the <body> tagis red, too.

- Values: any valid color value
- Example: *color: #FFFF33;*

Note: The preset link colors for the <a> tag override color inheritance. In the above example, any links inside the <body> tag would still be standard hyperlink blue. See Section 9.1 for ways to change preset link colors.

font (inherited)

This is a shortcut method for cramming the following text properties into a single style declaration: font-style, font-variant, font-weight, font-size, line-height, and font-family. (Read on for the individual descriptions.)

You must separate each value by a space and include at least font-size and font-family, and those two properties must be the last two items in the declaration. The others are optional. If you don't set a property, the browser uses its own preset value, potentially overriding inherited properties.

- Values: Any value that's valid for the specific font property. When including a line-height, add a slash followed by the line-height after the font size like this: 1.25em/150%.
- Example: *font: italic small-caps bold 1.25em/150% Arial, Helvetica, sans-serif;*

font-family (inherited)

Specifies the font the browser should use to display text. Fonts are usually specified as a series of three to four options to accommodate the fact that a particular font may not be installed on a visitor's computer. See Section 6.1.1.

- Values: A comma-separated list of font names. When a font has a space in its name, surround that font name with quotes. The last font listed is usually a generic font type instructing browsers to choose a suitable font if the other listed fonts aren't available: serif, sans-serif, monotype, fantasy, or cursive.
- Example: *font-family: "Lucida Grande", Arial, sans-serif;*

font-size (inherited)

Sets the size of text. This property is inherited, which can lead to some weird behaviors when using relative length measurements like percentages and ems.

- Values: Any valid CSS measurement unit (Section A.1.1.1), plus the following keywords: xx-small, x-small, small, medium, large, x-large, xx-large, larger, and smaller. Medium represents the Web browser's normal, preset font size, and the other sizes are multiples of medium. The exact numbers depend on the browser, but they're generally a factor of 1.2. For example, large is 1.2 times as big as medium. Due to the uncertainty of how each browser handles these keywords, many designers use pixels, ems, or percentages instead.
- Example: *font-size: 1.25em;*

Note: When the font-size property is inherited from another tag, these keywords multiply the inherited font size by the same factor (1.2 in most browsers).

font-style (inherited)

Makes text italic. Applied to italic text, it turns it back to plain text. The options italic and oblique are functionally the same.

- Values: italic, oblique, normal
- Example: *font-style: italic;*

font-variant (inherited)

Makes text appear in small caps, like this: SPECIAL PRESENTATION. The value normal removes small caps from text already formatted that way.

- Values: small-caps, normal
- Example: *font-variant: small-caps;*

font-weight (inherited)

Makes text bold, or removes bolding from text already formatted that way.

- Values: CSS actually provides 14 different font-weight keywords, but only a couple actually work with today's browsers and computer systemsbold and normal.
- Example: *font-weight: bold;*

letter-spacing (inherited)

Adjusts the space between letters to spread out letters (adding spacing between each) or cram letters together (removing space).

- Values: Any valid CSS measurement unit, though ems and pixels are most common. For this property, percentages don't work in most browsers. Use a positive value to increase the space between letters and a negative value to remove space (scrunch letters together). The value normal resets letter-spacing to its regular browser value of 0.

- Examples: *letter-spacing: -1px; letter-spacing: 2em;*

line-height (inherited)

Adjusts space between lines of text in a paragraph (often called line spacing in word processing programs). The normal line height is 120 percent of the size of the text (Section 6.4).

- Values: Most valid CSS lengths (Section A.1.1.1), though ems and pixels and percentages are most common.
- Example: *line-height: 200%;*

text-align (inherited)

Positions a block of text to the left, right, or center of the page or container element.

- Values: left, center, right, justify (the justify option often makes text difficult to read on monitors).
- Example: *text-align: center;*

text-decoration

Adds lines above, under, and/or through text. Underlining is common with links, so it's usually a good idea not to underline text that isn't a link. The color of the underline, overline, or strike-through line is the same as the font color of the tag being styled. The property also supports a blink value that makes text flash off and on obnoxiously.

- Values: underline, overline, line-through, blink, none. The none value turns off all decoration. Use this to hide the underline that normally appears under links. You can also add multiple decorations by listing the name of each type (except none) separated by a space.
- Example: *text-decoration: underline overline line-through;*

text-indent (inherited)

Sets the indent size of the first line of a block of text. The first line can be indented (as in many printed books) or outdented, so that the first line hangs off and over the left edge of the rest of the text.

- Values: Any valid CSS measurement unit. Ems and pixels are most common; percentages behave differently than with the font-size property. Here, percentages are based on the width of the box containing the text, which can be the width of the entire browser window. So 50% would indent the first line half of the way across the window (see Section 7.5.1 for a detailed explanation). To outdent (hang the first line off the left edge), use a negative value. This technique works well in conjunction with a positive left-margin property (Section A.1.1), which indents the left side of the other lines of text a set amount.
- Example: *text-indent: 3em;*

text-transform (inherited)

Changes the capitalization of text, so text appears in all uppercase letters, all lowercase, or only the first letter of each word capitalized.

- Values: uppercase, lowercase, capitalize, none. The none option returns the text to whatever case is in the actual HTML code. If aBCDefg are the actual letters typed in HTML, then none removes any other inherited case set by an ancestor tag and displays aBCDefg onscreen.
- Example: *text-transform: uppercase;*

vertical-align

Sets the baseline of an inline element relative to the baseline of the surrounding contents. With it, you can make a character appear slightly above or below surrounding text. Use this to create superscript characters like ™, ®, or ©. When applied to a table cell, the values top, middle, bottom, and baseline control the vertical placement of content inside the cell (Section 10.2.1).

- Values: baseline, sub, super, top, text-top, middle, bottom, text-bottom, a percentage value, or an absolute value (like pixels or ems). Percentages are calculated based on the element's line-height value (Section A.1.1).
- Examples: *vertical-align: top; vertical-align: -5px; vertical-align: 75%;*

white-space

Controls how the browser displays space characters in the HTML code. Normally, if you include more than one space between words"Hello Dave"a Web browser displays only one space"Hello Dave." You can preserve any white space exactly as is in the HTML using the pre value, which does the same as the HTML <pre> tag. In addition, Web browsers will split a line of text at a space, if the line won't fit within the window's width. To prevent text from wrapping, use the nowrap value. But the nowrap value makes all of the paragraph's text stay on one line, so don't use it with long paragraphs (unless you like the idea of making your visitors scroll endlessly to the right).

- Values: nowrap, pre, normal. Two other valuespre-line and pre-wrapdon't work in many browsers.
- Example: *white-space: pre;*

word-spacing (inherited)

Works like the letter-spacing property (Section A.1.1), but instead of letters, it adjusts space between words.

- Values: Any valid CSS measurement unit, though ems and pixels are most common;

percentages don't work in most browsers. Use a positive value to increase the space between words and a negative value to remove space (scrunch words together). The value normal resets word spacing to its regular browser value of 0.

- Examples: *word-spacing: -1px; word-spacing: 2em;*

A.3. List Properties

The following properties affect the formatting of bulleted lists (<ul>) and numbered lists (<ol>).

list-style (inherited)

This property is a shorthand method of specifying the three properties listed next. You can include a value for one or more of those properties, separating each by a space. You can even use this property as a shortcut for writing a single property and save a couple of keystrokes: list-style: outside, instead of list-style-position: outside. If you specify both a type and an image, a Web browser will display the bullet type (disc, square, and so on) only if it can't find the image. This way, if the path to your custom bullet image doesn't work, you don't end up with a bulletless bulleted list.

- Values: Any valid value for list-style-type, list-style-image, and/or list-style-position.
- Example: *list-style: disc url(images/bullet.gif) inside;*

list-style-image (inherited)

Specifies an image to use for a bullet in a bulleted list.

- Values: an URL value (Section A.1.1) or none.
- Example: *list-style-image: url(images/bullet.gif);*

Tip: The background-image property does the custom bullet job just as well and offers more control (see Section 8.2).

list-style-position (inherited)

Positions the bullets or numbers in a list. These markers can appear outside of the text, hanging off to the left, or inside the text (exactly where the first letter of the first line normally begins). The outside position is how Web browsers normally display bullets and numbers.

- Values: inside, outside
- Example: *list-style: inside;*

list-style-type (inherited)

Sets the type of bullet for a listround, square, roman numeral, and so on. You can theoretically turn an unordered (bulleted) list into an ordered (numbered) list by changing the list-style-type property, but it doesn't work in all browsers (including Internet Explorer for Windows). Use the none option to completely remove bullets or numbers from the list.

- Values: disc, circle, square, decimal, decimal-leading-zero, upper-alpha, lower-alpha, upper-roman, lower-roman, lower-greek, none
- Example: *list-style-type: square;*

A.4. Padding, Borders, and Margins

The following properties control the space around an element, and let you add border lines to a style.

border

Draws a line around the four edges of an element.

- Values: The width (thickness) of the border line in any valid CSS measurement unit (except percentages).

You can also specify a style for the line: solid, dotted, dashed, double, groove, ridge, inset, outset, none, and hidden. (See Figure 7-7 in Section 7.3 for an illustration of the different styles.) The none and hidden values do the same thingremove any border. Finally, you can specify a color using any valid CSS color type (a keyword like green or a hex number like #33fc44).

- Example: *border: 2px solid #f33;*

border-top, border-right, border-bottom, border-left

Adds a border to a single edge. For example, border-top adds a border to the top of the element.

- Values: same as for border.
- Example: *border-left: 1em dashed red;*

border-color

Defines the color used for all four borders.

- Values: Any valid CSS color type (a keyword like green or a hex number like #33fc44).
- Example: *border-color: rgb(255,34,100);*

border-top-color, border-right-color, border-bottom-color, border-left-color

Functions just like the border-color property but sets color for only one edge. Use these properties to override the color set by the border property. In this way, you can customize the color for an individual edge while using a more generic border style to define the basic size and style of all four edges.

- Values: see border-color above.
- Example: *border-left-color: #333;*

border-style

Defines the style used for all four borders.

- Values: One of these key words: solid, dotted, dashed, double, groove, ridge, inset, outset, none, and hidden. See Figure 7-7 in Section 7.3 for an illustration of the different styles. The none and hidden values act identicallythey remove any border.
- Example: *border-style: inset;*

border-top-style, border-right-style, border-bottom-style, border-left-style

Functions just like the border-style property, but applies only to one edge.

- Values: see border-style above.
- Example: *border-top-style: none;*

border-width

Defines the width or thickness of the line used to draw all four borders.

- Values: Any valid CSS measurement unit except percentages. The most common are ems and pixels.
- Example: *border-width: 1px;*

border-top-width, border-right-width, border-bottom-width, border-left-width

Functions just like the border-width property but applies only to one edge.

- Values: see border-width above.
- Example: *border-bottom-width: 3em;*

outline

This property is a shorthand way to combine outline-color, outline-style, and out-line-width (listed next). An outline works just like a border, except the outline takes up no space (that is, it doesn't add to the width or height of an element), and it applies to all four edges. It's intended more as a way of highlighting something on a page than as a design detail. Outline works in Firefox, Safari, and Opera, but not in Internet Explorer.

- Values: The same as for border with one exceptionsee outline-color next.
- Example: *outline: 3px solid #F33;*

outline-color

Specifies the color for an outline (see outline above).

- Values: Any valid CSS color, plus the value invert, which merely reverses the color the outline is sitting on. If the outline is drawn on a white background, the invert value makes the outline black. Works just like border-color (Section A.1.1).
- Example: *outline-color: invert;*

outline-style

Specifies the type of line for the outline dotted, solid, dashed, and so on.

- Values: Same as border-style (Section A.1.1).
- Example: *outline-style: dashed;*

outline-width

Specifies the thickness of the outline. Works just like border-width (Section A.1.1).

- Values: Any valid CSS measurement unit except percentages. The most common are ems and pixels.
- Example: *outline-width: 3px;*

padding

Sets the amount of space between the content and border and edge of the background. Use it to add empty space around text, images, or other content. (See Figure 7-1 in Section 7.2 for an illustration.)

- Values: Any valid CSS measurement unit, like pixels or ems. Percentage values are based on the width of the containing element. A headline that's a child of the <body> tag uses the width of the browser window to calculate a percentage value, so a padding of 20 percent adds 20 percent of the window's width. If the visitor resizes his browser, the padding size changes proportionately. You can specify the padding for all four edges by using a single value, or set individual padding sizes per edge using this order: top, right, bottom, left.
- Examples: *padding: 20px; padding: 2em 3em 2.5em 0;*

padding-top

Works just like the padding property, but sets padding for top edge only.

- Example: *padding-top: 20px;*

padding-right

Works just like the padding property, but sets padding for right edge only.

- Example: *padding-right: 20px;*

padding-bottom

Works just like the padding property, but sets padding for bottom edge only.

- Example: *padding-bottom: 20px;*

padding-left

Works just like the padding property, but sets padding for left edge only.

- Example: *padding-left: 20px;*

margin

Sets the amount of space between an element's border and the margin of other elements (see Figure 7-1 in Section 7.2). It lets you add white space between two elementsbetween one picture and another picture, or between a sidebar and the main content area of a page.

Note: Vertical margins between elements can collapse. That is, browsers use only the top or bottom margin and ignore the other, creating a smaller gap than expected (see Section 7.2.2).

- Values: Any valid CSS measurement unit like pixels or ems. Percentage values are based on the width of the containing element. A headline that's a child of the body tag uses the width of the browser window to calculate a percentage value, so a margin of 10 percent adds 10 percent of the window's width to the edges of the headline. If the visitor resizes his browser, the margin size changes. As with padding, you specify the margin for all four edges using a single value, or set individual margins in this order: top, right, bottom, left.
- Examples: *margin: 20px; margin: 2em 3em 2.5em 0;*

margin-top

Works just like the margin property, but sets margin for top edge only.

- Example: *margin-top: 20px;*

margin-right

Works just like the margin property, but sets margin for right edge only.

- Example: *margin-right: 20px;*

margin-bottom

Works just like the margin property, but sets margin for bottom edge only.

- Example: *margin-bottom: 20px;*

margin-left

Works just like the margin property, but sets margin for left edge only.

- Example: *margin-left: 20px;*

A.5. Backgrounds

CSS provides several properties for controlling the background of an element, including coloring the background, placing an image behind an element, and controlling how that background image is positioned.

background

Provides a shorthand method of specifying properties that appear in the background of an element, like a color, an image, and the placement of that image. It combines the five background properties (described next) into one compact line, so you can get the same effect with much less typing. However, if you don't set one of the properties, browsers use that property's normal value instead. For example, if you don't specify how a background image should repeat, browsers will tile that image from left to right and top to bottom (see Section 8.3).

- Values: The same values used for the background properties listed next. The order of the properties isn't important (except for positioning as described below) but usually follow the order of background-color, background-image, background-repeat, background-attachment, background-position.
- Example: *background: #333 url(images/logo.gif) no-repeat fixed left top;*

background-attachment

Specifies how a background image reacts when your visitor scrolls the page. The image either scrolls along with the rest of the content or remains in place. You can add a logo to the upper-left corner of a very long Web page, using the background-attachment property's fixed value, and make that image stay in the upper-left corner even when the page is scrolled. (In Internet Explorer 6 and earlier, this property works only for the <body> tag.)

- Values: scroll or fixed. Scroll is the normal behavior: An image will scroll off the screen along with text. Fixed locks the image in place.
- Example: *background-attachment: fixed;*

background-color

Adds a color to the background of a style. The background sits underneath the border and underneath a background image, a fact to keep in mind if you use one of the non-solid border styles like dashed or dotted. In these cases, the background color shows through the gaps between the dashes or dots.

- Values: any valid color value (Section A.1).
- Example: *background-color: #FFF;*

background-image

Places an image into the background of a style. Other page elements sit on top of the background image, so make sure that text is legible where it overlaps the image. You can always use padding to move content away from the image, too. The image tiles from left to right and top to bottom, unless you set the background-repeat property as well.

- Values: The URL of an image.
- Examples: *background-image: url(images/photo.jpg); background-image: url(http://www.example.org/photo.jpg);*

background-position

Controls the placement of an image in the background of a page element. Unless you specify otherwise, an image begins in the element's top-left corner. If the image tiles, background-position controls the image's start point (see background-repeat next). If you position an image in the center of an element, the browser puts the image there, and then tiles the image up and to the left and down and to the right. In many cases, the exact placement of an image doesn't cause a visible difference in the background tiling, but it lets you make subtle changes to the positioning of a pattern in the background.

- Values: You can use any valid CSS measurement unit like pixels or ems, as well as keywords or percentages. The values come in pairs, with the first being the horizontal position, and the second being vertical. Keywords include left, center, and right for horizontal positioning and top, center, and bottom for vertical. Pixel and em values are calculated from the top-left corner of the element, so to place a graphic 5 pixels from the left edge and 10 pixels from the top, you'd use a value of 5px 10px.

Percentage values map one point on the image to one point in the background of the element, calculated by the specified percentage from the left and top edges of the image and the specified percentage from the left and top edges of the element. 50% 50% places the point that's 50 percent across and 50 percent down the image on top of the point that's 50 percent across and 50 percent down the element. In other words, it puts the image directly in the middle of the element (see Section 8.4.3). You can mix and match these Values: If you want, use a pixel value for horizontal placement and a percentage value for vertical placement.

- Examples: *background-position: left top; background-position: 1em 3em; background-position: 10px 50%;*

background-repeat

Controls whether, or how, a background image repeats. Normally, background images tile from the top left to the bottom right, filling the element's entire background.

- Values: repeat, no-repeat, repeat-x, repeat-y. The repeat option is the normal methodtiling left to right, top to bottom. No-repeat places the image a single time in the background with no tiling. Repeat-x tiles the image top to bottom onlyperfect for adding a graphical sidebar. Repeat-y tiles the image from left to right only, so you can add a graphical bar to an element's top, middle, or bottom.
- Example: *background-repeat: no-repeat;*

A.6. Page Layout Properties

The following properties control the placement and size of elements on a Web page.

bottom

This property is used with absolute, relative, and fixed positioning (see Section A.1.1). When used with absolute or fixed positioning, bottom determines the position of the bottom edge of the style relative to the bottom edge of its closest positioned ancestor. If the styled element isn't inside of any positioned tags, then the placement is relative to the bottom edge of the browser window. You can use this property to place a footnote at the bottom of the browser window. When used with relative positioning, the placement is calculated from the element's bottom edge (prior to positioning). See Section 12.1.2.

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages. Percentages are calculated based on the width of the containing element.
- Example: *bottom: 5em;*

Note: Internet Explorer 6 and earlier can have a problem when positioning an element using the bottom property. See Section 12.1.4 for details.

clear

Prevents an element from wrapping around a floated element. Instead, the cleared element drops below the bottom of the floated element.

- Values: left, right, both, none. The left option means the element can't wrap around left-floated elements. Similarly, right drops the element below any right-floated items. The both value prevents an element from wrapping around either left-or right-floated elements. None turns the property off, so you use it to override a previously set clear property. This trick comes in handy when a particular tag has a style that drops below a floated element but you want the tag to wrap in just one case. Create a more specific style (Section 5.3) to override the float for that one tag.
- Example: *clear: both;*

clip

Creates a rectangular window that reveals part of an element. If you had a picture of your high-school graduating class, and the class bully was standing on the far right edge of the photo, you could create a display area that crops out the image of your tormentor. The full image is still intact, but the clipping area only displays the bully free portion of it. The clip property is most effective when used with JavaScript programming to animate the clip. You can start with a small clipping area and expand it until the full photo is revealed.

- Values: Coordinates of a rectangular box. Enclose the coordinates in parentheses and precede them by the keyword rect, like so: rect(5px,110px,40px,10px);.

Here's how the order of these coordinates works: The first number indicates the top offsetthe top edge of the clipping window. In this example, the offset is 5px, so everything in the first four rows of pixels is hidden. The last number is the left offsetthe left edge of the clipping window. In this example, the offset is 10px, so everything to the left (the first 9 pixels of the element) is hidden. The second number is the width of the clipping window plus the last number; if the left edge of the clip is 10 pixels and you want the visible area to be 100 pixels, the second number would be 110px. The third number is the height of the clipping region plus the top offset (the first number). So, in this example, the clipping box is 30 pixels tall (30px + 10px = 40px).

- Example: *clip: rect(5px,110px,40px,10px);*

Tip: Since the order of the coordinates is a little strange, most designers like to start with the first and last numbers, and then compute the two other numbers from them.

---

display

Determines the kind of box used to display a page elementblock-level or inline (Section 7.2.4). Use it to override how a browser usually displays a particular element. You can make a paragraph (block-level element) display without line breaks above and below itexactly like, say, a link (inline element).

- Values: block, inline, none. The display property accepts 17 values, most of which have no effect in the browsers available today. Block, inline, and none, however, work in almost all browsers. Block forces a line break above and below an element, just like other block-level elements (like paragraphs and headers). Inline causes an element to display on the same line as surrounding elements (just as text within a <strong> tag appears right on the same line as other text). None makes the element completely disappear from the page. Then, you can make the element reappear with some JavaScript programming or the :hover pseudo-class (see Section 3.1).
- Example: *display: block;*

float

Moves (floats) an element to the left or right edge of the browser window, or, if the floated element's inside another element, to the left or right edge of that containing element. Elements that appear after the floated element move up to fill the space to the right (for left floats) or left (for right floats), and then wrap around the floated element. Use floats for simple effects, like moving an image to one side of the page, or for very complex layouts like those described in Chapter 11.

Values: left, right, none. None turns off floating entirely, which comes in handy when a particular tag has a style with a left or right float applied to it, and you want to create a more specific style to override the float for that one tag.

Example: *float: left;*

height

Sets the height of the content areathe area of an element's box that contains content like text, images, or other tags. The element's actual onscreen height is the total of height, top and bottom margins, top and bottom padding, and top and bottom borders.

- Values: Any valid CSS measurement unit such as pixels, ems, or percentages. Percentages are calculated based on the height of the containing element.
- Example: *height: 50%;*

Note: Sometimes, your content ends up taller than the set heightif you type a lot of text, for instance, or your visitor increases text size in her browser. Browsers handle this situation differently: IE 6 and earlier simply make the box bigger, while other browsers make the content extend outside of the box. The overflow property controls what happens in this case (see Section 7.5.2).

---

left

When used with absolute or fixed positioning (Section 12.1), this property determines the position of the left edge of the style relative to the left edge of its closest positioned ancestor. If the styled element isn't inside of any positioned tags, then the placement is relative to the left edge of the browser window. You can use this property to place an image 20 pixels from the left edge of the browser window. When used with relative positioning, the placement is calculated from the element's left edge (prior to positioning).

- Values: Any valid CSS measurement unit such as pixels, ems, or percentages.
- Example: *left: 5em;*

max-height

Sets the maximum height for an element. That is, the element's box may be shorter than this setting, but it can't be any taller. If the element's contents are taller than the max-height setting, they overflow the box. You can control what happens to the excess using the overflow property (Section 7.5.2). Internet Explorer 6 (and earlier) doesn't understand the max-height property.

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages. Browsers calculate percentages based on the height of the containing element.
- Example: *max-height: 100px;*

max-width

Sets the maximum width for an element. The element's box can be narrower than this setting, but no wider. If the element's contents are wider than the max-width setting, they overflow the box, which you can control with the overflow property (Section 7.5.2). You mostly use max-width in liquid layouts (Section 11.1) to make sure a page design doesn't become unreadably wide on very large monitors. This property doesn't work in Internet Explorer 6 (or in earlier versions).

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages. Percentages are calculated based on the width of the containing element.
- Example: *max-width: 950px;*

min-height

Sets the minimum height for an element. The element's box may be taller than this setting, but it can't be shorter. If the element's contents aren't as tall as the min-height setting, the box's height shrinks to meet the min-height value. Internet Explorer 6 (and earlier) doesn't recognize this property.

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages. Percentages are based on the containing element's height.
- Example: *min-height: 20em;*

min-width

Sets the minimum width for an element. The element's box may be wider than this setting, but it can't be narrower. If the element's contents aren't as wide as the min-width value, the box simply gets as thin as the min-width setting. You can also use min-width in liquid layouts, so that the design doesn't disintegrate at smaller window widths. When the browser window is thinner than min-width, it adds horizontal scroll bars. Internet Explorer 6 (and earlier) doesn't understand this property.

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages. Percentages are based on the containing element's width.
- Example: *min-width: 760px;*

Note: You usually use the max-width and min-width properties in conjunction when creating liquid layouts. See Chapter 11 (Section 11.4.3).

---

overflow

Dictates what should happen to text that overflows its content area, like a photo that's wider than the value set for the width property.

Note: IE 6 (and earlier) handles overflow situations differently than other browsers. See Section 7.5.2.

- Values: visible, hidden, scroll, auto. Visible makes the overflowing content extend outside the boxpotentially overlapping borders and other page elements on the page. IE 6 (and earlier) simply enlarges the box (borders and all) to accommodate the larger content. Hidden hides any content outside of the content area. Scroll adds scroll bars to the element so a visitor can scroll to read any content outside the content areasort of like a mini-frame. Auto adds scrollbars only when they're necessary to reveal more content.
- Example: *overflow: hidden;*

position

Determines what type of positioning method a browser uses when placing an element on the page.

- Values: static, relative, absolute, fixed. Static is the normal browser modeone block-level item stacked on top of the next with content flowing from the top to the bottom of the screen. Relative positions an element in relation to where the element currently appears on the pagein other words, it can offset the element from its current position. Absolute takes an element completely out of the page flow. Other items don't see the absolute element and may appear underneath it. It's used to position an element in an exact place on the page, or to place an element in an exact position relative to a parent element that's positioned with absolute, relative or fixed positioning. Fixed locks an element on the page, so that when the page is scrolled, the fixed element remains on the screenmuch like HTML frames. Internet Explorer 6 (and earlier) ignores the fixed option.
- Example: *position: absolute;*

Note: You usually use relative, absolute, and fixed in conjunction with left, right, top, and bottom. See Chapter 12 for the full details on positioning.

---

right

When used with absolute or fixed positioning (Section 12.1), this property determines the position of the right edge of the style relative to the right edge of its closest positioned ancestor. If the styled element isn't inside of any positioned tags, then the placement is relative to the right edge of the browser window. You can use this property to place a sidebar a set amount from the right edge of the browser window. When used with relative positioning, the placement is calculated from the element's right edge (prior to positioning).

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages.
- Example: *left: 5em*;

Note: Internet Explorer 6 (and earlier) can have problems when positioning an element using the right property. See Section 12.1.4 for details.

---

top

Does the opposite of the bottom property (Section A.1.1). In other words, when used with absolute or fixed positioning, this property determines the position of the top edge of the style relative to the top edge of its closest positioned ancestor. If the styled element isn't inside of any positioned tags, then the placement is relative to the top edge of the browser window. You can use this property to place a logo a set amount from the top edge of the browser window. When used with relative positioning, the placement is calculated from the element's top edge (prior to positioning).

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages.
- Example: *top: 5em;*

visibility

Determines whether a Web browser displays the element. Use this property to hide part of the content of the page, such as a paragraph, headline, or <div> tag. Unlike the display property's none valuewhich hides an element and removes it from the flow of the pagethe visibility property's hidden option doesn't remove the element from the page flow. Instead, it just leaves an empty hole where the element would have been. For this reason, you most often use the visibility property with absolutely positioned elements, which have already been removed from the flow of the page.

Hiding an element doesn't do you much good unless you can show it again. JavaScript programming is the most common way to toggle the visibility property to show and hide items on a page. You can also use the :hover pseudo-class (Section 9.1) to change an element's visibility property when a visitor hovers over some part of the page.

- Values: visible, hidden. You can use the collapse value to hide a row or column in a table as well.
- Example: *visibility: hidden;*

width

Sets the width of the content area (the area of an element's box that contains text, images, or other tags). The amount of onscreen space actually dedicated to the element may be much wider, since it includes the width of the left and right margin, left and right padding, and left and right borders. IE 6 (and earlier) handles overflow situations differently than other browsers. (See Section 7.5.2.)

- Values: Any valid CSS measurement unit, like pixels, ems, or percentages. Percentages are based on the containing element's width.
- Example: *width: 250px;*

z-index

Controls the layering of positioned elements. Only applies to elements with a position property set to absolute, relative, or fixed (Section 12.1). It determines where on the Z-axis an element appears. If two absolutely positioned elements overlap, the one with the higher z-index appears to be on top.

- Values: An integer value, like 1, 2, or 10. You can also use negative values, but different browsers handle them differently. The larger the number, the more "on top" the element appears. An element with a z-index of 20 appears below an element with a z-index of 100 (if the two overlap). However, when the element is inside another positioned element, it's "positioning context" changes and it may not appear above another elementno matter what its z-index value. See Figure 12-6.
- Example: *z-index: 12;*

Tip: The values don't need be in exact integer order. If element A has a z-index of 1, you don't have to set element B's z-index to 2 to put it on top. You can use 5, 10, and so on to get the same effect, as long as it's a bigger number. So, to make sure an element always appears above other elements, simply give it a very large value, like 10000.

A.7. Table Properties

There are a handful of CSS properties that relate solely to HTML tables. Chapter 10 has complete instructions on using CSS with tables.

border-collapse

Determines whether the borders around the cells of a table are separated or collapsed. When they're separated, browsers put a space of a couple of pixels between each cell. Even if you eliminate this space by setting the cellspacing attribute for the HTML <table> tag to 0, browsers still display double borders. That is, the bottom-border of one cell will appear above the top border of the cell below causing a doubling of border lines. Setting the border-collapse property to collapse eliminates both the space between cells and this doubling up of borderlines (Section 10.2.1). This property works only when applied to a <table> tag.

- Values: collapse, separate
- Example: *border-collapse: collapse;*

border-spacing

Sets the amount of space between cells in a table. It replaces the <table> tag's cell-spacing HTML attribute. However, Internet Explorer doesn't understand the border-spacing property, so it's best to continue to use the cellspacing attribute in your <table> tags to guarantee space between cells in all browsers.

Note: If you want to eliminate the space browsers normally insert between cells, just set the border-collapse property to collapse.

- Values: Two CSS length values. The first sets the horizontal separation (the space on either side of each cell) and the second sets the vertical separation (the space separating the bottom of one cell from the top of the one below it).
- Example: *border-spacing: 0 10px;*

caption-side

When applied to a table caption, this property determines whether the caption appears at the top or bottom of the table. (Since, according to HTML rules, the <caption> tag must immediately follow the opening <table> tag, a caption would normally appear at the top of the table.)

- Values: top, bottom
- Example: *caption-side: bottom;*

Note: Unfortunately, this property has no effect in any versions of Internet Explorer (as of this writing), so it's safest to stick with the HTML equivalent: <caption align="bottom"> or <caption align="top">.

empty-cells

Determines how a browser should display a table cell that's completely empty, which in HTML would look like this: <td></td>. The hide value prevents any part of the cell from being displayed. Instead, only an empty placeholder appears, so borders, background colors, and background images don't show up in an emptied cell. Apply this property to a style formatting the <table> tag.

- Values: show, hide
- Example: *empty-cells: show;*

Note: The empty-cells property has no effect if the border-spacing property is set to collapse.

table-layout

Controls how a Web browser draws a table, and can slightly affect the speed at which the browser displays it. The fixed setting forces the browser to render all columns the same width as the columns in the first row, which (for complicated technical reasons) draws tables faster. The auto value is the normal "browser just do your thing" value, so if you're happy with how quickly your tables appear on a page, don't bother with this property. If you use it, apply table-layout to a style formatting the <table> tag.

- Values: auto, fixed
- Example: *table-layout: fixed;*

A.8. Miscellaneous Properties

CSS 2.1 offers a few additionaland sometimes interesting properties. They let you enhance your Web pages with special content and cursors, offer more control over how a page prints, and so on. (Unfortunately, browser understanding of these properties is spotty at best.)

content

Specifies text that appears either before or after an element. Use this property with the :after or :before pseudo-elements. You can add an opening quotation mark in front of quoted material and a closing quotation after the quote. This property isn't supported by Internet Explorer (not even IE 7 as of this writing), so its use is limited.

- Values: Text inside of quotes "like this", the keywords normal, open-quote, close-quote, no-open-quote, no-close-quote. You can also use the value of an HTML attribute. (See "Revealing Links in Print" in Section 13.3.4 for an example.)
- Examples: *p.advert:before { content: "And now a word from our sponsor…"; }*
  a:after { content: " (" attr(href) ") "; }

Note: Adding text in this way (like the opening and closing quote example) is called generated content. Read a simple explanation of the generated content phenomenon at www.westciv.com/style_master/academy/css_tutorial/advanced/generated_content.html. For a deeper explanation, visit www.w3.org/TR/CSS21/generate.html.

cursor

Lets you change the look of the mouse pointer when it moves over a particular element. You can make a question mark appear next to the cursor when someone mouses over a link that provides more information on a subject (like a word definition).

- Values: auto, default, crosshair, pointer, move, e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help, progress. You can also use an URL value to use your own graphic as a cursor (but see the Note below).
- Example: *cursor: help; cursor: url(images/cursor.cur);*

Note: Only Internet Explorer and Firefox recognize URL cursor values, and only in Windows. For more information, visit www.echoecho.com/csscursors.htm and www.quirksmode.org/css/cursor.html.

orphans

Specifies the minimum number of lines of text that can be left at the bottom of a printed page. Suppose you're printing your Web page on a laser printer, and a fiveline paragraph is split between two pages, with just one line at the bottom of page one, and the four remaining lines at the top of page two. Because a single line all by itself looks odd (sort of like a lost orphanget it?), you can tell the browser to break a paragraph only if at least, say, three lines are left on the bottom of the page. (At this writing, only the Opera browser understands this property.)

- Values: a number like 1, 2, 3, or 5.
- Example: *orphans: 3;*

page-break-after

Determines whether a page break (in printing) occurs after a particular element. With it, you can make sure that a particular paragraph is always the last item to appear on a printed page.

- Values: auto, always, avoid, left, right. Auto represents the normal value and lets the browser determine when and how to break content across printed pages. Always forces the element that follows to appear at the top of a separate printed page, and it's the only value that works consistently across browsers. Avoid prevents a page break after an element; it's a great way to keep a headline with the paragraph that follows it, but unfortunately, most browsers don't understand it. Left and right determine whether the element following appears on a left-or right-handed page, which may force the browser to print an extra empty page. But since no browsers understand these values, don't worry about wasting paper. Browsers treat left and right the same as always.
- Example: *page-break-after: always;*

page-break-before

Works like page-break-after, except the page break appears before the styled element, placing it at the top of the next printed page. You can use this property to make sure headlines for different sections of a long Web page each appear at the top of a page.

- Values: same as page-break-after.
- Example: *page-break-before: always;*

page-break-inside

Prevents an element from being split across two printed pages. If you want to keep a photo and its caption together on a single page, wrap the photo and caption text in a <div> tag, and then apply a style with page-break-inside to that <div>. (At this writing, only Opera understands this property.)

- Values: avoid
- Example: *page-break-inside: avoid;*

widows

The opposite of orphans (Section A.1.1), it specifies the minimum number of lines that must appear at the top of a printed page. Say the printer can manage to fit four out of five lines of a paragraph at the bottom of a page and has to move the last line to the top of the next page. Since that line might look weird all by itself, use widows to make the browser move at least two or three (or whatever number of) lines together to the top of a printed page. (Only Opera understands this property, so it's of limited use.)

- Values: a number like 1, 2, 3 or 5.
- Example: *widows: 3;*