WDG's CSS Reference

Style Sheets Now!

Change the appearance of hundreds of Web pages by changing just one file... Influence presentation without losing visitors... All with the power and flexibility of Web style sheets.

Quick Tutorial

A basic introduction to Cascading Style Sheets.

CSS Structure and Rules

An introduction to the various kinds of selectors, pseudo-classes, pseudo-elements, and cascading order.

CSS Properties

Descriptions of the various properties available in Cascading Style Sheets, level 1.

Linking Style Sheets to HTML

Various methods of incorporating style sheets into an HTML document.

Style Sheet Dependence

How to misuse style sheets and make your pages inaccessible.

CSS References

Links to CSS specifications and other documentation.

The WDG invites all to visit **http://www.htmlhelp.com/** for updates to this and other HTML authoring resources. If you find any errors or omissions in the technical content of this Help file, please contact Liam Quinn, or John Pozadzides.

This Help file was created by the **Web Design Group** for the greater good of the Web authoring community. It may be distributed freely via the Internet so long as the file is not modified in any way. For information regarding distributing this Help file with any program, or for any commercial uses, please see **http://www.htmlhelp.com/distribution/guidelines/ .**

WDG's Copyright and Trademark Information

About the Web Design Group

In the interest of addressing your questions about who makes up the WDG, we have decided to present a short introduction in the form of a series of Questions and Answers. If you have any other questions, just ask.

What is the WDG?

The Web Design Group is made up of experienced HTML authors that have banded together in the hopes of providing guidance and instruction to Web authors at all stages of development.

Who is in the Web Design Group?

The members of the web design group are: (in no particular order)

- Arnoud "Galactus" Engelfriet
- Ian Butler
- John Pozadzides
- Liam Quinn
- Tina Marie Holmboe
- Nick Kew

Retired Members include:

- Craig D. Horton
- Dave Salovesh
- Filip Gieszczykiewicz

Why does the Web Design Group exist?

The WDG's charter reads as follows:

The Web Design Group was formed to promote the use of valid and creative HTML documents. The WDG officially has no preference for browser type, screen resolution, HTML publishing tool or any other means by which HTML may be incorrectly manipulated. The WDG's sole interest is in promoting the creation of Non-browser Specific, Non-resolution Specific, Creative and Informative sites that are Accessible to ALL users worldwide.

When was the WDG formed?

The WDG had it's humble beginning on May 25, 1996 with the issuance of invitations into the group by John Pozadzides. Of the original 10 invited members, four were unable to accept due to extenuating circumstances. Unfortunately, the two women who were invited both were unable to join, leaving the group with nothing but men. Despite this shortcoming, the members were able to function enough to provide all of the information available on this site until Tina Marie accepted the WDG's invitation to join on December 22, 1996. :-)

Where is the WDG located?

The members of the WDG reside on a virtual planet known as CyberSpace. Their physical bodies are however trapped in different countries on Earth.

How do they do that?

It ain't easy...

What are "they" saying about HTMLHelp.com?

Just look and see! http://www.htmlhelp.com/about/awards.html

Style Sheets Now!

Web style sheets have been around for a number of years, but their power and importance went largely unnoticed since few implementations existed. Web authors, anxious to add creativity to their pages by influencing the presentation, began to use Netscape's proprietary extensions rather than the more powerful style sheets. This was quite natural, since Netscape's proprietary extensions could be seen by a significant portion of Web users, while few would see style sheet enhancements.

Today, more and more browsers are implementing style sheets, opening authors' eyes to unique features that allow influence over presentation while preserving platform independence. The advantages of style sheets became apparent, as did the disadvantages of continually creating more HTML tags for presentation effects.

Netscape's BODY attributes are now widely accepted on the Web, and will soon become standardized in HTML 3.2 . Yet BGCOLOR, TEXT and friends simply do not provide the flexibility of style sheets. Most background images would leave a page unreadable to someone whose display only offered 16 colors; many pages are difficult to read with only 256 colors. With conventional BODY attributes, an author must choose whether the benefits of a background image outweigh the costs; with style sheets, an author can offer a number of different images, in different style sheets, so that the user can choose the "24-bit style sheet" or "8-bit style sheet," depending on how many colors his or her system can display. If no author-supplied style suits the user, he or she can simply ignore the author's style.

Style sheets can make an author's life much easier. While one could use <HR WIDTH="75%" SIZE=5 ALIGN=center> for every horizontal rule, this becomes very cumbersome for the author. With style sheets, one only needs to specify such presentational preferences once, and the style can be applied to an entire site. And if the author decides that WIDTH="50%" would be better, then he or she only needs to change this preference in one place, rather than having to search through hundreds of pages to change the HTML. Not only that, but style sheets also reduce download time when one file contains all the style information.

Style sheets also offer much more flexibility in terms of the presentation effects that they provide. Properties such as color, background, margin, border, and many more can be applied to all elements. With just HTML and its proprietary extensions, one must rely on attributes like BGCOLOR, which are only available for a few elements. Style sheets give the flexibility of applying a style to all paragraphs, or all level-two headings, or all emphasized text.

With style sheets, authors can use the text-indent property to indent text, rather than resorting to ugly kludges like <DD> or <IMG SRC="blank.gif" WIDTH=10 ALT=""> that carry with them many negative side-effects. Margins can be suggested without having to put the entire page in a table. Style sheets also reduce the need for multi-file search and replace; if an author decides to change the indentation of all paragraphs on a site, he or she only has to change one line on a style sheet.

Style sheets represent an enormous step forward for the Web. With the separation of content and presentation between HTML and style sheets, the Web no longer needs to drift away from the strong ideal of platform independence that provided the medium with its initial push of popularity. Authors can finally influence the presentation of documents without leaving pages unreadable to users.

Current Browser Support for CSS

There are already a handful of browsers which support style sheets. Most notably, Microsoft's Internet Explorer 3 (for Macintosh and Windows 95/NT4.0) has partial support implemented with wide ranging support promised for the next release. Netscape is promising support for CSS1 in Navigator 4.0.

Style sheet support on the UNIX platform has long existed with Emacs-W3 and Arena, and the W3C's Amaya browser brings a large degree of CSS support along with a "WYSIWYG" editor for making CSS-enhanced Web pages.

Beyond CSS1

Future extensions to CSS1 are aimed at bringing better support for printing Web pages, styles for speech and

Braille output, layout and positioning, fonts, and device-independent colors.

CSS Quick Tutorial

A style sheet is made up of style rules that tell a browser how to present a document. There are various ways of linking these style rules to your HTML documents, but the simplest method for starting out is to use HTML's STYLE element. This element is placed in the document HEAD, and it contains the style rules for the page.

Note that while the STYLE element is a good method of experimenting with style sheets, it has disadvantages that should be considered before one uses this method in practice. The advantages and disadvantages of the various methods are discussed in the section on linking style sheets to HTML.

Each rule is made up of a selector--usually an HTML element such as BODY, P, or EM --and the style to be applied to the selector.

There are numerous properties that may be defined for an element. Each property takes a value, which together with the property describes how the selector should be presented.

Style rules are formed as follows:

```
selector { property: value }
```

Multiple style declarations for a single selector may be separated by a semicolon:

```
selector { property1: value1; property2: value2 }
```

As an example, the following code segment defines the color and font-size properties for H1 and H2 elements:

```
<HEAD>
<TITLE>CSS Example</TITLE>
<STYLE>
  H1 { font-size: x-large; color: red }
  H2 { font-size: large; color: blue }
</STYLE>
</HEAD>
```

The above style sheet tells the browser to show level-one headings in an extra-large, red font, and to show level-two headings in a large, blue font. The CSS1 Specification formally defines all properties and values available. The properties and values are also given in the CSS Properties section.

This tutorial is intended as a very basic introduction to Cascading Style Sheets, and should provide enough information to allow you to experiment with a few of your own styles. For a more in-depth look at Cascading Style Sheets, read the following sections:

- CSS Structure and Rules
- CSS Properties

CSS Structure and Rules

Table of contents:

Basic Syntax

Rules

Selectors

Any HTML element is a possible CSS1 selector. The selector is simply the element that is linked to a particular style. For example, the selector in:

```
P { text-indent: 3em }
```

is P .

Class Selectors

A simple selector can have different classes, thus allowing the same element to have different styles. For example, an author may wish to display code in a different color depending on its language:

```
code.html { color: #191970 }
code.css  { color: #4b0082 }
```

The above example has created two classes, css and html for use with HTML's CODE element. The CLASS attribute is used in HTML to indicate the class of an element, e.g.,

```
<P CLASS=warning>Only one class is allowed per selector. For example,
code.html.proprietary is invalid.</p>
```

Classes may also be declared without an associated element:

```
.note { font-size: small }
```

In this case, the note class may be used with any element.

A good practice is to name classes according to their function rather than their appearance. The note class in the above example could have been named small, but this name would become meaningless if the author decided to change the style of the class so that it no longer had a small font size.

ID Selectors

ID selectors are individually assigned for the purpose of defining on a per-element basis. This selector type should only be used sparingly due to its inherent limitations. An ID selector is assigned by using the indicator "#" to precede a name. For example, an ID selector could be assigned as such:

```
#svp940 { text-indent: 3em }
```

This would be referenced in HTML by the ID attribute:

```
<P ID=svp94O>Text indented 3em</P>
```

Contextual Selectors

Contextual selectors are merely strings of two or more simple selectors separated by white space. These selectors can be assigned normal properties and, due to the rules of cascading order, they will take precedence over simple selectors. For example, the contextual selector in

```
P EM { background: yellow }
```

is P EM . This rule says that emphasized text within a paragraph should have a yellow background; emphasized text in a heading would be unaffected.

Declarations

Properties

A property is assigned to a selector in order to manipulate its style. Examples of properties include color, margin, and font.

Values

The declaration value is an assignment that a property receives. For example, the property color could receive the value red.

Grouping

In order to decrease repetitious statements within style sheets, grouping of selectors and declarations is allowed. For example, all of the headings in a document could be given identical declarations through a grouping:

```
H1, H2, H3, H4, H5, H6 {color: red; font-family: sans-serif }
```

Inheritance

Virtually all selectors which are nested within selectors will inherit the property values assigned to the outer selector unless otherwise modified. For example, a color defined for the BODY will also be applied to text in a paragraph.

There are some cases where the inner selector does not inherit the surrounding selector's values, but these should stand out logically. For example, the margin-top property is not inherited; intuitively, a paragraph would not have the same top margin as the document body.

Comments

Comments are denoted within style sheets with the same conventions that are used in C programming. A sample CSS1 comment would be in the format:

```
/* COMMENTS CANNOT BE NESTED */
```

Pseudo-classes and Pseudo-elements

Pseudo-classes and pseudo-elements are special "classes" and "elements" that are automatically recognized by CSS-supporting browsers. Pseudo-classes distinguish among different element types (e.g., visited links and active links represent two types of anchors). Pseudo-elements refer to sub-parts of elements, such as the first letter of a paragraph.

Rules with pseudo-classes or pseudo-elements take the form

```
selector:pseudo-class { property: value }
```

or

```
selector:pseudo-element { property: value }
```

Pseudo-classes and pseudo-elements should not be specified with HTML's CLASS attribute. Normal classes may be used with pseudo-classes and pseudo-elements as follows:

```
selector.class:pseudo-class { property: value }
```

or

```
selector.class:pseudo-element { property: value }
```

Anchor Pseudo-classes

Pseudo-classes can be assigned to the A element to display links, visited links and active links differently. The anchor element can give the pseudo-classes link, visited, or active. A visited link could be defined to render in a different color and even a different font size and style.

An interesting effect could be to have a currently selected (or "active") link display in a slightly larger font size with a different color. Then, when the page is re-selected the visited link could display in a smaller font size with a different color. The sample style sheet might look like this:

```
A:link    { color: red }
A:active  { color: blue; font-size: 125% }
A:visited { color: green; font-size: 85% }
```

First Line Pseudo-element

Often in newspaper articles, such as those in the Wall Street Journal, the first line of text in an article is displayed in bold lettering and all capitals. CSS1 has included this capability as a pseudo-element. A first-line pseudo-element may be used in any block-level element (such as P, H1, etc.). An example of a first-line pseudo-element would be:

```
P:first-line {
  font-variant: small-caps;
  font-weight: bold }
```

First Letter Pseudo-element

The first-letter pseudo-element is used to create drop caps and other effects. The first letter of text within an assigned selector will be rendered according to the value assigned. A first-letter pseudo-element may be used in any block-level element. For example:

```
P:first-letter { font-size: 300%; float: left }
```

would create a drop cap three times the normal font size.

Cascading Order

When multiple style sheets are used, the style sheets may fight over control of a particular selector. In these situations, there must be rules as to which style sheet's rule will win out. The following characteristics will determine the outcome of contradictory style sheets.

1.! important

Rules can be designated as important by specifying ! important. A style that is designated as important will win out over contradictory styles of otherwise equal weight. Likewise, since both author and reader may specify important rules, the author's rule will override the reader's in cases of importance. A sample use of the ! important statement:

```
BODY { background: url(bar.gif) white;
       background-repeat: repeat-x ! important }
```

2.Origin of Rules (Author's vs. Reader's)

As was previously mentioned, both authors and readers have the ability to specify style sheets. When rules between the two conflict, the author's rules will win out over reader's rules of otherwise equal weight. Both author's and reader's style sheets override the browser's built-in style sheets.

Authors should be wary of using ! important rules since they will override any of the user's ! important rules. A user may, for example, require large font sizes or special colors due to vision problems, and such a user would likely declare certain style rules to be ! important, since some styles are vital for the user to be able to read a page. Any ! important rules will override normal rules, so authors are advised to use normal rules almost exclusively to ensure that users with special style needs are able to read the page.

3.Selector Rules: Calculating Specificity

Style sheets can also override conflicting style sheets based on their level of specificity, where a more specific style will always win out over a less specific one. It is simply a counting game to calculate the specificity of a selector.

a. Count the number of ID attributes in the selector.
b. Count the number of CLASS attributes in the selector.
c. Count the number of HTML tag names in the selector.

Finally, write the three numbers in exact order with no spaces or commas to obtain a three digit number. (Note, you may need to convert the numbers to a larger base to end up with three digits.) The final list of numbers corresponding to selectors will easily determine specificity with the higher numbers winning out over lower numbers. Following is a list of selectors sorted by specificity:

```
#id1         {xxx} /* a=1 b=0 c=0 --> specificity = 100 */
UL UL LI.red {xxx} /* a=0 b=1 c=3 --> specificity = 013 */
LI.red       {xxx} /* a=0 b=1 c=1 --> specificity = 011 */
LI           {xxx} /* a=0 b=0 c=1 --> specificity = 001 */
```

4.Order of Specification

To make it easy, when two rules have the same weight, the last rule specified wins.

CSS1 Units

Length Units

A length value is formed by an optional + or -, followed by a number, followed by a two-letter abbreviation that indicates the unit. There are no spaces in a length value; e.g., 1.3 em is not a valid length value, but 1.3em is valid. A length of 0 does not require the two-letter unit identifier.

Both relative and absolute length units are supported in CSS1. Relative units give a length relative to another length property, and are preferred since they will better adjust to different media. The following relative units are available:

- em (ems, the height of the element's font)
- ex (x-height, the height of the letter "x")
- px (pixels, relative to the canvas resolution)

Absolute length units are highly dependent on the output medium, and so are less useful than relative units. The following absolute units are available:

- in (inches; 1in=2.54cm)
- cm (centimeters; 1cm=10mm)
- mm (millimeters)
- pt (points; 1pt=1/72in)
- pc (picas; 1pc=12pt)

Percentage Units

A percentage value is formed by an optional + or -, followed by a number, followed by %. There are no spaces in a percentage value.

Percentage values are relative to other values, as defined for each property. Most often the percentage value is relative to the element's font size.

Color Units

A color value is a keyword or a numerical RGB specification.

The 16 keywords are taken from the Windows VGA palette: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow.

RGB colors are given in one of four ways:

- #rrggbb (e.g., #00cc00)
- #rgb (e.g., #0c0)
- rgb(x,x,x) where x is an integer between 0 and 255 inclusive (e.g., rgb(0,204,0))
- rgb(y%,y%,y%) where y is a number between 0.0 and 100.0 inclusive (e.g., rgb(0%,80%,0%))

The examples above all specify the same color.

URLs

A URL value is given by url(foo), where foo is the URL. The URL may be optionally quoted with either single (') or double (") quotes and may contain whitespace before or after the (optionally quoted) URL.

Parentheses, commas, spaces, single quotes, or double quotes in the URL must be escaped with a backslash. Partial URLs are interpreted relative to the style sheet source, not to the HTML source.

Examples:

```
BODY { background: url(stripe.gif) }
BODY { background: url(http://www.htmlhelp.com/stripe.gif) }
BODY { background: url( stripe.gif ) }
BODY { background: url("stripe.gif") }
BODY { background: url(\"Ulalume\".png) } /* quotes in URL escaped */
```

CSS Properties

Syntax for CSS Properties

Font Properties

- Font Family
- Font Style
- Font Variant
- Font Weight
- Font Size
- Font

Color and Background Properties

- Color
- Background Color
- Background Image
- Background Repeat
- Background Attachment
- Background Position
- Background

Text Properties

- Word Spacing
- Letter Spacing
- Text Decoration
- Vertical Alignment
- Text Transformation
- Text Alignment
- Text Indentation
- Line Height

Box Properties

- Top Margin
- Right Margin
- Bottom Margin
- Left Margin
- Margin
- Top Padding
- Right Padding
- Bottom Padding
- Left Padding
- Padding
- Top Border Width
- Right Border Width
- Bottom Border Width
- Left Border Width
- Border Width
- Border Color

Classification Properties

Units

Linking Style Sheets to HTML

There are many ways to link style sheets to HTML , each carrying its own advantages and disadvantages. New HTML elements and attributes have been introduced to allow easy incorporation of style sheets into HTML documents.

Linking to an External Style Sheet

An external style sheet may be linked to an HTML document through HTML's LINK element:

```
<LINK REL=StyleSheet HREF="color-8b.css" TYPE="text/css" TITLE="8-bit Color Style">
<LINK REL=StyleSheet HREF="color-24b.css" TYPE="text/css" TITLE="24-bit Color Style">
```

The <LINK> tag is placed in the document HEAD. The optional TYPE attribute is used to specify a media type--text/css for a Cascading Style Sheet--allowing browsers to ignore style sheet types that they do not support. The optional TITLE attribute is used to give a title to the style sheet. A title is especially important when multiple style sheets are used, since the title will allow the user to better select a style sheet. Multiple style sheets may be given with separate <LINK> tags; the order of the <LINK> tags is unimportant.

An external style sheet is ideal when the style is applied to numerous pages. With an external style sheet, an author could change the look of an entire site by simply changing one file. As well, most browsers will cache an external style sheet, thus avoiding a delay in page presentation once the style sheet is cached.

Microsoft Internet Explorer 3 does not support background images from linked style sheets. Given this bug, authors may wish to provide another mechanism for including a background image, such as embedding or inlining the style, or by using the BACKGROUND attribute of the BODY element.

Embedding a Style Sheet

A style sheet may be embedded in a document with the STYLE element:

```
<STYLE TYPE="text/css">
<!--
  BODY  { background: url(foo.gif) red; color: black }
  P EM  { background: yellow; color: black }
  .note { margin-left: 5em; margin-right: 5em }
-->
</STYLE>
```

The STYLE element is placed in the document HEAD . The optional TYPE attribute is used to specify a media type, as is its function with the LINK element. Similarly, the TITLE attribute may also be given to assign a title to the embedded style sheet.

Older browsers, unaware of the STYLE element, would normally show its contents as if they were part of the BODY, thus making the style sheet visible to the user. To prevent this, the contents of the STYLE element are contained within an SGML comment (<!-- comment -->), as in the example above.

An embedded style sheet should be used when a single document has a unique style. If the same style sheet is

used in multiple documents, then an external style sheet would be more appropriate.

Importing a Style Sheet

A style sheet may be imported into an HTML document with CSS's @import statement. This statement is used inside the STYLE element as follows:

```
<STYLE TYPE="text/css">
<!--
  @import url(http://www.htmlhelp.com/style.css);
  @import url(/stylesheets/punk.css);
  DT { background: yellow }
-->
</STYLE>
```

Note that other CSS rules may still be included in the STYLE element, but that all @import statements must occur at the start of the style sheet. Any rules specified in the style sheet itself override conflicting rules in the imported style sheets. For example, even if one of the imported style sheets contained DT { background: aqua }, definition terms would still have a yellow background.

The order in which the style sheets are imported is important in determining how they cascade. In the above example, if the style.css imported style sheet specified that STRONG elements be shown in red and the punk.css style sheet specified that STRONG elements be shown in yellow, then the latter rule would win out, and STRONG elements would be in yellow.

Imported style sheets would be useful for purposes of modularity. For example, a site may separate different style sheets by the selectors used. There may be a simple.css style sheet that gives rules for common elements such as BODY, P, H1, and H2. In addition, there may be an extra.css style sheet that gives rules for less common elements such as CODE, BLOCKQUOTE, and DFN. A tables.css style sheet may be used to define rules for table elements. These three style sheets could be included in HTML documents, as needed, with the @import statement.

Inlining Style

Style may be inlined by using the STYLE attribute. The STYLE attribute may be applied to any BODY element (including BODY itself). The attribute takes as its value any number of CSS declarations, where each declaration is separated by a semicolon. An example follows:

```
<P STYLE="color: red; font-family: 'New Century Schoolbook', serif">This paragraph is
styled in red with the New Century Schoolbook font,if available.</P>
```

Note that New Century Schoolbook is contained within single quotes in the STYLE attribute since double quotes are used to contain the style declarations.

Inlining styles loses many of the advantages of style sheets by mixing content with presentation. This method should be used sparingly, such as when a style is to be applied to a single occurrence of an element.

The CLASS Attribute

The **CLASS** attribute is used to specify the style class to which the element belongs. The style sheet may have created the punk and warning classes:

```
.punk     { color: lime; background: #ff80c0 }
P.warning { font-weight: bolder; color: red }
```

These classes could be referenced in HTML with the CLASS attribute:

```
<H1 CLASS=punk>Proprietary Extensions</H1>
<P CLASS=warning>Many proprietary extensions can have negative side-effects, both on
```

```
supporting and non-supporting browsers...
```

In this example, the punk class may be applied to any BODY element since it does not have an HTML element associated with it in the style sheet. Using the style sheet given above, the warning class may only be applied to the P element.

A good practice is to name classes according to their function rather than their appearance. The warning class in the above example could have been named red, but this name would become meaningless if the author decided to change the style of the class to a different color.

Classes can be a very effective method of applying different styles to structurally identical sections of an HTML document. For example, this page uses classes to give a different style to CSS code and HTML code.

The ID Attribute

The ID attribute is used to define a unique style for an element. A CSS rule such as

```
#wdg96 { font-size: large }
```

may be applied in HTML through the ID attribute:

```
<P ID=wdg96>Welcome to the Web Design Group!</P>
```

Each ID attribute must have a unique value over the document. The value must be an initial letter followed by letters, numbers, digits, hyphens, or periods. The letters are restricted to A-Z and a-z.

Use of ID for applying style is discouraged since the style may only be applied once in the document.

The SPAN Element

The SPAN element was introduced into HTML to allow authors to give style that could not be attached to a structural HTML element. SPAN may be used as a selector in a style sheet, and it also accepts the STYLE, CLASS, and ID attributes.

SPAN is a text-level element, so it may be used just as elements such as EM and STRONG in HTML. The important distinction is that while EM and STRONG carry structural meaning, SPAN has no such meaning. It exists purely to apply style, and so has no effect when the style sheet is disabled.

Some examples of SPAN follow:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML Experimental 19960712//EN">
<HTML>
<HEAD>
<TITLE>Example of SPAN</TITLE>
<STYLE TYPE="text/css">
<!--
  .firstwords { font-variant: small-caps }
-->
</STYLE>
</HEAD>
<BODY>
<P><SPAN CLASS=firstwords>The first few words</SPAN> of a paragraph could be in small-
caps. Style may also be inlined, such as to change the style of a word like <SPAN
STYLE="font-family: Arial">Arial</SPAN>.</P>
```

The DIV Element

The DIV element is similar to the SPAN element in function, with the important distinction that DIV (short for "division") is a block-level element. DIV may contain paragraphs, headings, tables, and even other divisions. This makes DIV ideal for marking different classes of containers, such as a chapter, abstract, or note. For example:

```
<DIV CLASS=note>
<H1>Divisions</H1>
<P>The DIV element is defined in HTML 3.2, but only the ALIGN attribute is permitted
in HTML 3.2. The next version of HTML, Cougar, adds the CLASS, STYLE, and ID
attributes, among others.</P>
<P>Since DIV may contain other block-level containers, it is useful for marking large
sections of a document, such as this note.</P>
<P>The closing tag is required.</P>
</DIV>
```

A Note about Validation

Few CSS-styled documents will validate at the HTML 3.2 (Wilbur) level HTML 3.2 does not define the SPAN element, nor the CLASS, STYLE, or ID attributes, and also lacks support for TYPE and TITLE attributes on the LINK and STYLE elements.

These style-related elements and attributes are not harmful to non-supporting browsers, as they are safely ignored. Documents using these elements and attributes may be validated against the Cougar DTD. While not many validators support Cougar yet, WebTechs Validation Service does.

Style Sheet Dependence

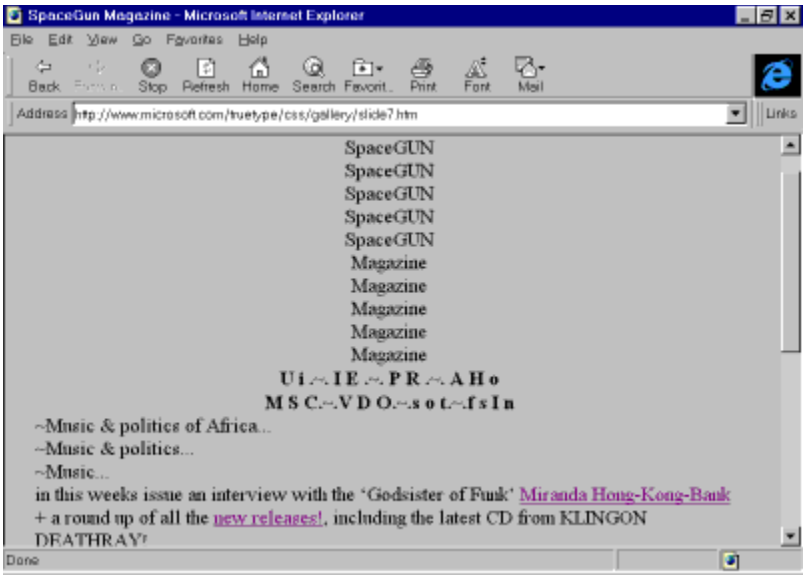
Cascading Style Sheets have still not come into widespread use, but misuse of CSS has already begun. Style sheets, when used properly, can be an effective tool for providing a unique and attractive presentation, while still allowing a page to be accessible to all users. However, as soon as a page's message becomes dependent on the style sheet, the page has become a failure on the Web.

Style sheets were designed to allow the author to influence the presentation, but not to control it. Style sheets can be overridden by users who may choose--or require--their own style sheet. For these reasons, authors who depend on a certain style will find their pages inaccessible to a significant portion of users.

Poor uses of style sheets are demonstrated in many so-called CSS galleries. One common hack that has appeared in various places is that of the "drop shadow." This is created by using negative margins, and involves a large amount of dependence on the style sheet. When the style sheet is removed, either at the user's discretion or by using a browser that does not support CSS, the page is often unusable.

As an example, take a look at SpaceGUN Magazine, a fictional magazine featured in Microsoft's CSS Gallery. Using Microsoft Internet Explorer 3.01 with style sheets enabled, the presentation is certainly unique and eye-catching:

But as soon as style sheets are disabled, the result is very different, even using the same browser:



Quite clearly, the page is unusable on the Web due to its drop shadows and other negative margin tricks that leave many readers with a jumbled mess. The example also shows that dependence on style sheets is a recipe for failure. A Web document is not effective if it is not accessible.

Style sheet designers should take care to always ensure that their Web pages are in no way dependent on the style sheet. Some authors have tried to use the WingDings font to produce glyphs without the hassle and added download time of actual images. While the motive of reducing download time is admirable, such authoring is dependent on the user having the WingDings font and on the style sheet being enabled--conditions that can never be assured on the World Wide Web.

CSS References

Cascading Style Sheets, level 1

The W3C's Recommendation gives the complete specification for CSS1.
http://www.w3.org/pub/WWW/TR/REC-CSS1

Quick Reference to Cascading Style Sheets, level 1

A handy quick reference for the properties and units of CSS1.
http://www.cwi.nl/%7Esteven/www/css1-qr.html

CSS draft specifications

The W3C's CSS home, with notes on browser and editor support, plus all the CSS drafts past and present.
http://www.w3.org/pub/WWW/Style/css/

CSS1 not supported in Internet Explorer 3.0

A look at CSS1 bugs and missing support in Microsoft Internet Explorer 3.0 for Windows 95 & NT 4.0.
http://www.shadow.net/%7Ebraden/nostyle/

The CSS1/MSIE 3.0 (Mac) Watch

A look at CSS1 bugs and missing support in Microsoft Internet Explorer 3.0 for the Macintosh.
http://www.cwru.edu/lit/homes/eam3/css1/msie-css1.html

CSS support in Amaya

Documentation of the CSS support in Amaya, the W3C's test-bed browser.
http://www.w3.org/pub/WWW/Amaya/User/CSS.html

Syntax Used in CSS Properties

<Foo>
Value of type Foo. Common types are discussed on the Units page.

Foo
A keyword that must appear literally (though without case sensitivity). Commas and slashes must also appear literally.

A B C
A must occur, then B, then C, in that order.

A | B
A or B must occur.

A || B
A or B or both must occur, in any order.

[Foo]
Brackets are used to group items together.

Foo*
Foo is repeated zero or more times.

Foo+
Foo is repeated one or more times.

Foo?
Foo is optional.

Foo{A,B}
Foo must occur at least A times and at most B times.

Font Family

Appearance:

font-family: [[<family-name> | <generic-family>],]* [<family-name> | <generic-family>]

Possible Values:

<family-name>

- Any font family name may be used

<generic-family>

- serif (e.g., Times)
- sans-serif (e.g., Arial or Helvetica)
- cursive (e.g., Zapf-Chancery)
- fantasy (e.g., Western)
- `monospace` (e.g., Courier)

Initial Value:

Determined by browser

Applies to:

All elements

Inherited:

Yes

Font families may be assigned by a specific font name or a generic font family. Obviously, defining a specific font will not be as likely to match as a generic font family. Multiple family assignments can be made, and if a specific font assignment is made it should be followed by a generic family name in case the first choice is not present.

A sample **font-family** declaration might look like this:

```
P { font-family: "New Century Schoolbook", Times, serif }
```

Notice that the first two assignments are specific type faces: **New Century Schoolbook** and **Times**. However, since both of them are serif fonts, the generic font family is listed as a backup in case the system does not have either of these but has another serif font which meets the qualifications.

Any font name containing whitespace must be quoted, with either single or double quotes.

The font family may also be given with the font property.

Font Style

Appearance:
font-style: <value>

Possible Values:
normal | italic | oblique

Initial Value:
normal

Applies to:
All elements

Inherited:
Yes

The **font-style** property defines that the font be displayed in one of three ways: normal, *italic* or oblique (slanted). A sample style sheet with **font-style** declarations might look like this:

```
H1 { font-style: oblique }
P  { font-style: normal }
```

Font Variant

Appearance:
font-variant: <value>

Possible Values:
normal | SMALL-CAPS

Initial Value:
normal

Applies to:
All elements

Inherited:
Yes

The **font-variant** property determines if the font is to display in **normal** or **small-caps**. SMALL-CAPS are displayed when all the letters of the word are in capitals with uppercase characters slightly larger than lowercase. Later versions of CSS may support additional variants such as condensed, expanded, small-caps numerals or other custom variants. An example of a font-variant assignment would be:

```
SPAN { font-variant: small-caps }
```

Font Weight

Appearance:
font-weight: <value>

Possible Values:
normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

Initial Value:
normal

Applies to:
All elements

Inherited:
Yes

The **font-weight** property is used to specify the weight of the font. The **bolder** and **lighter** values are relative to the inherited font weight, while the other values are absolute font weights.

Note: Since not all fonts have nine possible display weights some of the weights may be grouped together in assignment. If the specified weight is not available, an alternate will be chosen on the following basis:

- **500** may be switched with **400**, and vice versa
- **100**-**300** may be assigned to the next lighter weight, if any, or the next darker weight otherwise
- **600**-**900** may be assigned to the next darker weight, if any, or the next lighter weight otherwise

Some example **font-weight** assignments would be:

```
H1 { font-weight: 800 }
P  { font-weight: normal }
```

Font Size

Appearance:

font-size: <absolute-size> | <relative-size> | <length> | <percentage>

Possible Values:

<absolute-size>

xx-small | x-small | small | medium | large | x-large | xx-large

<relative-size>

larger | smaller

<length>

<percentage> (in relation to parent element)

Initial Value:

medium

Applies to:

All elements

Inherited:

Yes

The **font-size** property is used to modify the size of the displayed font. Lengths should be used sparingly, due to their weakness in adjusting to varying resolutions.

Some example size assignments would be:

```
H1     { font-size: large }
P      { font-size: 12pt }
LI     { font-size: 90% }
STRONG { font-size: larger }
```

Authors should be aware that Microsoft Internet Explorer 3.x incorrectly applies percentage values relative to its default font size for the selector, rather than relative to the parent element's font size. This makes rules like:

```
H1 { font-size: 200% }
```

dangerous in that the size will be twice **IE's** default font size for level-one headings, rather than twice the parent element's font size. In this case, BODY would most likely be the parent element, and it would likely define a "medium" font size, whereas the default level-one heading font size imposed by IE would probably be considered "xx-large."

Given this bug, authors should take care in using percentage values for **font-size**.

Font

Appearance:
font: <value>

Possible Values:
[<font-style> || <font-variant> || <font-weight>]? <font-size> [/ <line-height>]? <font-family>

Initial Value:
Not defined

Applies to:
All elements

Inherited:
Yes

The **font** property may be used as a shorthand for the various font properties, as well as the line height. For example:

```
P { font: italic bold 12pt/14pt Times, serif }
```

specifies paragraphs with a bold and italic Times or serif font with a size of 12 points and a line height of 14 points.

Color

Appearance:
color: <color>

Initial Value:
Determined by browser

Applies to:
All elements

Inherited:
Yes

The color property allows authors to specify the color of an element. See the Units section for color value descriptions. Some example color rules include:

```
H1 { color: blue }
H2 { color: #000080 }
H3 { color: #0c0 }
```

Background Color

Appearance:
background-color: <value>

Possible Values:
<color> | transparent

Initial Value:
transparent

Applies to:
All elements

Inherited:
No

The **background-color** property sets the background color of an element. For example:

```
BODY { background-color: white }
H1   { background-color: #000080 }
```

Authors may also use the shorthand background property, which is currently better supported than the **background-color** property.

Background Image

Appearance:
background-image: <value>

Possible Values:
<url> | none

Initial Value:
none

Applies to:
All elements

Inherited:No

The **background-image** property sets the background image of an element. For example:

```
BODY { background-image: url(/images/foo.gif) }
P    { background-image: url(http://www.htmlhelp.com/bg.png) }
```

When a background image is defined, a similar background color should also be defined for those not loading images.

Authors may also use the shorthand **background** property, which is currently better supported than the **background-image** property.

Background Repeat

Appearance:
background-repeat: <value>

Possible Values:
repeat | repeat-x | repeat-y | no-repeat

Initial Value:
repeat

Applies to:
All elements

Inherited:
No

The **background-repeat** property determines how a specified background image is repeated. The **repeat-x** value will repeat the image horizontally while the **repeat-y** value will repeat the image vertically. For example:

```
BODY { background: white url(candybar.gif);
       background-repeat: repeat-x }
```

In the above example, the image will only be tiled horizontally.

Authors may also use the shorthand **background** property, which is currently better supported than the **background-repeat** property.

Background Attachment

Appearance:
background-attachment: <value>

Possible Values:
scroll | fixed

Initial Value:
scroll

Applies to:
All elements

Inherited:
No

The **background-attachment** property determines if a specified background image will scroll with the content or be fixed with regard to the canvas. For example, the following specifies a fixed background image:

```
BODY { background: white url(candybar.gif);
       background-attachment: fixed }
```

Authors may also use the shorthand **background** property, which is currently better supported than the **background-attachment** property.

Background Position

Appearance:
background-position: <value>

Possible Values:
[<percentage> | <length>]{1,2} | [top | center | bottom] || [left | center | right]

Initial Value:
0% 0%

Applies to:
Block-level and replaced elements

Inherited:
No

The **background-position** property gives the initial position of a specified background image. This property may only be applied to block-level elements and replaced elements. (A replaced element is one for which only the intrinsic dimensions are known; HTML replaced elements include IMG, INPUT, TEXTAREA, SELECT, and OBJECT.)

The easiest way to assign a background position is with keywords:

- Horizontal keywords (left, center, right)
- Vertical keywords (top, center, bottom)

Percentages and lengths may also be used to assign the position of the background image. Percentages are relative to the size of the element. Although lengths are allowed, they are not recommended due to their inherent weakness in dealing with differing display resolutions.

When using percentages or lengths, the horizontal position is specified first, followed by the vertical position. A value such as 20% 65% specifies that the point 20% across and 65% down the image be placed at the point 20% across and 65% down the element. A value such as 5px 10px specifies that the upper left corner of the image be placed 5 pixels to the right of and 10 pixels below the upper left corner of the element.

If only the horizontal value is given, the vertical position will be 50%. Combinations of lengths and percentages are allowed, as are negative positions. For example, 10% -2cm is permitted. However, percentages and lengths cannot be combined with keywords.

The keywords are interpreted as follows:

- top left = left top = 0% 0%
- top = top center = center top = 50% 0%
- right top = top right = 100% 0%
- left = left center = center left = 0% 50%
- center = center center = 50% 50%
- right = right center = center right = 100% 50%
- bottom left = left bottom = 0% 100%
- bottom = bottom center = center bottom = 50% 100%
- bottom right = right bottom = 100% 100%

If the background image is fixed with regard to the canvas, the image is placed relative to the canvas instead of the element.

Authors may also use the shorthand **background** property, which is currently better supported than the **background-position** property.

Background

Appearance:
background: <value>

Possible Values:
<background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **background** property is a shorthand for the more specific background-related properties. Some examples of background declarations follow:

```
BODY       { background: white url(http://www.htmlhelp.com/foo.gif) }
BLOCKQUOTE { background: #7fffd4 }
P          { background: url(../backgrounds/pawn.png) #f0f8ff fixed }
TABLE      { background: #0c0 url(leaves.jpg) no-repeat bottom right }
```

A value not specified will receive its initial value. For example, in the first three rules above, the **background-position** property will be set to 0% 0%.

Word Spacing

Appearance:
word-spacing: <value>

Possible Values:
normal | <length>

Initial Value:
normal

Applies to:
All elements

Inherited:
Yes

The **word-spacing** property defines an additional amount of space between words. The value must be in the length format; negative values are permitted.

Examples:

```
P EM   { word-spacing: 0.4em }
P.note { word-spacing: -0.2em }
```

Letter Spacing

Appearance:
letter-spacing: <value>

Possible Values:
normal | <length>

Initial Value:
normal

Applies to:
All elements

Inherited:
Yes

The **letter-spacing** property defines an additional amount of space between characters. The value must be in the length format; negative values are permitted. A setting of 0 will prevent justification.

Examples:

```
H1     { letter-spacing: 0.1em }
P.note { letter-spacing: -0.1em }
```

Text Decoration

Appearance:
text-decoration: <value>

Possible Values:
none | [underline || overline || line-through || blink]

Initial Value:
none

Applies to:
All elements

Inherited:
No

The **text-decoration** property allows text to be decorated through one of five properties: underline, overline, line-through, blink, or the default, none.

For example, one can suggest that links not be underlined with

```
A:link, A:visited, A:active { text-decoration: none }
```

Vertical Alignment

Appearance:
vertical-align: <value>

Possible Values:
baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage>

Initial Value:
baseline

Applies to:
Inline elements

Inherited:
No

The **vertical-align** property may be used to alter the vertical positioning of an inline element, relative to its parent element or to the element's line. (An inline element is one which has no line break before and after it, for example, EM, A, and IMG in HTML.)

The value may be a percentage relative to the element's line-height property, which would raise the element's baseline the specified amount above the parent's baseline. Negative values are permitted.

The value may also be a keyword. The following keywords affect the positioning relative to the parent element:

- baseline (align baselines of element and parent)
- middle (align vertical midpoint of element with baseline plus half the x-height--the height of the letter "x"--of the parent)
- sub (subscript)
- super (superscript)
- text-top (align tops of element and parent's font)
- text-bottom (align bottoms of element and parent's font)

The keywords affecting the positioning relative to the element's line are

- top (align top of element with tallest element on the line)
- bottom (align bottom of element with lowest element on the line)

The **vertical-align** property is particularly useful for aligning images. Some examples follow:

```
IMG.middle { vertical-align: middle }
IMG        { vertical-align: 50% }
.exponent  { vertical-align: super }
```

Text Transformation

Appearance:
text-transform: <value>

Possible Values:
none | capitalize | uppercase | lowercase

Initial Value:
none

Applies to:
All elements

Inherited:
Yes

The **text-transform** property allows text to be transformed by one of four properties:

- Capitalize (capitalizes first character of each word)
- UPPERCASE (capitalizes all characters of each word)
- lowercase (uses small letters for all characters of each word)
- none (the initial value)

Examples:

```
H1 { text-transform: uppercase }
H2 { text-transform: capitalize }
```

The **text-transform** property should only be used to express a stylistic desire. It would be inappropriate, for example, to use text-transform to capitalize a list of countries or names.

Text Alignment

Appearance:
text-align: <value>

Possible Values:
left | right | center | justify

Initial Value:
Determined by browser

Applies to:
Block-level elements

Inherited:
Yes

The **text-align** property can be applied to block-level elements (P, H1, etc.) to give the alignment of the element's text. This property is similar in function to HTML's ALIGN attribute on paragraphs, headings, and divisions.

Some examples follow:

```
H1          { text-align: center }
P.newspaper { text-align: justify }
```

Text Indentation

Appearance:
text-indent: <value>

Possible Values:
<length> | <percentage>

Initial Value:
0

Applies to:
Block-level elements

Inherited:
Yes

The **text-indent** property can be applied to block-level elements (P, H1, etc.) to define the amount of indentation that the first line of the element should receive. The value must be a length or a percentage; percentages refer to the parent element's width. A common use of **text-indent** would be to indent a paragraph:

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

Line Height

Appearance:
line-height: <value>

Possible Values:
normal | <number> | <length> | <percentage>

Initial Value:
normal

Applies to:
All elements

Inherited:
Yes

The **line-height** property will accept a value to control the spacing between baselines of text. When the value is a number, the line height is calculated by multiplying the element's font size by the number. Percentage values are relative to the element's font size. Negative values are not permitted.

Line height may also be given in the font property along with a font size. The **line-height** property could be used to double space text:

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

Top Margin

Appearance:
margin-top: <value>

Possible Values:
<length> | <percentage> | auto

Initial Value:
0

Applies to:
All elements

Inherited:
No

The property sets the **top margin** of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted.

For example, the following rule would eliminate the top margin of a document:

```
BODY { margin-top: 0 }
```

Note: adjoining vertical margins are collapsed to use the maximum of the margin values.

Right Margin

Appearance:
margin-right: <value>

Possible Values:
<length> | <percentage> | auto

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **margin-right** property sets the right margin of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted.

Example:

```
P.narrow { margin-right: 50% }
```

Note: adjoining horizontal margins are not collapsed.

Bottom Margin

Appearance:
margin-bottom: <value>

Possible Values:
<length> | <percentage> | auto

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **margin-bottom** property sets the bottom margin of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted.

Example:

```
DT { margin-bottom: 3em }
```

Note: adjoining vertical margins are collapsed to use the maximum of the margin values.

Left Margin

Appearance:
margin-left: <value>

Possible Values:
<length> | <percentage> | auto

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **margin-left** property sets the left margin of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted.

Example:

```
ADDRESS { margin-left: 50% }
```

Note: adjoining horizontal margins are not collapsed.

Margin

Appearance:
margin: <value>

Possible Values:
[<length> | <percentage> | auto]{1,4}

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **margin** property sets the margins of an element by specifying between one and four values, where each value is a length, a percentage, or auto. Percentage values refer to the parent element's width. Negative margins are permitted.

If four values are given, they apply to top, right, bottom, and left margin, respectively. If one value is given, it applies to all sides. If two or three values are given, the missing values are taken from the opposite side.

Examples of margin declarations include:

```
BODY { margin: 5em }             /* all margins 5em */
P    { margin: 2em 4em }         /* top and bottom margins 2em,
                                    left and right margins 4em */
DIV  { margin: 1em 2em 3em 4em } /* top margin 1em,
                                    right margin 2em,
                                    bottom margin 3em,
                                    left margin 4em */
```

Note: adjoining vertical margins are collapsed to use the maximum of the margin values. Horizontal margins are not collapsed.

Using the **margin** property allows one to set all margins; alternatively, the properties margin-top, margin-bottom, margin-left, and margin-right may be used.

Top Padding

Appearance:
padding-top: <value>

Possible Values:
<length> | <percentage>

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **padding-top** property describes how much space to put between the top border and the content of the selector. The value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are not permitted.

Right Padding

Appearance:
padding-right: <value>

Possible Values:
<length> | <percentage>

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **padding-right** property describes how much space to put between the right border and the content of the selector. The value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are not permitted.

Bottom Padding

Appearance:
padding-bottom: <value>

Possible Values:
<length> | <percentage>

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **padding-bottom** property describes how much space to put between the bottom border and the content of the selector. The value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are not permitted.

Left Padding

Appearance:
padding-left: <value>

Possible Values:
<length> | <percentage>

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **padding-left** property describes how much space to put between the left border and the content of the selector. The value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are not permitted.

Padding

Appearance:
padding: <value>

Possible Values:
[<length> | <percentage>]{1,4}

Initial Value:
0

Applies to:
All elements

Inherited:
No

The **padding** property is a shorthand for the **padding-top**, **padding-right**, **padding-bottom**, and **padding-left** properties.

An element's padding is the amount of space between the border and the content of the element. Between one and four values are given, where each value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are not permitted.

If four values are given, they apply to top, right, bottom, and left padding, respectively. If one value is given, it applies to all sides. If two or three values are given, the missing values are taken from the opposite side.

For example, the following rule sets the top padding to 2em, the right padding to 4em, the bottom padding to 5em, and the left padding to 4em:

```
BLOCKQUOTE { padding: 2em 4em 5em }
```

Top Border Width

Appearance:
border-top-width: <value>

Possible Values:
thin | medium | thick | <length>

Initial Value:
medium

Applies to:
All elements

Inherited:
No

The **border-top-width** property is used to specify the width of an element's top border. The value may be one of three keywords, which are not affected by font size, or a length, which can be used to achieve relative widths. Negative values are not allowed.

One may also use the **border-top**, **border-width**, or **border** shorthand properties.

Right Border Width

Appearance:
border-right-width: <value>

Possible Values:
thin | medium | thick | <length>

Initial Value:
medium

Applies to:
All elements

Inherited:
No

The **border-right-width** property is used to specify the width of an element's right border. The value may be one of three keywords, which are not affected by font size, or a length, which can be used to achieve relative widths. Negative values are not allowed.

One may also use the **border-right**, **border-width**, or **border** shorthand properties.

Bottom Border Width

Appearance:
border-bottom-width: <value>

Possible Values:
thin | medium | thick | <length>

Initial Value:
medium

Applies to:
All elements

Inherited:
No

The **border-bottom-width** property is used to specify the width of an element's bottom border. The value may be one of three keywords, which are not affected by font size, or a length, which can be used to achieve relative widths. Negative values are not allowed.

One may also use the **border-bottom**, **border-width**, or **border** shorthand properties.

Left Border Width

Appearance:
border-left-width: <value>

Possible Values:
thin | medium | thick | <length>

Initial Value:
medium

Applies to:
All elements

Inherited:
No

The **border-left-width** property is used to specify the width of an element's left border. The value may be one of three keywords, which are not affected by font size, or a length, which can be used to achieve relative widths. Negative values are not allowed.

One may also use the **border-left**, **border-width**, or **border** shorthand properties.

Border Width

Appearance:
border-width: <value>

Possible Values:
[thin | medium | thick | <length>]{1,4}

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **border-width** property is used to set the border width of an element by specifying between one and four values, where each value is a keyword or a length. Negative lengths are not permitted.

If four values are given, they apply to top, right, bottom, and left border width, respectively. If one value is given, it applies to all sides. If two or three values are given, the missing values are taken from the opposite side.

This property is a shorthand for the **border-top-width**, **border-right-width**, **border-bottom-width**, and **border-left-width** properties.

One may also use the **border** shorthand property.

Border Color

Appearance:
border-color: <value>

Possible Values:
<color>{1,4}

Initial Value:
The value of the color property

Applies to:
All elements

Inherited:
No

The **border-color** property sets the color of an element's border. Between one and four color values are specified. If four values are given, they apply to top, right, bottom, and left border color, respectively. If one value is given, it applies to all sides. If two or three values are given, the missing values are taken from the opposite side.

One may also use the **border** shorthand property.

Border Style

Appearance:
border-style: <value>

Possible Values:
[none | dotted | dashed | solid | double | groove | ridge | inset | outset]{1,4}

Initial Value:
none

Applies to:
All elements

Inherited:
No

The **border-style** property sets the style of an element's border. This property must be specified for the border to be visible.

Between one and four keywords are specified. If four values are given, they apply to top, right, bottom, and left border style, respectively. If one value is given, it applies to all sides. If two or three values are given, the missing values are taken from the opposite side.

One may also use the **border** shorthand property.

Top Border

Appearance:
border-top: <value>

Possible Values:
<border-top-width> || <border-style> || <color>

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **border-top** property is a shorthand for setting the width, style, and color of an element's top border.

Note: that only one **border-style** value may be given.

One may also use the **border** shorthand property.

Right Border

Appearance:
border-right: <value>

Possible Values:
<border-right-width> || <border-style> || <color>

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **border-right** property is a shorthand for setting the width, style, and color of an element's right border.

Note: that only one **border-style** value may be given.

One may also use the **border** shorthand property.

Bottom Border

Appearance:
border-bottom: <value>

Possible Values:
<border-bottom-width> || <border-style> || <color>

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **border-bottom** property is a shorthand for setting the width, style, and color of an element's bottom border.

Note: that only one **border-style** value may be given.

One may also use the **border** shorthand property.

Left Border

Appearance:
border-left: <value>

Possible Values:
<border-left-width> || <border-style> || <color>

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **border-left** property is a shorthand for setting the width, style, and color of an element's left border.

Note: that only one **border-style** value may be given.

One may also use the **border** shorthand property.

Border

Appearance:
border: <value>

Possible Values:
<border-width> || <border-style> || <color>

Initial Value:
Not defined

Applies to:
All elements

Inherited:
No

The **border** property is a shorthand for setting the width, style, and color of an element's border.

Examples of **border** declarations include:

```
H2        { border: groove 3em }
A:link    { border: solid blue }
A:visited { border: thin dotted #800080 }
A:active  { border: thick double red }
```

The border property can only set all four borders; only one border width and border style may be given. To give different values to an element's four borders, an author must use one or more of the border-top, border-right, border-bottom, border-left, border-color, border-width, border-style, border-top-width, border-right-width, border-bottom-width, or border-left-width properties.

Width

Appearance:
width: <value>

Possible Values:
<length> | <percentage> | auto

Initial Value:
auto

Applies to:
Block-level and replaced elements

Inherited:
No

Each block-level or replaced element can be given a **width**, specified as a length, a percentage, or as auto. (A replaced element is one for which only the intrinsic dimensions are known; HTML replaced elements include IMG, INPUT, TEXTAREA, SELECT, and OBJECT.) The initial value for the width property is auto, which results in the element's intrinsic width (i.e., the width of the element itself, for example the width of an image). Percentages refer to the parent element's width. Negative values are not allowed.

This property could be used to give common widths to some INPUT elements, such as submit and reset buttons:

```
INPUT.button { width: 10em }
```

Height

Appearance:
height: <value>

Possible Values:
<length> | auto

Initial Value:
auto

Applies to:
Block-level and replaced elements

Inherited:
No

Each block-level or replaced element can be given a **height**, specified as a length or as auto. (A replaced element is one for which only the intrinsic dimensions are known; HTML replaced elements include IMG, INPUT, TEXTAREA, SELECT, and OBJECT.) The initial value for the height property is auto, which results in the element's intrinsic height (i.e., the height of the element itself, for example the height of an image). Negative lengths are not allowed.

As with the **width** property, **height** can be used to scale an image:

```
IMG.foo { width: 40px; height: 40px }
```

In most cases, authors are advised to scale the image in an image editing program, since browsers will not likely scale images with high quality, and since scaling down causes the user to download an unnecessarily large file. However, scaling through the **width** and **height** properties is a useful option for user-defined style sheets in order to overcome vision problems.

Float

Appearance:
float: <value>

Possible Values:
left | right | none

Initial Value:
none

Applies to:
All elements

Inherited:
No

The **float** property allows authors to wrap text around an element. This is identical in purpose to HTML 3.2's ALIGN=left and ALIGN=right for the IMG element, but CSS1 allows all elements to "float," not just the images and tables that HTML 3.2 allows.

Clear

Appearance:
clear: <value>

Possible Values:
none | left | right | both

Initial Value:
none

Applies to:
All elements

Inherited:
No

The **clear** property specifies if an element allows floating elements to its sides. A value of left moves the element below any floating element on its left; right acts similarly for floating elements on the right. Other values are none, which is the initial value, and both, which moves the element below floating elements on both of its sides. This property is similar in function to HTML 3.2's <BR CLEAR=left|right|all|none>, but it can be applied to all elements.

Display

Appearance:
display: <value>

Possible Values:
block | inline | list-item | none

Initial Value:
block

Applies to:
All elements

Inherited:
No

The **display** property is used to define an element with one of four values:

- block (a line break before and after the element)
- inline (no line break before and after the element)
- list-item (same as block except a list-item marker is added)
- none (no display)

Each element typically is given a default display value by the browser, based on suggested rendering in the HTML specification.

The **display** property can be dangerous because of its ability to display elements in what would otherwise be an improper format. The use of the value none will turn off display of the element to which it is assigned, including any children elements!

Whitespace

Appearance:
white-space: <value>

Possible Values:
normal | pre | nowrap

Initial Value:
normal

Applies to:
Block-level elements

Inherited:
Yes

The **white-space** property will determine how spaces within the element are treated. This property takes one of three values:

- normal (collapses multiple spaces into one)
- pre (does not collapse multiple spaces)
- nowrap (does not allow line wrapping without a
 tag)

List Style Type

Appearance:
list-style-type: <value>

Possible Values:
disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none

Initial Value:
disc

Applies to:
Elements with display value list-item

Inherited:
Yes

The **list-style-type** property specifies the type of list-item marker, and is used if list-style-image is *none* or if image loading is turned off.

Examples:

```
LI.square { list-style-type: square }
UL.plain  { list-style-type: none }
OL        { list-style-type: upper-alpha }  /* A B C D E etc. */
OL OL     { list-style-type: decimal }      /* 1 2 3 4 5 etc. */
OL OL OL  { list-style-type: lower-roman }  /* i ii iii iv v etc. */
```

List Style Image

Appearance:
list-style-image: <value>

Possible Values:
<url> | none

Initial Value:
none

Applies to:
Elements with display value list-item

Inherited:
Yes

The **list-style-image** property specifies the image that will be used as list-item marker when image loading is turned on, replacing the marker specified in the **list-style-type** property.

Examples:

```
UL.check { list-style-image: url(/LI-markers/checkmark.gif) }
UL LI.x  { list-style-image: url(x.png) }
```

List Style Position

Appearance:
list-style-position: <value>

Possible Values:
inside | outside

Initial Value:
outside

Applies to:
Elements with display value list-item

Inherited:
Yes

The **list-style-position** property takes the value *inside* or *outside*, with *outside* being the default. This property determines where the marker is placed in regard to the list item. If the value inside is used, the lines will wrap under the marker instead of being indented. An example rendering is:

Outside rendering:

```
* List item 1
  second line of list item
```

Inside rendering:

```
* List item 1
second line of list item
```

List Style

Appearance:
list-style: <value>

Possible Values:
<list-style-type> || <list-style-position> || <url>

Initial Value:
Not defined

Applies to:
Elements with display value list-item

Inherited:
Yes

The **list-style property** is a shorthand for the **list-style-type**, **list-style-position**, and **list-style-image** properties.

Examples:

```
LI.square { list-style: square inside }
UL.plain  { list-style: none }
UL.check  { list-style: url(/LI-markers/checkmark.gif) circle }
OL        { list-style: upper-alpha }
OL OL     { list-style: lower-roman inside }
```

liam@htmlhelp.com

liam@htmlhelp.com
http://www.htmlhelp.com/~liam/

john@htmlhelp.com

john@htmlhelp.com
http://www.htmlhelp.com/~john/

Arnoud "Galactus" Engelfriet

galactus@htmlhelp.com
http://www.htmlhelp.com/~galactus/

Ian Butler

ian@htmlhelp.com
http://www.htmlhelp.com/~ian/

Tina Marie Holmboe

tina@htmlhelp.com
http://www.htmlhelp.com/~tina/

Nick Kew

nick@htmlhelp.com
http://www.htmlhelp.com/~nick/

HTML 3.2

http://www.htmlhelp.com/reference/wilbur/
http://www.w3.org/pub/WWW/MarkUp/Wilbur/

Amaya

http://www.w3.org/pub/WWW/Amaya/

Emacs-W3

http://www.cs.indiana.edu/elisp/w3/docs.html

CSS Positioning

http://www.w3.org/pub/WWW/TR/WD-positioning

Fonts and the Web

http://www.w3.org/pub/WWW/Fonts/

ACSS

http://www.w3.org/pub/WWW/Style/CSS/Speech/NOTE-ACSS

Printing support

http://www.w3.org/pub/WWW/Printing/

Layout

http://www.w3.org/pub/WWW/TR/WD-layout

Arena

http://www.yggdrasil.com/Products/Arena/

MSIE 3

http://www.microsoft.com/ie/default.asp

Navigator 4.0

http://home.netscape.com/comprod/products/communicator/navigator.html

Style

http://htmlhelp.com/reference/wilbur/head/style.html

CODE

http://htmlhelp.com/reference/wilbur/phrase/code.html

Nick Kew

nick@htmlhelp.com
http://www.htmlhelp.com/~nick/

BODY element

http://www.htmlhelp.com/reference/wilbur/body/body.html

HR - horizontal rule

http://www.htmlhelp.com/reference/wilbur/block/hr.html

DD - definition

http://www.htmlhelp.com/reference/wilbur/list/dd.html

Paragraph

http://www.htmlhelp.com/reference/wilbur/block/p.html

Emphasis

http://www.htmlhelp.com/reference/wilbur/phrase/em.html

LINK element

http://www.htmlhelp.com/reference/wilbur/head/link.html

HEAD element

http://www.htmlhelp.com/reference/wilbur/head/head.html

TITLE element

http://www.htmlhelp.com/reference/wilbur/head/title.html

BLOCKQUOTE

http://www.htmlhelp.com/reference/wilbur/block/blockquote.html

DFN - definition

http://www.htmlhelp.com/reference/wilbur/phrase/dfn.html

DIV - logical division

http://www.htmlhelp.com/reference/wilbur/block/div.html

WebTechs

http://www.webtechs.com/html-val-svc/

IMG - images

http://www.htmlhelp.com/reference/wilbur/special/img.html

INPUT element

http://www.htmlhelp.com/reference/wilbur/form/input.html

TEXTAREA

http://www.htmlhelp.com/reference/wilbur/form/textarea.html

A - anchor element

http://www.htmlhelp.com/reference/wilbur/special/a.html