

Cascading Style Sheets

This document is intended to present a high level overview of cascading style sheets. It will give you an overview of what style sheets are and how they can be used with Web documents. More detailed information is available from many places on the Web. References to some of these sites will be provided at the end of this document.

It is best if you do not think of style sheets as a "sheet", but simply as an instruction that is included as part of an HTML element. While a set of style sheet instructions may be included on a separate "sheet" or "page", they are actually just a set of instructions.

- A. What are Style Sheet instructions ?
 - a. Very basically, Style Sheets are a set of instructions that allow you to override default browser settings.
 - b. A style sheet is a page, or part of a page, that defines one or more rules that govern the appearance, or style, about any page or portions of pages linked to that style sheet.

Why use Style Sheets?

- c. Allow you to have precise control of the appearance of your web page.
- d. Reduce bandwidth
- e. Provides an easy venue to change the appearance of an entire web page or a web site.
- f. Friendly to all browsers (although some applications of styles may vary from browser to browser).
- g. Allows compliancy to future web standards.

Types of Style Sheets:

- A. Inline
 - a. An Inline style sheet is simply a style sheet instruction that it is included as part of the HTML element that it affects.
- B. Embedded
 - a. Embedded style sheet instructions are included in the document between a pair of <STYLE> and </STYLE> tags.
- C. Linked
 - a. Also called external or master style sheets
 - b. The style instructions included in a master style sheet form a completely separate document and apply the same formatting to any other documents that link to it.

- c. Uses the <LINK> tag.
- D. Imported
 - a. Imported style sheets may be used in a CSS file or inside the **STYLE** element.
 - b. Think of it as a style within a style
 - c. Uses the **@import** statement.

Style Sheet instruction syntax

Basic syntax

Selector {declaration}

The selector indicates which HTML element [i.e. tag] is affected by the rule, and the declaration indicates which style specification(s) apply to that element. Points to remember:

Element(s) {Property₁ : Value₁ ; Property₂ : Value_{2a} , Value_{2b} }

- You may use either upper- or lowercase; lowercase appears to be "traditional" for style sheets
- The declaration is always placed inside curly braces
- Declarations may contain one or more different properties of the selector to which it refers.
- More than one selector may share the same declaration.
 - If they are type selectors, separate them with commas
 - If they are contextual selectors, separate them with spaces
- Properties are always separated from values by colons.
- Properties are always either one word or compound words separated by hyphens, e.g. *color*, *font-family*.
- Values are usually single words, but if multiple words (e.g. a font name) they can be enclosed in quotation marks.
- Multiple values are separated by commas.
- Multiple property/value combinations are separated by semi-colons.

Varieties of Selectors

ID Selectors

There are four kinds of selectors: **Type**, **Attribute**, **Contextual**, and **External**.

Type selectors refer to individual HTML elements (tags). In other words, the style property or properties listed in the declaration apply to a specific kind of HTML tag, e.g. headers or paragraphs or lists. The examples used earlier are all type selectors. If multiple elements share a common set of properties, they should be separated by commas, e.g.

P, H1, UL { color: black }

Attribute selectors refer to generic classes of style properties that can be applied to more than one HTML element. This is similar to the way that *attributes* modify the basic nature of what HTML tags do. For example, a common tag attribute is **ALIGN=**, which can be applied to headers, paragraphs, tables, images, horizontal rules, and other HTML tags. In each case the **ALIGN=** attribute controls the horizontal positioning of the text or object to which it refers. In much the same way an attribute selector refers to a generic class of style properties that can be applied to more than one HTML element. You define attribute selectors by using a period as a *flag character* to mark the following name as a class definition. Such names can consist of single words only. Then you can apply that name to any appropriate HTML tag. For example:

.germantext {font-family: courier, monospace ; text-align: center ; font-size: 16pt ; color: blue}

The above class definition could be applied to any HTML element, perhaps indicating by the use of the mono-spaced fonts and the color blue that the information displayed was in German. You would do this by appending the class name to a specific instance of an HTML tag in your document:

<H3 class=germantext>Eine Deutsche Maxime</H3>

<P class=germantext>Warum so einfach wenn es so schön kompliziert geht?</P>

Eine Deutsche Maxime

Warum so einfach wenn es so schön kompliziert geht?

Why do this when you could just as well use the **** tag to accomplish the same thing? Well, suppose you had a lengthy document that was a mixture of English and German, with the German text shown in blue Courier tagged with **** tags. Suddenly, your

boss tells you to change from blue Courier to green Arial. You would laboriously have to search for each instance of German text in your document and make the necessary changes for each one. However, if you had used an attribute selector instead, all you would have to do is make the change in the attribute declaration and all the affected text would change instantly.

Contextual selectors are used to apply styles to tags used in a certain context or under certain conditions. For example, you might want normal paragraph text to appear in 12-point Times New Roman with any included *emphasized text* appearing in red.

However, when text is block quoted (like this), you might prefer *emphasized text* to appear blue. So in the context of block quoting, the **** tag is handled *differently* from how it is in the rest of the document.

Here is the style sheet information that would accomplish this effect. Note that the selectors are *not* separated by commas:

```
EM {color: red}  
BLOCKQUOTE EM {color: blue}
```

What this says is that within the context of a normal paragraph, text surrounded by **** **** tags should appear *in red*. But if a section of block quoted text happens to appear, the rule for emphasized text in the context of a block quote should override the previous instruction and appear in blue. If the specific instruction for block quoted text was not included, emphasized text within a block quote would appear in the same color as that for emphasized text in a normal paragraph.

External selectors refer to elements and attributes that depend on some condition external to the HTML tags themselves to have meaning. An example might be hypertext anchors, where the anchor text in your document could have a different color depending on whether the anchor has been visited or not by the user.

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**.

Non-ID Selectors

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.

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 }
```

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="style.css" TYPE="text/css"  
MEDIA=screen>  
<LINK REL=StyleSheet HREF="color-8b.css" TYPE="text/css"  
TITLE="8-bit Color Style" MEDIA="screen, print">  
<LINK REL="Alternate StyleSheet" HREF="color-24b.css"  
TYPE="text/css" TITLE="24-bit Color Style" MEDIA="screen,  
print">  
<LINK REL=StyleSheet HREF="aural.css" TYPE="text/css"  
MEDIA=aural>
```

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. Configuring the server to send **text/css** as the **Content-type** for CSS files is also a good idea.

External style sheets should *not* contain any HTML tags like **<HEAD>** or **<STYLE>**. The style sheet should consist merely of style rules or statements. A file consisting solely of

```
P { margin: 2em }
```

could be used as an external style sheet.

The **<LINK>** tag also takes an optional **MEDIA** attribute, which specifies the medium or media to which the style sheet should be applied. Possible values are

- **screen** (the default value), for presentation on non-paged computer screens;
- **print**, for output to a printer;
- **projection**, for projected presentations;
- **aural**, for speech synthesizers;
- **braille**, for presentation on braille tactile feedback devices;
- **tty**, for character cell displays (using a fixed-pitch font);
- **tv**, for televisions;
- **all**, for all output devices.

Multiple media are specified through a comma-separated list or the value **all**.

[Netscape Navigator 4.x](#) incorrectly ignores any linked or **embedded** style sheets declared with **MEDIA** values other than **screen**. For example,

MEDIA="screen, projection" will cause the style sheet to be ignored by Navigator 4.x, even if the presentation device is a computer screen. Navigator 4.x also ignores style sheets declared with **MEDIA=all**.

The **REL** attribute is used to define the relationship between the linked file and the HTML document. **REL=StyleSheet** specifies a *persistent* or *preferred* style while **REL="Alternate StyleSheet"** defines an *alternate* style. A **persistent** style is one that is always applied when style sheets are enabled. The absence of the **TITLE** attribute, as in the first **<LINK>** tag in the [example](#), defines a persistent style.

A **preferred** style is one that is automatically applied, such as in the second **<LINK>** tag in the [example](#). The combination of **REL=StyleSheet** and a

TITLE attribute specifies a preferred style. Authors cannot specify more than one preferred style.

An **alternate** style is indicated by **REL="Alternate StyleSheet"**. The third **<LINK>** tag in the [example](#) defines an alternate style, which the user could choose to replace the preferred style sheet.

Note that current browsers generally lack the ability to choose alternate styles.

A single style may also be given through multiple style sheets:

```
<LINK REL=StyleSheet HREF="basics.css"
TITLE="Contemporary">
<LINK REL=StyleSheet HREF="tables.css"
TITLE="Contemporary">
<LINK REL=StyleSheet HREF="forms.css"
TITLE="Contemporary">
```

In this example, three style sheets are combined into one "Contemporary" style that is applied as a preferred style sheet. To combine multiple style sheets into a single style, one must use the same **TITLE** with each style sheet.

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 for Windows 95/NT4 does not support **BODY background** images or colors from linked style sheets. Given this bug, authors may wish to provide another mechanism for including a background image or color, 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" MEDIA=screen>
<!--
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 required **TYPE** attribute is used to specify a media type, as is its function with the **LINK** element. Similarly, the **TITLE** and **MEDIA** attributes may also be specified with **STYLE**.

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 should be contained within an SGML comment (`<!-- comment -->`), as in the preceding example.

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* with CSS's **@import** statement. This statement may be used in a CSS file or inside the **STYLE** element:

```
<STYLE TYPE="text/css" MEDIA="screen, projection">
<!--
  @import url(http://www.htmlhelp.com/style.css);
  @import url(/stylesheets/punk.css);
  DT { background: yellow; color: black }
-->
</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 are 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. The three style sheets could also be [combined](#) via the [LINK](#) element.

Inlining Style

Style may be inlined using the **STYLE** attribute. The **STYLE** attribute may be applied to any [BODY](#) element (including **BODY** itself) except for [BASEFONT](#), [PARAM](#), and [SCRIPT](#). 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 style is far more inflexible than the other methods. To use inline style, one must declare a single style sheet language for the entire document using the **Content-Style-Type** HTTP header extension. With inlined CSS, an author must send **text/css** as the **Content-Style-Type** HTTP header or include the following tag in the [HEAD](#):

```
<META HTTP-EQUIV="Content-Style-Type" CONTENT="text/css">
```

Inlining style loses many of the advantages of style sheets by mixing content with presentation. As well, inlined styles implicitly apply to all media, since there is no mechanism for specifying the intended medium for an inlined style. This method should be used sparingly, such as when a style is to be applied on all media to a single occurrence of an element. If the style should be applied to

a single element instance but only with certain media, use the **ID** attribute instead of the **STYLE** attribute.

The CLASS Attribute

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

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

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 example's style sheet, 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 previous 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, or if the author wished to define an aural style for those using speech synthesizers.

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

```
#wdg97 { font-size: larger }
```

may be applied in HTML through the **ID** attribute:

```
<P ID=wdg97>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, digits, or hyphens. The letters are restricted to A-Z and a-z.

Note that [HTML 4.0 allows](#) periods in **ID** attribute values, but [CSS1 does not allow](#) periods in ID selectors. Also note that CSS1 allows the Unicode characters 161-255 as well as escaped Unicode characters as a numeric code, but HTML 4.0 does not allow these characters in an **ID** attribute value.

The use of **ID** is appropriate when a style only needs to be applied once in any document. **ID** contrasts with the [STYLE](#) attribute in that the former allows medium-specific styles and can also be applied to multiple documents (though only once in each 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 an [inline 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 4.0//EN">
<HTML>
<HEAD>
<TITLE>Example of SPAN</TITLE>
<META HTTP-EQUIV="Content-Style-Type"
CONTENT="text/css">
<STYLE TYPE="text/css" MEDIA="screen, print, projection">
<!--
  .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 main difference being 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. HTML 4.0 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>
```

Tag-Less Styles

Sometimes you may want to modify the style of a portion of a tag's contents rather than the entire contents of a tag. For example, you might want to change the size or appearance of a single sentence within a paragraph. This can be done using either the **** **** or **<DIV>** **</DIV>** tag pairs. These allow you to delimit blocks of text to which styles and classes can be applied. (Both tags also allow many other attributes which are beyond the scope of this class.) As a general rule, you might enclose small blocks of text, e.g. portions of a sentence or a few sentences within a long paragraph, with the **** tag and use the **<DIV>** tag to enclose large blocks of text, e.g. multiple paragraphs or whole sections of a long document.

To assign styles to either of these tags, you would treat them just as any other HTML tag. You can assign a single style to the tag or create a class that can be used with the tag. They can be inline styles, embedded styles, or linked styles. And both **** and **<DIV>** can be nested, if desired.

Creating a Style Sheet:

Style sheets can be created using any type of basic text editor. Notepad will work just fine. If you want something a little bit fancier, there are several options available to you.

Most popular wysiwyg html authoring programs (such as Dreamweaver for example) have built-in style sheet editors.

Or, you can order free standing style sheet editors

StyleMaster at <http://www.westciv.com/> (about \$50).

Topstyle Pro (<http://www.bradsoft.com/>) \$80

AceHTML (<http://www.visicommedia.com/acehtml/>) Free. Html Editor with built-in css and other features

Inheritance and Cascading

When the user agent wants to render a document, it needs to find values for style properties, e.g. the font family, font style, size, line height, text color and so on. The exact mechanism depends on the style sheet language, but the following description is generally applicable:

The cascading mechanism is used when a number of style rules all apply directly to an element. The mechanism allows the user agent to sort the rules by specificity, to determine which rule to apply. If no rule can be found, the next step depends on whether the style property can be inherited or not. Not all properties can be inherited. For these properties the style sheet language provides default values for use when there are no explicit rules for a particular element.

If the property can be inherited, the user agent examines the immediately enclosing element to see if a rule applies to that. This process continues until an applicable rule is found. This mechanism allows style sheets to be specified compactly. For instance, authors may specify the font family for all elements within the BODY by a single rule that applies to the BODY element.

Not all properties inherit, which can make for frustration when trying to determine why something doesn't seem to work right in your document. To help you debug your problems, here is a chart showing which properties inherit and which do not.

These Properties <i>Do</i>	These Properties <i>Don't</i>
-----------------------------------	--------------------------------------

Inherit	Inherit
<ul style="list-style-type: none"> • color • font • font-family • font-size • font-style • font-variant • font-weight • letter-spacing • line-height • list-style • list-style-image • list-style-position • list-style-type • text-align • text-indent • text-transform • white-space • word-spacing 	<ul style="list-style-type: none"> • background • background-attachment • background-color • background-image • background-position • background-repeat • border • border-color • border-left (-right, -top, -bottom) • border-left-width (-right-width, -top-width, -bottom-width) • border-style • border-width • clear • display • float • height • margin • margin-left (-right, -top, -bottom) • padding • padding-left (-right, -top, -bottom) • text-decoration • vertical-align • width

Referencing Style sheets:

```

<HEAD>
  <STYLE type="text/css">
    H1 {border-width: 1; border: solid; text-align: center;}
  </STYLE>
</HEAD>

```

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

```
<LINK REL=StyleSheet HREF="style.css" TYPE="text/css"
MEDIA=screen>
<LINK REL=StyleSheet HREF="color-8b.css" TYPE="text/css"
TITLE="8-bit Color Style" MEDIA="screen, print">
<LINK REL="Alternate StyleSheet" HREF="color-24b.css"
TYPE="text/css" TITLE="24-bit Color Style" MEDIA="screen,
print">
<LINK REL=StyleSheet HREF="aural.css" TYPE="text/css"
MEDIA=aural>
```

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. Configuring the server to send **text/css** as the **Content-type** for CSS files is also a good idea.

The **<LINK>** tag also takes an optional **MEDIA** attribute, which specifies the medium or media to which the style sheet should be applied. Possible values are

- **screen** (the default value), for presentation on non-paged computer screens;
- **print**, for output to a printer;
- **projection**, for projected presentations;
- **aural**, for speech synthesizers;
- **braille**, for presentation on braille tactile feedback devices;
- **tty**, for character cell displays (using a fixed-pitch font);
- **tv**, for televisions;
- **all**, for all output devices.

Multiple media are specified through a comma-separated list or the value **all**.

External style sheets should *not* contain any HTML tags like **<HEAD>** or **<STYLE>**. The style sheet should consist merely of style rules or statements. A file consisting solely of

```
P { margin: 2em }
```

could be used as an external style sheet.

References:

<http://www.w3c.org>

<http://www.w3schools.com>