

# Web Authoring: Cascading Style Sheets

Academic Computing Services  
A Division of Information Services

[www.ku.edu/acs](http://www.ku.edu/acs)

---

**Abstract:** This document is used in conjunction with ACS workshops on Cascading Style Sheets. It covers the basics of using style sheets, including basic selectors and properties, and style sheet linking and embedding.

---

## Contents

Introduction .....	3
Objectives .....	3
Prerequisites .....	3
Related Training Available from ACS.....	3
Introduction .....	5
The Basics .....	5
What you need .....	5
Creating a HTML 4.0 compliant page .....	6
The Cascade.....	7
Precedence .....	7
Inheritance .....	7
Basic CSS Syntax.....	8
Case and punctuation .....	8
Selectors .....	9

## ACS Computer Training

### Web Authoring: Cascading Style Sheets

---

Properties.....	9
Values .....	9
Properties.....	10
Text-level properties.....	10
Block-level properties.....	13
Additional CSS Syntax .....	15
Grouping styles .....	16
Context-specific styles .....	16
Comments in styles.....	16
Going back to HTML .....	16
The <code>class</code> attribute .....	17
The <code>&lt;DIV&gt;</code> element .....	17
Using CSS with HTML .....	18
Creating a style sheet as a separate document.....	18
Embedding CSS within HTML documents .....	18
Using them all together .....	19
Using the “media” attribute to specify specialty styles .....	20
For More Information.....	20
Getting Additional Help .....	21

## Introduction

Cascading Style Sheets, or CSS, are analogous to “styles” that can be set within documents created by desktop publishing programs. In such programs, styles are assigned to selected characters or text, and the assigned style contains formatting such as font, typeface, style and position information. Editing the style affects all the text assigned that style, so that the designer doesn’t have to go through the document and edit the layout of individual items. CSS works exactly the same way. It is a new set of rules and codes, separate from HTML, that can be linked with and embedded within an HTML document, and used to control the layout of elements of the document.

## Objectives

The goal of this workshop is to introduce participants to the basic structure and implementation of Cascading Style Sheets, for use with HTML documents. Topics to be covered and goals include:

- Understand the importance of using style sheets in conjunction with the HTML 4.0 specifications, in order to separate content from display
- Learn the basic syntax of style sheets
- Create Web pages suitable for style sheets
- Use style sheets to manipulate text-level element properties such as font, size, and color.
- Use style sheets to manipulate block-level element properties such as margins and borders
- Understand how to embed and link style sheets into Web pages

## Prerequisites

It is assumed that participants in this workshop have taken the *Web Authoring: Introduction*, *Web Authoring: Intermediate* and *Publish your Web Page on the Internet* workshops. Alternatively, the participants should be comfortable creating Web pages using a simple text editor, they should have a solid knowledge of the basic HTML vocabulary, and they should be able to create pages that include images and both relative and absolute hyperlinks. It is also assumed participants are familiar with publishing Web pages on a UNIX server. If you do not have these skills, you will be asked to look on with another class participant who has the prerequisites.

## Related Training Available from ACS

All workshops offered by Academic Computing Services (ACS), a division of Information Services, are free to KU students, staff, faculty, and [approved affiliates](#). The general public is also welcome to most workshops, but some ACS workshops require a [registration fee](#) for them.

To learn more about or register for workshops, receive automatic announcements of upcoming workshops, and track workshops you've registered for and have attended, visit the ACS Web site at [www.ku.edu/acs/train](http://www.ku.edu/acs/train). You can also check our online schedule at [www.ku.edu/acs/schedule](http://www.ku.edu/acs/schedule) for a list of class offerings and their availability. For further workshop related questions, please email [training@ku.edu](mailto:training@ku.edu).

The following workshops are offered in the area of web authoring:

**Web Authoring: Introduction**

Learn to create and organize simple documents for the World Wide Web using Hypertext Markup Language (HTML). Hypertext links and basic formatting elements are discussed. Prerequisites: Web Browsing or equivalent skills. Participants must also be very comfortable with word processing.

**Web Authoring: Intermediate**

This course covers advanced techniques for creating hypertext links, working with placement of images (graphics), using special characters such as ©, adding backgrounds, and using other HTML niceties in your Web pages. Prerequisite: Web Authoring: Introduction (or equivalent skills).

**Web Authoring: Tables, Frames, and Imagemaps**

Enhance your Web page layout through the use of tables; divide your Web page into multiple, scrollable regions using frames; and produce image maps that allow individual portions of the same image (graphic) to serve as hypertext links. Prerequisite: Web Authoring: Intermediate or equivalent skills.

**Web Authoring: Forms and CGI scripts**

Learn to program dynamic, interactive Web sites with this introduction to Perl programming and CGI scripting, including coverage of HTML forms. Prerequisites: Web Authoring: Publish Your Web Page on the Internet, Web Authoring: Intermediate, and UNIX: Introduction.

**Web-database integration**

Learn basic database fundamentals, covering database design; SQL; table creation; and inserting, updating, and selecting table data. Create a Web-based interface to a database with an HTML form and CGI scripting, and see how to combine a database and CGI script to produce dynamic Web content. Work through the hands-on examples using mSQL and Perl in the UNIX environment of ACS's multiuser systems. The wide range of other available tools is also discussed. Prerequisite: Web Authoring: Forms and CGI scripts or equivalent skills.

## Introduction

Cascading Style Sheets, or CSS, are analogous to “styles” that can be set within documents created by desktop publishing programs. In such programs, styles are assigned to selected characters or text, and the assigned style contains formatting such as font, typeface, style and position information. Editing the style affects all the text assigned that style, so that the designer doesn’t have to go through the document and edit the layout of individual items. CSS works exactly the same way. It is a new set of rules and codes, separate from HTML, that can be linked with and embedded within an HTML document, and used to control the layout of elements of the document.

CSS are an important part of the revolution in the way Web pages are designed and implemented. This is a revolution that is still underway, however, so as this workshop teaches you the basics of using CSS in your own Web pages, you may find that even the latest browser versions do not treat the style elements you create in a consistent manner. As with any Web-related creation, it is in your best interest to check your Web pages on as many browsers as possible. CSS are not supported by browsers prior to *Internet Explorer* version 3.x or *Netscape Navigator* version 4.x., and any CSS information you include in your Web pages will be ignored by such browsers.

This handout is not intended to be a comprehensive list of all the style sheet statements you can add to a Web page. In fact, it only introduces the basics. In creating this handout, we hope that you can use this in combination with other tutorials that have already been published on the Web, as well as the actual specification describing the complete CSS vocabulary, to learn as much CSS as you want or need.

## The Basics

The first thing to understand about CSS is that CSS *are not HTML*. Although CSS are way of telling browsers how to display Web pages, the code that makes up CSS is separate and different from HTML. CSS can either be embedded inside a HTML document inside a special tag or attribute, or a CSS style sheet can be stored in a separate file and linked to the HTML document.

Another important aspect of understanding CSS is appreciating the goals of the World Wide Web Consortium (W3C) in developing its latest specifications. That goal was to separate the content of a Web page from its layout. Text marked up with HTML is the content part of that equation, and CSS is the layout part. These specifications are published at the W3C Web site:

[www.w3.org/TR/REC-CSS1](http://www.w3.org/TR/REC-CSS1) and [www.w3.org/TR/REC-CSS2](http://www.w3.org/TR/REC-CSS2)

## *What you need*

### HTML 4.0

The first thing you need to create a Web page with CSS is an HTML-coded page with few or no *deprecated* HTML tags and attributes, as defined by the HTML 4.0 specification. Deprecated tags are tags that are being phased out of the HTML syntax. They are still part of the official specification in order that pages coded in earlier HTML versions continue to display correctly, but authors of such pages and browser developers

can expect that subsequent HTML versions will not include these tags. As a group, the deprecated tags and attributes in HTML 4.0 include all those tags which dictate the way text is displayed.

## **Style**

Once you have your unadorned, essentially unformatted Web page, you will need to know how you want it to look: how you want the headings to display, where you want your images, how you want to arrange your tables, etc.

Sometimes, the easiest way to figure out what you want is to experiment with what you can do. Look at the list of properties in this handout, check out the tutorials on the Web (see the section on “Other Resources”), and see how these things work. But the important thing to remember about style sheets is that they represent an overall layout and design for your Web page, instead of the individual formatting of tags and elements as we have done in the past. Instead of thinking in terms of a particular element or section in your page, think of all headings, all lists, all images, etc.

Style sheets, by their very nature, will force you to design your page, rather than just mark it up.

## ***Creating a HTML 4.0 compliant page***

How do you find out what tags are deprecated in HTML 4.0 and which are acceptable? If you are already familiar with HTML, the best resource is the HTML 4.0 specification published by the World Wide Web Consortium (or W3C), available in HTML, .pdf and other formats at:

[www.w3.org/TR/REC-html40](http://www.w3.org/TR/REC-html40)

If you are new to HTML, you might want to try either the other handouts available from Academic Computing Services, or the tutorial sites both listed at the end of this handout.

In addition, if you want to check your Web site for compliance with the specification, the W3C also provides a validation service, which allows you to check a published site (i.e., a site already located on a Web server computer) for correct use of HTML tags and attributes, in accordance with the official document type definition, or DTD:

[validator.w3.org](http://validator.w3.org)

## **Adding the <STYLE> tag**

Style sheets are associated with Web pages via several methods. A basic way to embed CSS in your page that clearly shows the syntax of style sheets is to use the <STYLE> tag. In addition, as you are starting out with CSS, this method allows you to work within a single document, rather than having to link separate documents.

The <STYLE> Placed inside the <HEAD> of your HTML document, and contains your CSS declarations. The <STYLE> tag requires at least one attribute: type=“text/css.” The value for the attribute tells the browser that the style sheet is for a text medium (as opposed to music or video) and that the language is CSS.

HTML comment markers `<!-- comment -->` should be placed inside the `<STYLE>` tags, around the content of your style sheet, to hide your style declarations from older browsers that don't recognize styles.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
  <TITLE>A page with Style</TITLE>
  <STYLE type="text/css"> <!--
    your style information goes in here
  --></STYLE>
</HEAD>
<BODY>. . .
```

*Example use of the `<STYLE>` tag*

## The Cascade

Why is it called “cascading?” The official definition has to do with the way style sheets affect Web pages. Rather than having to limit themselves to a single style sheet, Web authors can create “layers” of style sheets, through which a page’s content is filtered.

1. With CSS-compliant browsers, this layering effect is already at work. Any Web page you view with such a browser is seen through at least three style sheets:
2. any styles embedded in the page by the Web author,
3. the Web author’s style sheet(s) that loaded with the page,
4. the user-defined style sheet, which you, the user, could create, and
5. the default style sheet that the browser uses to display pages.

## Precedence

The CSS specifications emphasize the cascade by defining the order in which these style sheets will have precedence over any page. The list above represents the precedence order. The author’s style sheets have the highest precedence and will over-ride any other styles that may be set up by the user or the browser. Next, the user has authority over the display of elements, and finally the browser gets a chance to say how a page will display, if the previous two left any element untouched.

It’s important to understand how these style sheets really do work like a filtering system, directing the display of the web page on an element-by element basis. If, for example, the Web author creates a style sheet but does not define the way a particular element is displayed, then the user’s styles or the browser’s style will dictate how the element displays.

## Inheritance

Another essential aspect to style sheets is the concept of inheritance. This is another part of the concept of the cascade. In essence, inheritance says that HTML elements that are embedded inside of other elements will inherit the styles of the outer element.

For example, in the following code:

```
. . .  
<BODY>  
<P><EM>This is an emphasized paragraph. There is a  
<SMALL>small</SMALL> word inside this paragraph.</EM></P>  
</BODY>  
. . .
```

the `<SMALL>` tag is inside an `<EM>` tag which is inside a `<P>` tag, all of which are inside the `<BODY>` tag.

Any styles that are created for the body are inherited by all the tags inside. Likewise, the `<EM>` tag inherits any styles set for the `<P>` and `<BODY>` tags, and the `<SMALL>` tag inherits the `<EM>`, `<P>` and `<BODY>` styles.

You can see this at work even if no styles are explicitly set for the page, because the default browser styles will affect the display. The text style for the body will be consistent throughout the page, and the `<SMALL>` element will inherit the `<EM>` default display style, which usually sets the text to be italic, thus making the word appear both small and italic.

## Basic CSS Syntax

The following represents the basic CSS syntax:

```
SELECTOR { property1: value1;  
           property2: value2;  
           property3: value3; ... }
```

where

**SELECTOR** is the HTML element for which you are defining a style (e.g., the `BODY`).

**PROPERTY** is the aspect of that HTML element you are defining (e.g., an element's color, its margins, or its font).

**VALUE** is the specific way you want that aspect to display or behave (e.g., green or 20 pixels tall).

## Case and punctuation

CSS is not case sensitive and typing each property definition on a separate line is not required, but for readability, this handout uses these capitalization and spacing conventions.

The punctuation is a requirement of CSS. Following the selector, properties are listed between “curly” brackets, and each property name is separated from its assigned value by a colon. If there are multiple properties defined in a style, a semi-colon separates each pair (property and value) from the next pair. For the very last property and value pair, the semi-colon is not required, but you may want to include it anyway, should you choose to add other pairs later.

**Note:** If you make a mistake with punctuation in your style sheet, what generally happens is that all statements following the error are ignored, so care with your punctuation is very important.

---

## Selectors

Selectors in CSS are simply the HTML tags that exist in your web page. Remember, however, that CSS is closely tied to HTML 4.0, so the HTML tags and elements that are affected by your style sheet should be HTML 4.0 tags, rather than deprecated or proprietary tags.

For example, <BLINK> is a proprietary tag that was added to HTML not by the W3C but rather by the developers at Netscape, who created browsers that recognized and supported the tag. The tag <BLINK> is not an official tag in either HTML 3.2 or HTML 4.0. Creating a CSS style with this tag as the selector, even if the browser makes the contained text blink, the style should have no affect on the display of the page.

## Properties

Properties are aspects of HTML tags and elements that you can adjust with your style sheet. They are somewhat analogous to the attributes that can be set for some tags in HTML. However, unlike the limited list of attributes that you can assign to some tags, the list of properties that you can determine with your style sheet is widely varied.

## Values

Values can be arranged into two general groups: keywords and numeric values. Most properties take either kind of value, though some take only keywords.

An excellent example is the color property. Color can take for its value any of the 16 color keywords (see list below) or any of the RGB hexadecimal values, preceded by the # sign. Thus, color: navy is the same as color: #000080.

Keyword	RGB hex #	Keyword	RGB hex #
black	#000000	green	#008000
silver	#C0C0C0	lime	#00FF00
gray	#808080	olive	#808000
white	#FFFFFF	yellow	#FFFF00
maroon	#800000	navy	#000080
red	#FF0000	blue	#0000FF
purple	#800080	teal	#008080

fuchsia	#FF00FF	aqua	#00FFFF
---------	---------	------	---------

Unlike HTML attribute values, CSS property values, with a few rare exceptions, are not placed in quotes.

Following is a complete CSS example using a <STYLE> tag:

```
<STYLE type="text/css">
<!--
    H1    {color: navy;
          background-color: #FFFFFF;}
    P     {color: #800000;}
-->
</STYLE>
```

In this example, the <H1> element text will appear in navy with a white background and paragraphs will have maroon text.

## Properties

CSS properties can be broken down into two major groups: those properties which affect text and text display, and the properties which affect larger elements. These groups are referred to as text-level and block-level properties, respectively.

You can think of text-level properties as display effects that might influence a single item separately in any line of text, as compared with block-level elements, which affect entire blocks of text or elements on a page.

Examples of text-level properties would be things like the color of the text, the typeface of the text, its size, and so forth. Block-level elements, by contrast, would include things like borders and margins.

This distinction is important for anticipating which properties can be applied to which HTML tags:

- in-line elements and tags have text-level properties. E.g., EM, STRONG, DFN, CODE, KBD, CITE, ABBR
- block elements have block-level properties, as well as text-level properties. E.g., BODY, H1, H2, etc., OL, UL, DL, P, LI, TABLE

## *Text-level properties*

At text-level, every element in HTML can have both color (the text color) and a background color. In addition, these properties can be applied to block-level elements, as mentioned above.

## Colors

### *color*

This is the general property for applying color to an element. It can take either of the following values:

- the 16 color keywords
- any RGB hexadecimal color value, preceded by the # sign (See <http://www.habanero.com/hex> for a list of hexadecimal color values.)

Example:

```
BODY { color: #663300; }
```

### *background-color*

This is the property for applying a background color to an element. It can take either of the following values:

- the 16 color keywords
- any RGB hexadecimal color value, preceded by the # sign (See <http://www.habanero.com/hex> for a list of hexadecimal color values.)

Example:

```
BODY { background-color: #663300; }
```

## **Fonts**

Unlike the deprecated <FONT> tag of HTML, style sheets provide a much wider range of options for display, alignment and sizing of text on the page.

### *font-family*

You can create a preference list of fonts for the browser, since not every computer will have the same fonts installed. In addition, you can give the browser a “generic” font-family, in the case that none of your favorite fonts are available. Your preference list should be separated by commas, in descending order of preference. List the generic font-family last. This property takes font names as its values. Multi-word font names should be enclosed in quotes. In addition, it can take “generic” names of fonts, e.g. serif, sans-serif, monospace.

Example:

```
P { font-family: tahoma, "antique olive", sans-serif; }
```

### *font-size*

You can assign either absolute values or relative values to font-size. If you use a relative value, it will be based on the base font, i.e., the font size inherited by the element (in some cases, this may be the browser default.) The size values you can assign to this property include the following units:

- **px**: pixels (a relative measurement based on the monitor display)
- **em**: the m-height of the base font (a relative measurement)
- **ex**: the x-height of the base font (a relative measurement)
- **pt**: points (an absolute measurement equal to 1/72 of an inch)

- **pc**: picas (1 pica is equal to 12 points: 6pc = 1 inch)
- **in**: inches
- **cm**: centimeters
- **mm**: millimeters

Not all of these units are supported by all browsers. In addition, you can set the size of the font relative to the base font using % or the keywords larger and smaller.

Examples:

```
EM { font-size: 36px; }
BODY { font-size: 14pt; }
P { font-size: 1.5em; }
H2 { font-size: 200%; }
```

---

**Note:** These size values can be applied to other properties that have size, such as margins, borders, etc.

---

### *font-style*

This property sets a font to italic, normal or oblique (slanted to the left). It can take the values of normal, italic, or oblique

Example:

```
H1 { font-style: italic; }
P { font-style: oblique; }
```

### *font-weight*

This property can set a range of values that determine how bold, relative to the base font, the text displays. It takes a range of values, 100 to 900. These values give the range of boldness, where 400 is the normal base font, and 700 is the equivalent of “bold.” In addition, the following keywords are acceptable: normal, (the same as 400), bold (the same as 700), bolder, and lighter.

Example:

```
H3 { font-weight: 800; }
P { font-weight: lighter; }
```

### *text-decoration*

This property specifies the decoration, such as underlining, of an element. It can take the following values: none, blink, underline, line-through, and overline.

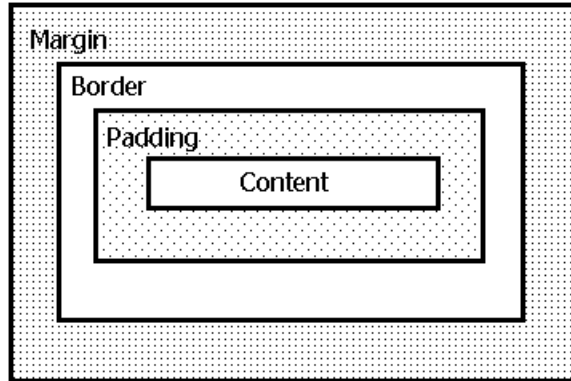
Example:

```
A { text-decoration: none; }
```

## ***Block-level properties***

The block-level properties are based on a “box model,” which describes a series of nested boxes that surround a block of content (which may be either text or images).

Content is the smallest, inner-most of these boxes. It is surrounded next by **padding**, which separates the content from the next box, the **border**. The last, largest box is the **margin**. A simplified graphical representation looks like this:



**Padding**-, **border**- and **margin**-related properties all have similar syntax and ranges of values. This handout describes only **border**- and **margin**-related properties.

## **Text alignment**

### ***text-align***

The text-align property affects blocks of content within the box model. It can take the values *left*, *right*, *center*, and *justify*.

Example:

```
H1 { text-align: center; }
```

## **Margins**

### ***margin-top, margin-right, margin-bottom, margin-left***

These four properties each describe a side of the **margin** box, allowing you to explicitly set the width of the margin surrounding the box on that particular side.

The CSS specification describes the following values that can be used to set the width of the **margin**, **border** or **padding**.

- **px**: pixels (a relative measurement based on the monitor display)
- **em**: the m-height of the base font (a relative measurement)
- **ex**: the x-height of the base font (a relative measurement)
- **pt**: points (an absolute measurement equal to 1/72 of an inch)

- **pc**: picas (1 pica is equal to 12 points: 6pc = 1 inch)
- **in**: inches
- **cm**: centimeters
- **mm**: millimeters

As mentioned earlier, not all of these units are supported by all browsers.

Example:

```
BODY { margin-left: 15px;
        margin-top: 10pt; }
```

## *margin*

While the properties listed above take just a single value, **margin** can take up to four values, allowing you to set the margins around a block of text in a single declaration. The values are separated by spaces. The values are specified as follows:

- 1 value: This sets all four margins to that one value.
- 2 values: The 1st value is assigned to the top and bottom margins. The 2nd value is assigned to the left and right margins.
- 3 values: The 1st value sets the top margin. The 2nd value sets the right and left margins. The 3rd value sets the bottom margin.
- 4 values: These are assigned to the margin sides in the following order: top, right, bottom, left (in clockwise order, starting from the top).

Example:

```
UL { margin: 2em 2em 2em 4em; }
BODY { margin: 5px 3em 10pt; }
```

## **Borders**

The syntax for borders is very similar to margins, except that borders have two additional sets of properties, in addition to their width. These properties set the color of the border, and the style in which the border is displayed.

### *border-width*

Just as with margins and font-size, **border-width** takes the same wide range of units. It can also take the keyword values *thin*, *medium*, *thick*.

Examples:

```
P { border-width: thick; }
H2 { border-width: 10px; }
```

### *border-style*

Different graphic styles can be assigned to borders using the **border-style** keywords *solid*, *double*, *groove*, *ridge*, *inset*.

Example:

```
P { border-style: inset; }
```

### *border-color*

This is the property for applying a color to border. It can take either of the following values:

- the 16 color keywords
- any RGB hexadecimal color value, preceded by the # sign (See <http://www.habanero.com/hex> for a list of hexadecimal color values.)

Example:

```
H3 { border-color: #663300; }
```

According to the specifications, each of these property sets — **border-width**, **border-style** and **border-color** — can be assigned separately to the four sides of a block, by inserting the name of the side into the property name like this:

```
H1 { border-left-width: 10px; }  
BLOCKQUOTE { border-bottom-color: green; }
```

However, *Internet Explorer* does not support this aspect of the specification.

### *border, border-top, border-right, border-bottom, border-left*

These five properties all use the same syntax. They can take one, two or three space-separated values, to set the three aspects of border properties. The order of values is **width, style, color**.

If any of the values are omitted, the following defaults will be assigned:

```
border-width: medium  
border-style: solid  
border-color: black
```

Examples:

```
H1 { border-right: 10px solid #008000; }  
OL { border: inset; }
```

---

**Note:** Netscape *Navigator* does not support the **border-top**, **border-right**, etc. properties and *Internet Explorer* only supports the `solid` style with single borders.

---

## Additional CSS Syntax

Within any style sheet, you can list any selector as many times as necessary to create the style you want, and you can list as many property/values pairs as you want for each selector. Keep this in mind as you are creating your style sheets, because even if you can sometimes create the same style declarations in a very minimal number of statements,

you may find it easier to organize your styles by using more declaration statements. Conversely, the longer your style sheets, the larger your file be, and the longer, therefore, it will take to load.

The CSS syntax supports a number of conventions that allow you to group, organize and specify styles for ease of coding and for fine-tuning your page display.

## **Grouping styles**

Rather than repeating style declarations for a group of elements that you want all to display similarly, you can group your style declarations in a single statement. In your style declaration, list the selectors separated by commas.

For example, if you want all your H1, H2 and H3 tags to display with a particular text color, the statement might look like this:

```
H1, H2, H3 { color: black; }
```

When grouping, selectors can be listed in any order.

## **Context-specific styles**

Sometimes you will find that a style declaration for a particular element looks great in general, but when that element appears embedded in another tag the style just doesn't work. In that case, you'd want to create a specific style for that element in that context.

Such styles are called *context-specific styles*, and to create them, list the selectors in the order that they are embedded in the web page.

For example, to set a specific style for anchor (A) tags that appear in headings (e.g., H1), use the following syntax:

```
H1 A { background-color: white; }
```

This syntax can be used for as many levels of embedding as you may have in your Web page, but the order of the selectors must reflect the embedding order, where the last selector listed is the inner most element.

## **Comments in styles**

Just as in HTML, you can add comments to your CSS statements to explain or remind yourself what your codes are doing. CSS comments use different syntax than HTML. Comments follow this syntax:

```
/* this is a CSS comment */
```

CSS comments can appear on a single line or on multiple lines. This is a multi-line comment:

```
/* this is a CSS comment that wraps  
around two lines */
```

## **Going back to HTML**

Some of the ways you can fine-tune your style sheets require you to go back to your HTML code, in order to give you some additional tools for your style sheets.

## The *class* attribute

If you want to create a set of styles that affect a select group of elements in your Web pages, or if you want to identify specific instances of an element in your page, using the class attribute in your HTML gives you this power.

An example of HTML content using the class attribute might look like this:

```
. . .  
<BODY>  
    <H1>Heading One</H1>  
    <P>A regular paragraph.</P>  
    <H1 class="special">A special heading</H1>  
    <P class="special">A special paragraph.</P>  
. . .
```

In this example, the value “special” is a name you create to identify this class: it can be any word or string of characters without spaces. In the style sheet, you can reference these classes in two different ways:

- To refer to specific element types of a class, use this syntax:

```
H1.special { color: green; }
```

- To refer to all elements of a particular class, use the following syntax:

```
.special { font-style: italic; }
```

## The *<DIV>* element

DIV stands for “DIVIDE” and is used to section off parts of a document or group parts of a document together. For example, if you want to apply a certain formatting to a paragraph and a table or a list and a paragraph, you could use DIV to treat both elements as one group. For example,

```
<DIV>  
    <P>The following table illustrates the various  
    percentages of ...</P>  
    <TABLE>  
    <TR> <TD>50%</TD>  
        <TD>76%</TD> </TR>  
    </TABLE>  
</DIV>
```

*<DIV>* is a block-level element and, in HTML 4.0, has no function until given a stylesheet attribute such as `class="special"`. See the section on the `class` attribute above.

## Using CSS with HTML

There are four distinct ways of combining CSS with HTML. These can be mixed and matched to provide you with exact control over the various elements of the pages in your web site. Two of the methods are for linking external documents of style sheet declarations with your web pages, and two are for embedding styles directly within a HTML document.

### *Creating a style sheet as a separate document*

If you are creating a style sheet as a separate document, to be linked to your Web site, use the following steps:

1. Create a file of style sheet declarations for your Web site.
2. Save the file as plain text, and name the file with a .css file extension (e.g., my-style.css).
3. Publish your style sheet on a Web server, just as you would any other file or page on your site, and set the proper access permissions, if required by your ISP.
4. Use @import or <LINK> (see below) to link the style sheet to your web page.

### @import

There are two ways of linking a HTML web page with a style sheet saved as a separate external file. This is the first. Inside the <STYLE> element, you can place the following statement syntax to reference an external style sheet:

```
@import url(SHEET_URL);
```

where the SHEET\_URL is the location of your published style sheet.

---

**Note:** This method does not work with Netscape Navigator.

---

### <LINK>

You can use the <LINK> element in the HEAD of your web page to link an external style sheet with the Web page. Unlike the @import declaration, this method is supported by both Internet Explorer and Navigator.

The syntax looks like this:

```
<LINK rel="stylesheet" type="text/css" href="URL">
```

where the URL is the location of your published style sheet.

---

**Note:** <LINK> is not a container and does not take an end tag.

---

### *Embedding CSS within HTML documents*

There are two ways of embedding CSS style declarations within an HTML document:

## The <STYLE> tag

This is the case described in the example at the beginning of this handout. Using the <STYLE> element embeds your style sheet into your HTML document.

## Inline styles

CSS declarations can be embedded inside elements in an HTML document using the style attribute. These are called inline styles.

To create an inline style, add the style attribute to an element. The attribute's value is the style declaration pairs of property and value, as would have appeared inside of "curly" brackets in a style sheet (without the "curly" brackets). For example:

```
<H1 style="text-align: center">Heading One</H1>
<A style="text-decoration: none; color: red">Click
  here.</A>
```

## Using them all together

You can use these different methods together to make your page display exactly the way you want. A complete example, using all four methods might look like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN"
  "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
  <TITLE>title</TITLE>

  <LINK rel="stylesheet" type="text/css"
    href="http://eagle.cc.ukans.edu/~imajhawk/style1.css"
    title="my_sheet">

  <STYLE type="text/css"><!--
    @import
      url(http://eagle.cc.ukans.edu/~imajhawk/style2.css);

    H1 { color: blue; size: larger; }
  --></STYLE>
</HEAD>
<BODY>
  <H1>The Headline is blue</H1>
  <P style="color: green">While the paragraph is
    green.</P>
</BODY>
</HTML>
```

## Using the “media” attribute to specify specialty styles

The “media” attribute can be used in conjunction with the <LINK> element, the <STYLE> element, or independently as an @media declaration. It allows you to specify what delivery medium will be affected by a particular style or stylesheet.

Here’s an example of the media attribute used in conjunction with the <LINK> element:

```
<LINK rel="stylesheet" type="text/css" media="screen"
href="mystyle.css">

<LINK rel="stylesheet" type="text/css" media="print"
href="printstyle.css">
```

In the preceding example, two stylesheets are applied to the page: the first one (mystyle.css) affects only the way the page is presented onscreen; the second one (printstyle.css) affects only the way the page appears when printed.

Available values for the media attribute include:

- **all**: all devices (default).
- **aural**: text-to-speech browsers that “speak” the page content.
- **braille**: braille tactile feedback devices.
- **embossed**: paged braille printers.
- **handheld**: handheld computers, or PDAs (typically small screen, monochrome, limited bandwidth).
- **print**: paged printouts and for documents viewed on screen in print preview mode.
- **projection**: viewed through a projector or printed on transparencies.
- **screen**: color computer screens.
- **tty**: media using a fixed-pitch character grid, such as teletypes, terminals (such as “greenscreen” terminals), or portable devices with limited display capabilities. Authors should not use pixel units with the "tty" media type.
- **tv**: television-type devices, such as WebTV (low resolution, color, limited-scrollability screens, sound available).

## For More Information

The following Web sites have excellent information, tutorials and reference information for using CSS in Web pages:

- The W3C specifications defining CSS:  
[www.w3.org/TR/REC-CSS2](http://www.w3.org/TR/REC-CSS2)
- CSS tutorials and references on the web:  
[www.hotwired.com/webmonkey/stylesheets](http://www.hotwired.com/webmonkey/stylesheets)  
[www.projectcool.com/developer/dynamic](http://www.projectcool.com/developer/dynamic)

[www.wdvl.com](http://www.wdvl.com)

[www.htmlgoodies.com](http://www.htmlgoodies.com)

- What CSS properties are supported by different browsers:

[www.projectcool.com/developer/reference/css\\_style.html](http://www.projectcool.com/developer/reference/css_style.html)

- HTML tutorials:

[www.cc.ukans.edu/cgiwrap/acs/subject.pl](http://www.cc.ukans.edu/cgiwrap/acs/subject.pl)

[www.htmlgoodies.com](http://www.htmlgoodies.com)

[www.projectcool.com](http://www.projectcool.com)

[www.hotwired.com/webmonkey](http://www.hotwired.com/webmonkey)

- RGB hexadecimal colors for use on the web:

[www.habanero.com/hex](http://www.habanero.com/hex)

[junior.apk.net/~jbarta/weblinks/color\\_picker/](http://junior.apk.net/~jbarta/weblinks/color_picker/)

[www.stone.com/java/cc/ColorCoordinator.html](http://www.stone.com/java/cc/ColorCoordinator.html)

[www.lynda.com/hexh.html](http://www.lynda.com/hexh.html)

[members.xoom.com/serbach/introhtm/colors02.htm](http://members.xoom.com/serbach/introhtm/colors02.htm)

[www.phoenix.net/~jacobson/pages/rgbhex.html](http://www.phoenix.net/~jacobson/pages/rgbhex.html)

## Getting Additional Help

ACS provides consulting and Q&A help in a variety of ways:

785/864-0200

[question@ku.edu](mailto:question@ku.edu)

[www.ku.edu/acs/help](http://www.ku.edu/acs/help)

*Last Update: 03/18/2003*