Adding Styles to a Document

Style rules can be applied to documents in three ways: as inline style directions, as style elements embedded at the top of the document itself, and as external files that can be either linked to or imported into the document.

When attaching styles to a document, it is important to keep in mind that other style sheets may apply to your document as well. User agents, such as browsers, have built-in style sheets for rendering content. In addition, individual users may create their own style sheets and apply them to a single site or to all the sites they visit in order to make the text comfortable to read or to meet special needs. Which style sheet takes precedence is covered in the upcoming "Document Structure and Inheritance" section.

Inline Styles

You can add style information to an individual element by using the `style` attribute within the HTML tag for that element. The value of the `style` attribute is one or more standard style declarations, as shown here:

```
<h1 style="color: red">This Heading will be Red</h1>

<p style="font-size: 12px; font-family: 'Trebuchet MS', sans-serif">
This is the content of the paragraph to be set with the
described styles.</p>
```

Note that if the `style` attribute uses double quotation marks as shown, quoted values within the list (such as the font name "Trebuchet MS" in the example) must use single quotation marks. The reverse is also valid: if the document uses single quotes for attributes, then contained quoted values require double quotes.

Although a perfectly valid use of style information, inline styles are equivalent to the `font` extension to HTML in that they pollute the document with presentation information. With inline styles, presentation information is still tied to individual content elements, so any changes must be made in each individual tag in every file. Inline styles are best used only occasionally to override higher-level rules. In fact, the `style` attribute has been deprecated in XHTML 1.1 and does not appear in other XML languages.

Embedded Style Sheets

A more compact method for adding style sheets is to embed a style block in the top of the HTML document using the `style` element summarized here.

style

```
<style> ... </style>
```

Attributes

Internationalization (lang, dir, xml:lang)
media="all|aural|braille|handheld|print|projection|screen|tty|tv"
title="*text*"
type="*content type*" (*Required*)

The following example shows these sample rules embedded in an XHTML document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<style type="text/css">
   h1 {color: #666;}
   p  {font-size: 90%;
       font-family: Verdana, sans-serif; }
</style>
<title>Style Sheets</title>
</head>
...
</html>
```

The style element must be placed within the head tags in the document. Currently, Cascading Style Sheets is the only widely supported style sheet language, but the W3C has prepared for the possibility of additional languages to be added in the future by providing the type attribute within the style element. The only viable style type as of this writing is text/css. The type attribute is required in both HTML and XHTML; if it is omitted, some browsers may ignore the entire style sheet.

In addition, the media attribute in the style element (not shown in the example) may be used to target the medium (screen, print, handheld, etc.) to which the style sheet should be applied. If it is not present, the default is "all" media. The media attribute is discussed in the "CSS for Other Media" section.

Browsers that do not support style sheets (such as Version 2 browsers) will not recognize the style element and may display the style rules on the page. If for some reason you need to support non-CSS browsers, you can prevent the contents from displaying by placing them within comments, as shown in this example:

```
<style type="text/css">
<!--
   h1 {color: #36C;}
-->
</style>
```

Although once standard markup, the inclusion of comments in the `style` element is no longer conventional as older browsers disappear from use.

External Style Sheets

The most powerful way to use CSS is to collect all the style rules in a separate text document and create links to that document from all the pages in a site. In this way, you can make stylistic changes consistently across a whole site by editing the style information in a single document. This is a powerful tool for large-scale sites (and small ones, too, for that matter).

Style sheet content

The style sheet document is a plain-text document that contains at least one style sheet rule. It may *not* contain HTML tags (after all, it isn't an HTML document) and so including HTML tags may cause parts of the style sheet to be ignored. HTML comments are also not permitted, however, comments may be inserted in the style sheet by using the CSS comment syntax shown here:

```
/* This is the end of the chapter */
```

There are two ways to refer to external style sheets (which should be named with the *.css* suffix) from within a document: the `link` element and the `@import` directive.

Using link

The best-supported method for referring to external style sheets is to create a link to the CSS document using the `link` element in the `head` of the document, as shown in this example:

```
<head>
<link rel="stylesheet" href="/pathname/stylesheet.css" type="text/css" />
</head>
```

The `rel` attribute defines the linked document's relation to the current document—a "style sheet." The `href` attribute provides the URL of the style sheet document. Authors may link to more than one style sheet in a document and both will apply.

Importing

An alternative to linking is to import an external style sheet into a document using the `@import` function in the `style` element:

```
<style type="text/css">
<!--
   @import url(http://pathname/stylesheet.css);
   p {font-face: Verdana;}
-->
</style>
```

In this example, an absolute URL is provided, but a relative URL may also be used. `@import` commands must come *before* anything else (except `@charset`).

Alternate Style Sheets

CSS 2 introduced the ability to specify alternate style sheets by setting the value of the `rel` attribute to `alternate stylesheet`, as shown in the following example.

```
<link rel="stylesheet" type="text/css"
    href="/pathname/basic.css" title="Basic Style" />
<link rel="alternate stylesheet" type="text/css"
    href="/pathname/largetype.css" title="Larger type" />
<link rel="alternate stylesheet" type="text/css"
    href="/pathname/minimal.css" title="Minimal Design" />
```

When the document in the above example loads into the browser, the *basic.css* style sheet is applied by default because it is the one specified as `stylesheet`. Browsers that support alternate style sheets would create a drop-down menu in the browser interface where users can select the other style options by title.

Alternate style sheets are supported in Mozilla, Netscape 6+, and Opera 7+. Internet Explorer Versions 6 and earlier do not support alternate style sheets except through an optional extension called a *favelet* (see *www.favelets.com* for more info). IE7 (in beta as of this writing) does not have built-in support either.

It is possible to use JavaScript and/or server-side scripting such as PHP to give your users the choice of alternate style sheets without relying on the browser's interface to support the alternate style sheet method in the previous example. A List Apart has two articles that serve as a good starting point for learning more.

- "Alternative Style: Working with Alternate Style Sheets," by Paul Sowden (*www.alistapart.com/articles/alternate*). Discusses using JavaScript and the DOM.
- "Build a PHP Switcher," by Chris Clark (*www.alistapart.com/articles/phpswitch/*). Shows how to use PHP to switch style sheets.

Importing allows multiple style sheets to be applied to the same document. When additional `@import` functions are added within the `style` element, the style information from the last file read (the one at the bottom of the list) takes precedence over the previous ones.

The `@import` directive may also be used in the style sheet itself to reference information in other external *.css* files. See the sidebar "Modular Style Sheets" for more information.

`@import` is not supported by Netscape 4, Internet Explorer 3, and Opera 3. This limitation is often used as part of a technique for hiding unsupported style information from these browsers. Fortunately, they make up a small fraction of browsers in use as of this writing.

CSS for Other Media

CSS 2 introduced the ability to target style sheets to specific presentation media. This is done using the `media` attribute in the `link` element or `@media` or `@import`

Modular Style Sheets

The `@import` command may also be used within a style sheet document (the *.css* file) to pull style information in from other style sheets. With this method, one external style sheet attached to the HTML document accesses style rules from multiple *.css* files. This functionality is used strategically as a way to modularize styles and reuse them efficiently.

For example, frequently used styles related to navigation could be stored in a navigation style sheet. Basic typography settings could be stored in another, form styles in another, and so on. These style modules are added to the main style sheet with the `@import` command as shown here:

```
/* basic typography */
@import url("type.css");

/* form inputs */
@import url("forms.css");

/* navigation */
@import url("list-nav.css");
```

rules in a style sheet. The complete list of accepted values for the `media` attribute follows, but currently, only `screen`, `print`, and `all` are widely supported. Support for `handheld` is getting a lot of attention by the W3C's Mobile Web Initiative. Multiple values may be provided in a comma-separated list.

`all`
: Used for all media.

`aural`
: Used for screen readers and other audio versions of the document. This value is deprecated in favor of `speech` in future versions of CSS.

`braille`
: Used when rendering the document with a Braille device.

`embossed`
: Used with Braille printing devices.

`handheld`
: Used for web-enabled cell phones or PDAs.

`print`
: Used for printing the document or for displaying a "print preview."

`projection`
: Used for projection media such as a slideshow presentation.

`screen`
: Used for display on a computer monitor. This is the media that applies to all browsers running on computers.