Cascading Style Sheets Overview:
An Introduction to CSS-1, CSS-2, and CSS-P

CSS Abstract:

The CSS specification is a method for creating more richly styled documents. It offers the ability to create HTML with more specificity and variety. As a general rule CSS is a method of implementing aesthetic change, but may be used with JavaScript or other client side technologies to design interactive and computational web sites.

CSS is often used in architectures requiring HTML, XML, XSL or DHTML.

- CSS Introduction
- How it works
  Inline CSS, Internal CSS, and External CSS
- What it does
  CSS-1 and CSS-2
- Unconditional, Relative, and Conditional Styling
- How it should be used
  Lessons by Example

CSS Introduction

In order to understand the benefits of Cascading Style Sheet (CSS) you must understand the relationship of HTML's history to its present uses. HTML was originally created as a simple, universal language for conveying information between clients over the network we now call the World Wide Web. HTML has become the web publishing standard. HTML 4, the current version of HTML, is a reaction to the perpetually increasing demands of web authors and web surfers.

In order to meet these increasing demands a patchwork of solutions have been created. New HTML tags have been created, existing tags have been extended, and tags that produce overlapping results have corrupted the standards of HTML. In addition HTML clients (like Netscape and Internet Explorer), in a race to win market share, have offered user agent HTML (such as IE's <title> or Netscape's <layer>) that is not part of the HTML specification.

Despite all of these changes the language continues to fall short of many publication needs.

CSS attempts to address these publication needs. CSS extends standard HTML by providing a more comprehensive library of formatting tools. Unlike HTML, CSS was designed with the sole purpose of formatting. In particular, it was designed to format HTML, but offers the ability to style XSL, XHTML, and other emerging technologies. It also affords the ability to create a global styling template for entire HTML documents, or even sites. These templates can be applied to an entire site to apply specific styles to each occurrence of any particular tag. Global styling templates were previously impossible in HTML.

How it works

HTML has admitted limitations when it comes to very specific formatting. While it may be possible to say that an image (<img/>) must be placed directly right of a Heading Tag (<H1/>) HTML doesn't allow specific positioning of that <img> element. CSS allows more specificity than had previously been allowed by HTML. With CSS that same <img/> tag can now be specified to fall 3 Inches below the top of the page and 4 Pixels to the right. This is a decided improvement for HTML designers, and the quality of production web sites.

CSS is commonly expressed in three different ways: *Inline CSS*, *Internal CSS*, and External CSS (aka CSS Templates).

Inline CSS is simply the use of the Style attribute from within a valid HTML tag. Here is an example:

```
<a href  = "http://www.bigplace.com" style  = "font-size:50pt;position:absolute"> </a>
```

Inline CSS offers little benefit from standard HTML attributes when it comes to maintenance. This is because in order to maintain a series of tags, HTML developers must explicitly change each respective style tag. It does however offer the publication improvements offered by CSS.

Internal CSS nests all of the CSS for a single document at the head of that document. *Internal CSS* uses the CSS syntax as specified for CSS template making. Each entry in a CSS template is composed of a *selector*. The *selector* has a *declaration*, which is made of a *property* and a *value*.

In most cases the selector will read the element's name. The property is any valid CSS property, and the value is any valid value for the specified property.

The syntax outline is then:

```
Element  Name, Tag Name, or ID value {
      Attribute: Attribute Value;
}
```

The following example colors every font tag within a single document yellow and makes the font size 12 point. This

```
Font {font-color: yellow;
          font-size: 12pt;
}
```

External Style sheets are CSS templates. The template is a collection of CSS instructions placed in a separate file containing only CSS. It uses the same syntax as internal CSS.

Templates must be explicitly referenced from within an HTML document. The HTML <link> tag is the best way to point HTML to external document. To use External CSS in an HTML document follow the following format:

```
<link rel = "stylesheet" type = "text/css" href ="relative or absolute path to stylesheet" >
```

The CSS Template eliminates the need to alter individual attributes of a subject document. It serves as a central styling rule set. This makes it possible to create a series of simple HTML document and enforce complex stylistic decisions on those documents from one location.

What it does

Precision

CSS supports hundreds of publication needs. The specification supports all of the fundamental HTML 4.0 attributes. It also supports a variety of read-only and client side properties. Examples of read only properties include: First-*letter* selector and the client side *position* element. While it would be impossible to describe all of the CSS properties in a document of this size, the improvements in units of measure, positioning, and accessibility are worth describing in brief.

Units of Measure

CSS extends the standard HTML 4.0 property values by providing more units of measure. These units include but are not limited to; Inch (in), Centimeters (CM), Millimeters (mm), points (pt), Picas, Pixels, Percentages (%).

Position

Another popular publishing boon in CSS is more detailed control over the placement of HTML document elements. HTML 4.0 relies heavily on respective and relative positioning. Save for a few browser specific tricks, there is no way to place the content of a tag without consideration for its adjacent elements in the document's hierarchy.

CSS Positioning introduces the ability to explicitly and accurately position items within the user agent environment. This means that elements can be placed, as publicists would place them: 4 Inches from the Top, 2 cm wide, or even respective of the user agent environment. This ability clearly bridges the gap between the advanced publication tools (like McMaster's Xvision) and HTML.

The three valid position attributes are Relative, Absolute, and Fixed (or Block). The relative property value places elements as they would be placed in standard HTML, respective of their positions within the document hierarchy. Absolute positions the subject tag regardless of overlapping or obscuring other elements. Block treats it's subject tags as a group of tags. This means that if, for instance, a div tag surrounds a set of 4 <td> tags those tags will be changed in harmony by changing the block attribute.

In choosing to absolutely positions an elements there must be a way for the user agent to understand which of an overlapping set of elements must exist "on top" of the others. This is established by the Z-index attribute. The Z-index specifies an integer value within the third axis of the user agent plane, named X. Higher Z-index values will appear in front of lower Z-index values.

Accessibility

Until recently, web pages were not designed to be accessible to the blind or reading disabled. CSS-2 supports an aural stylesheet that allows web pages to be read with precision that rivals the human voice. Aural stylesheet properties include: pitch, pitch-range, pause, play-during, richness and speak. At the writing of this document none of the widely held browsers support aural style sheets.

The aural stylesheet model indicates a growing effort to support the abilities of wide range of web users. This growing support will speculatively be provided by ancillary technologies like CSS.

CSS-1 and CSS-2

Currently there are two CSS specs in existence. These are, unsurprisingly, CSS-1, and CSS-2.

http://www.w3.org/TR/REC-CSS1

http://www.w3.org/TR/REC-CSS2/

Because of the number of client side properties, CSS is intrinsically browser dependent. It is important then to keep in mind which browsers support CSS, to which extent they support it, and which version (CSS-1 or CSS-2) they support.

As a general rule Internet Explorer 4 supports only CSS-1, while Internet Explorer 5 and higher support CSS-2 with respectively increasing ability. Netscape on the other hand had not started full CSS-1 support until Navigator 4.7 and above. Earlier browser version of Netscape's browser may ignore, use incorrectly or even crash a browser or computer by trying to interpret CSS. *We must stress, early version of any browser are prone to irregular or dangerous behavior when using CSS. Developing with CSS is more volatile than HTML. As of the writing of this document CSS bugs are capable of crashing browsers and personal computers.*

It is also important to note that just as rudimentary understanding of English does not allow you to interpret Shakespeare, excellent CSS implementations require more than core CSS support by clients. Some may even warn that like poor interpretations of Shakespeare, CSS support may be dangerous.

The following is a table of General CSS1 Support as calculated by data from O'Reilly's Definitive Guide to CSS, RichinStyle.com, and CSS.NU.

1	Internet Explorer 5 for the
2	Macintosh
3	Oprah 3.5 for Windows
4	Internet Explorer 5 for Windows
5	Netscape 4.7 for Windows
	Internet Explorer 4 for Windows

CSS is also one of the 3 elements required for Dynamic HTML (DHTML). DHTML is a combination of JavaScript, CSS, and HTML to create web pages that react to events within a client. These events may be as simple as the popular "onmouseover event" or as complicated as GUI developer can imagine. For more information on DHTML we suggest reading Microsoft's Documentation or visiting Webreference.com for impartial source.

Internet Explorer 5 and above actually supports client side styling with CSS. This means that documents can be shipped to the browser and can be displayed via an external stylesheet on the client computer. Such support extends to both HTML and XML documents. The benefits of such styling are beyond the scope of this discussion, but there are clear advantages to this in future developments architectures.

Any discussion of support must also include an explanation of the languages other than HTML that can be used with CSS. As of the writing of this document only three types of documents are commonly styled by CSS. These are HTML, XHTML, and XML documents. In general CSS is a viable solution for web based mark-up languages and is supported accordingly.

Unconditional, Relative, and Conditional Styling

Most of the aesthetic definitions created by CSS will be unconditional style decisions. These are simply properties of an element that are set as a cascading style sheet is enforced on a document.

Relative styling is the dependence of one property or attribute on another. You may for example decide to make an image as wide as the browser window or you may simply set the width of an image equal to the width of the browser. When the document is rendered the image will become the width of the browser window, but when the browser window is resized the document should remain the same size.

This type of styling is distinct from the HTML percentage attribute value for 2 reasons. CSS allows relative attribute population of most property values, not just width and height. This means that font sizes can be respective of other font sizes, or more practically, of image and icon sizes. Secondly, relative positioning occurs once, and only once. Unlike CSS, the HTML percentage display may change after the document is rendered.

Conditional styling, or event based styling, is a more advanced application of CSS. Somewhere between simple DHTML and JavaScript, conditional styling uses simple events to calculate property values. In the above listed image example, once you have added the code that evaluates for the event, "browser resize", and set the image accordingly, you have created conditional styling.

Conditional styling is more commonly part of Dynamic HTML (DHTML). DHTML is nothing more than a glamorous name for a combination of 3 core technologies; Javascript, CSS, and HTML. DHTML creates web pages that react to events received by the web client. These events may be as simple as the popular "onmouseover" event or as complicated as a developer can imagine. DHTML events are analogous to Visual Basic events, sans event values.

DHTML is an exciting, but extremely evasive technology that has gained wide acceptance, but limited browser support. At the writing of this document the number

of events provided by Internet Explorer 5.5 is most exhaustive. Internet explorer also supports the widest array of CSS property *sets* and *gets*. Netscape Navigator 4 is the only browser to offer several convenience methods for positioning. Given this information it is generally accepted that Internet 5.5 provides the best DHTML support, while Netscape Navigator 4 trails from a considerably distance.

As of the writing of this document DHTML support is buggy and inconsistent across all browsers, with the widest disparity occurring again between Internet Explorer 5.5 and Netscape Navigator 4.

DHTML is instrumental in creating web browser rendered environment that looks and feel like a traditional desktop application. For this reason many web applications are built in part on DHTML designs.

For more information on DHTML I'd suggest reading Microsoft's Documentation or visiting WebReference.com for an impartial source.

How it should be used

Microsoft has been using CSS with XML for years as the foundation Microsoft Word transformations. Open the source code of an MS-Word document that has been transformed to HTML and you will notice Namespace definitions (an XML convention) and *Internal CSS*.

There is a lot to learn from Microsoft's example. To start the documents are rendered wonderfully in recent versions of Internet Explorer. They are not however, rendered very well in Netscape Navigator. The most common problems include unrecognized syntax and unparsed tags.

Lesson 1: CSS should be used very carefully and precisely to ensure browser compatibility.

Microsoft also refrains form using external style sheets to style their documents. The most important reasons for this should be very obvious. An external style sheet must be reference, so either the style sheet must ship with documents on every move, or the reference must be dynamically maintained. Another less obvious, and less important, reason is that the CSS document references vary slightly between browsers. One common difference is that some browsers require the .css extension on an externally referenced sheet, while others do not.

Lesson 2: CSS designs must reflect its uses.

Documents designed for portability are probably best paired with *Internal CSS* or in-line styling. Documents designed for maintainability are best used with External or *Internal CSS*. Documents designed for the highest levels of interoperability are best used with *Internal CSS* where unrecognized style tags can most easily be ignored.

Now purists might notice that while Microsoft does not reference the CSS documents externally they may create a directory with at least an XML file. Discussing the benefits of this are out side of the scope of our discussion of CSS, but it's interesting to note that this dependency exists.

The final lesson is not necessarily to be learned from Microsoft's example, but is proven in all web design.

Lesson 3: Use technology appropriately.

It is often not worth the risk and volatility of CSS attributes, if your needs are met easily by other technologies. XHTML will probably increase designer's dependency on CSS or CSS like technologies, but the core designer questions must always be asked. What are my reasons for using CSS instead of another technology? Is the design improved or does it require the maintainability and publication tools provided by CSS?