

Introduction to Cascading Style Sheets (CSS)

CSS is a method of presenting data in a web page. CSS offers web developers methods to control the presentation of data with many more options offered by HTML alone. One of its major strengths is that a web developer can create a stylesheet (filename.css) save it with the other web pages which are then linked to this style sheet. A web developer can update or change one stylesheet and thereby alter the presentation of all the web pages on a web site -- it is particularly useful in constructing large or corporate web sites. CSS has some limitations, the first is that not all browsers support all the features so it is important to check your web pages on several browsers or at least the one used by your clients. You will also need a reference book or web site as there are a lot of properties used to format the various elements on a web page. Although CSS is still under development – but it is ready for prime time use now.

This introduction to CSS will show you where to place scripts, the basic syntax and get you started with some real world applications that you can use to enhance or build your own web site. Before learning CSS you should already have a good familiarity with HTML and Web Development tools like Dreamweaver can help you learn and write CSS for your web pages.

1. CSS Basics

- 1.1 Understand what CSS is, and why its useful
- 1.2 Understand Selectors, Properties and Values
- 1.3 Add <style> tags to a web page
- 1.4 Write some simple CSS statements and place them a) Inline b) Head section of a web page and c) in an external script
- 1.5 Understand how to import CSS into your web page
- 1.6 Working with classes, pseudo classes
- 1.7 Work with ids
- 1.8 Writing CSS to modify font properties
- 1.9 Write CSS to create rollovers and hovers on simple hyperlinks

2. The Box model of page formatting

- 2.1 Basics Divs and Spans
 - 2.2 Classes and Ids and pseudo classes
 - 2.3 Rounded corners
 - 2.4 Adding drop shadows
-
3. Working with Lists
 4. Modifying Background Properties of a Web page
 5. Working with and formatting form fields
 6. Scroll Bars

A **style** is a definition of fonts, colors, etc.

Each style has a unique name: a **selector**.

The **selectors** and their **styles** are defined in one place.

In your HTML contents you simply refer to the **selectors** whenever you want to activate a certain **style**.

For example:

Instead of defining fonts and colors each time you start a new table cell, you can define a style and then, simply refer to that style in your table cells.

Compare classic HTML with CSS

```
<body><table>

<tr><td bgcolor="#FFCC00" align="left"><font face="arial" size="2"
color="red"><b>this is line 1</b></font></td></tr>

<tr><td bgcolor="#FFCC00" align="left"><font face="arial" size="2"
color="red"><b>this is line 2</b></font></td></tr>

<tr><td bgcolor="#FFCC00" align="left"><font face="arial" size="2"
color="red"><b>this is line 3</b></font></td></tr>

</table></body>
```

Here in CSS Format

```
<head>
<style type="text/css">
.subtext { font-family: arial;
           font-size: 10px;
           color: red;
         }
</style>
</head>
```

```
<body>
```

```
<table>
<tr><td class="subtext">this is line 1</td></tr>
<tr><td class="subtext">this is line 2</td></tr>
<tr><td class="subtext">this is line 3</td></tr>
</table>
</body>
```

The one disadvantage of CSS is:

- **these will only work on version 4 browsers or newer**. However, more than 95% of all browsers live up to that
SELECTORS

Selectors are the names that you give to your different styles you refer to these selectors in the body of your HTML document to apply these styles..

There are three types of Selectors

1. HTML selectors – you define the properties for specific HTML tags, e.g. P, H1, B, etc
2. Class selectors – used to define styles without referring to a particular tag, they can be used to apply the same style to many different HTML tags
3. ID selectors are used to define unique parts of a web page e.g. footer, header, table or other unique element – each Id element is unique and can only be used once

HTML Tag Selectors

HTML Selector { property: value; property2: value2; etc }

Note the Selector is usually an HTML tag like H1, B, P etc

Property: value are always separated by a colon, additional property:values; are separated by semicolons. When the code is written inside the <head></head> tags the property: values; are often typed vertically so its easier to read them as a list e.g.

```
<html>
<title>Simple style</html>
<head>
```

```
<style type="text/css">
<!--
```

```
H1 { font-family: arial;
      font-size: 20px;
      color: red;
    }
```

```
-- >
</style>
</head>
<body>
```

<h1>This text will be styled according to the properties defined in the style tags in the head </h1>

```
</body>
</html>
```

Note the <!-- -- > are HTML comment tags are used so that if someone visits your web page using an older web browser that does not recognize or support styles the code will be ignored.

CLASS SELECTORS

.ClassSelector (Property: Value }

Class selectors are preceded by dot followed by a name and then a series of property: values.

Class selectors can have almost any name, usually you should pick on that describes it function e.g.

```
.redBold { color: red;
           font-weight: bold;
         }
```

This class function can then be used to apply this style to several different HTML tags in the body of your web page e.g.

```
<p class="redBold">this text will be red and bold</p>
```

```
<h2 class="redBold">This text will also be red and bold </p>
```

Class selectors are used when you want to define a style that does not redefine an HTML tag entirely it just modifies or enhances it.

PSEUDO CLASS SELECTORs

```
B.redBold { color: red; font-weight bold; }
```

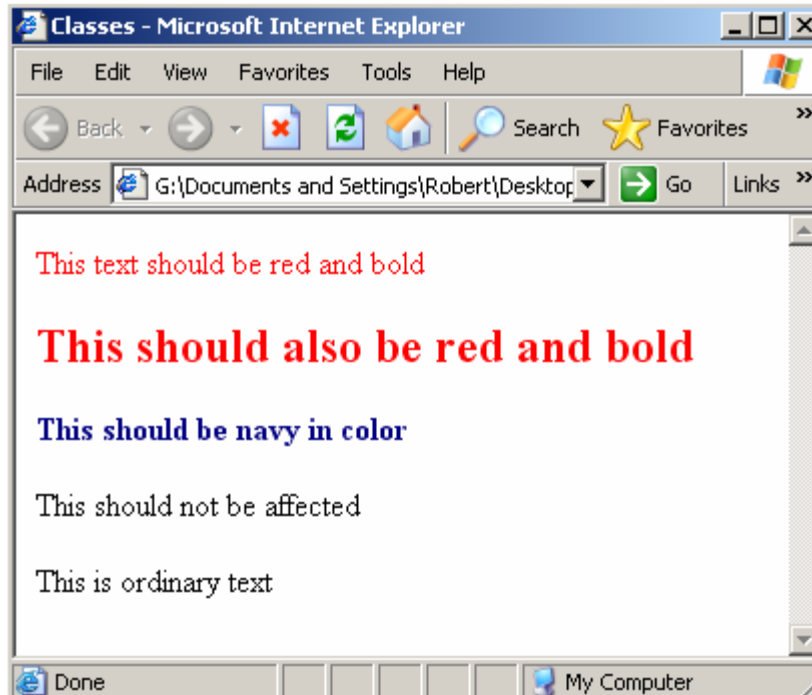
Pseudo class selectors include the specific selector followed by a period and a name and property: values – these property values can only be applied to the specific selector you defined

E.g.

```
<html>
<head>
<title>Classes</title>
<style type="text/css">
.redBold { color: red }
B.navy {color: navy}
</style>
</head>
<body>
<p class="redBold"> This text should be red and bold</p>
<h2 class="redBold"> This should also be red and bold</h2>
<p><b class="navy">This should be navy in color</b></p>
<p class="navy">This should not be affected</p>
<p>This is ordinary text</p>

</body>
</html>
```

The appearance of your web page is show below (I.E 6.0)



Note using the pseudo class to try and redefine the paragraph tag does not work and leaves the text unformatted.

When referring to a Class selector you simply add the class to an HTML tag like in the above example (**class="classname"**) do not include the period before the name.

SPAN and DIV as carriers

Two tags are particularly useful in combination with class selectors: **** and **<DIV>**.

Both are "dummy" tags that don't do anything in themselves. Therefore, they are excellent for carrying CSS styles.

**** is an "inline-tag" in HTML, meaning that no line breaks are inserted before or after the use of it.

-need some examples

<DIV> is a "block tag", meaning that line breaks are automatically inserted to distance the block from the surrounding content (like **<P>** or **<TABLE>** tags).

<div align="justify">Small businesses are no longer restricted to providing services locally but can expand their distribution range to other regions of the country or the world. Calgary has one of the highest populations of Internet users making it ideal for doing business online. The costs for expanding your business on line have never been lower. If you are **</div>**

This will justify the text inside the div style tags.

ID SELECTORS

The general syntax for an ID selector is:

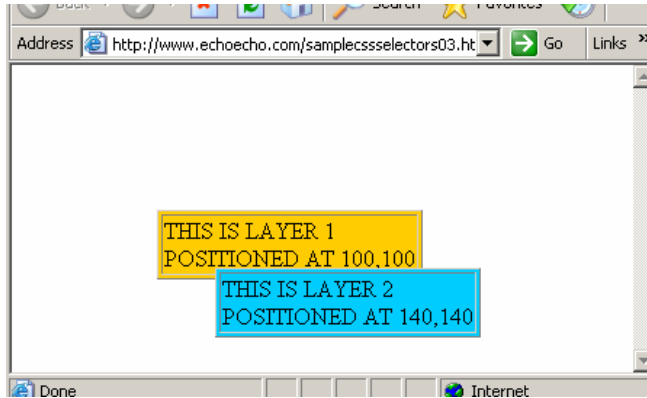
#IDSelector {Property:Value;}

Idselectors are preceded with the # hash symbol and are unique – in otherwords you can only have one specific idselector on a web page. They are used to define specific “boxes” on the page or layers.

```
HTML>
<HEAD>
<style type="text/css">
#layer1 {position:absolute; left:100;top:100; z-Index:0}
#layer2 {position:absolute; left:140;top:140; z-Index:1}
</style>
</HEAD>

<BODY>
<div ID="layer1">
<table border="1" bgcolor="#FFCC00"><tr><td>THIS IS LAYER
1<br>POSITIONED AT 100,100</td></tr></table>
</div>

<div ID="layer2">
<table border="1" bgcolor="#00CCFF"><tr><td>THIS IS LAYER
2<br>POSITIONED AT 140,140</td></tr></table>
</div>
</BODY>
</HTML>
```



ID selectors are used when you want to define a style relating to an object with a unique ID.

GROUPED SELECTORS

Most often selectors will share some of the same styles, for example, being based on the same font.

In these cases, rather than defining the font for each and every selector, one by one, you can group them, and thus assign the font to all the selectors at once.

Look at this example, made without grouping

```
.headlines{  
font-family:arial; color:black; background:yellow; font-size:14pt;  
}  
  
.sublines {  
font-family:arial; color:black; background:yellow; font-size:12pt;  
}  
  
.infotext {  
font-family:arial; color:black; background:yellow; font-size:10pt;  
}
```

As you can see, the only style that varies is the **font-size**.

In the next example we have grouped the selectors, and defined the common styles at once

```
.headlines, .sublines, .infotext {  
font-family:arial; color:black; background:yellow;  
}  
  
.headlines {font-size:14pt;}  
.sublines {font-size:12pt;}  
.infotext {font-size: 10pt;}
```

CONTEXT DEPENDANT SELECTORS

It is possible to make selectors that will only work in certain contexts.

For example, you can define a style for the **** tag that is only triggered if the text is not only bold but also written in italics.

For example, the style should be in effect here:

THE SYNTAX

Simply adding a normal style to the **** tag is done like this:

```
B {font-size:16px}
```

Adding a context dependent style, like the one described above is done like this:

```
I B {font-size:16px;}
```

USING GROUPED AND CONTEXT DEPENDENT SELECTORS AT THE SAME TIME:

It is possible to use context dependent and grouped selectors at the same time.

For example, like this:

In the example the font-size of 16 pixels is in effect on:

- 1) All **** tags enclosed by **<I>** tags
- 2) All headlines classes.
- 3) sublines classes enclosed by **** tags.

WHERE TO PLACE IT

CSS can be added to your pages at 3 different levels.

1. Inline – the selector affects only the specific tag it surrounds inside the page
2. Defined in the **<head></head>** section of a web page the selector can be used to modify many tags in this page only.
3. Styles are defined in an external page file.css and linked to each web page these can cascade and affect tags in all of the pages of a web site.

In actual practice web developers will often use a combination of these. The style closest to the tag has precedence, so an inline tag will over ride one in the head or one defined in an external style sheet.

NOT ME This is an example of an inline style and it affects the text only in this line – the style is specific to the words NOT ME

You should limit your use of single tag CSS.

If you define your styles for each and every tag they're used on, you will lose much of the power associated with CSS.

For example, you will have to define the style over and over again whenever it's used, rather than just defining it once and then referring to that one definition whenever it's used.

Furthermore, if you wanted to change a certain style, you'd have to change it all over in your document, rather than in one place.

DEFINE Styles in HEAD and they affect only SINGLE PAGES

```
<html>
<head>
<title>MY CSS PAGE</title>
<style type="text/css">
.headlines, .sublines, infotext {font-face:arial; color:black; background:yellow;
font-weight:bold;}
.headlines {font-size:14pt;}
.sublines {font-size:12pt;}
.infotext {font-size: 10pt;}
</style>
</head>

<body>
<span class="headlines">Welcome</span><br>

<div class="sublines">
This is an example page using CSS.<br>
The example is really simple,<br>
and doesn't even look good,<br>
but it shows the technique.
</div>

<table border="2"><tr><td class="sublines">
As you can see:<br>
The styles even work on tables.
```

```
</td></tr></table>
```

```
<hr>
```

```
<div class="infotext">  
Example from EchoEcho.Com.  
</div>
```

```
<hr>  
</body>  
</html>
```

By defining styles for entire pages, you will gain the freedom to easily change the styles even after the entire page has been made. If the web site is small (under 20 web pages) you could create a template page where you define your styles in the head and then save this template as your different web pages. More than 20 pages and it becomes easier to modify the styles if you place them inside an external text document e.g. mystyles.css

ENTIRE SITES – Define styles in an external stylesheet and link this sheet to all of your web pages.

```
<html>  
<head>  
<title>MY CSS PAGE</title>  
<link rel="stylesheet" href="whatever.css" type="text/css">  
</head>  
<body>  
<span class="headlines">Welcome</span><br>
```

```
<div class="sublines">  
This is an example of a page using CSS.<br>  
The example is really simple,<br>  
and doesn't even look good,<br>  
but it shows the technique.  
</div>
```

```
<table border="2"><tr><td class="sublines">  
As you can see:<br>  
The styles even work on tables.  
</td></tr></table>
```

```
<hr>
```

```
<div class="infotext">Example from EchoEcho.Com.</div>
```

```
<hr>
</body>
</html>
```

External Style sheet called whatever.css is stored in the root along with your other web pages.

```
headlines, .sublines, infotext {font-face:arial; color:black; background:yellow; font-weight:bold;}
.headlines {font-size:14pt;}
.sublines {font-size:12pt;}
.infotext {font-size: 10pt;}
```

Now if you just add the line **<link rel=stylesheet href="whatever.css" type="text/css">** to the **<head>** of all your pages, then the one style definition will be in effect for your entire site

Note it is possible to link to more than one stylesheet, some web developers break their stylesheets up into separate documents, one for fonts, one for forms etc. The only advantage is one of organization.

At this point of the tutorial you should know:

- 1: how to define styles for tags, classes and objects with ID's.
- 2: how to group styles and make them context dependent
- 3: how to add styles to single tags, single pages and entire sites

All we need now is a walkthrough of the various style attributes that can be assigned.

We will divide them into three categories

- 1: Inline attributes. (Works on tags like: , and <I>).
- 2: Block attributes. (Works on block tags: <DIV>, <TD> and <P>).
- 3: Link attributes. (Works on links and use a special syntax).

These options can entirely replace the **** tag, but there's even more. CSS allows you to define these styles much more powerfully than you could ever do with plain HTML.

FONT PROPERTIES

Property	Values	NS	IE	Example
----------	--------	----	----	---------

font-family	font name	4+	4+	font-family:arial
	generic font	4+	4+	font-family:arial, helvetica
font-style	normal	4+	4+	font-style:normal
	italic	4+	4+	font-style:italic
	oblique		4+	font-style:oblique
font-variant	normal		4+	font-variant:normal
	small-caps		4+	font-variant:small-caps
font-weight	normal	4+	4+	font-weight:normal
	bold	4+	4+	font-weight:bold
	bolder	4W	4+	font-weight:bolder
	lighter	4W	4+	font-weight:lighter
	100-900	4+	4+	font-weight:250
font-size	normal	4+	4+	font-size:normal
	length	4+	4+	font-size:14px
	length	4+	4+	font-size:14pt
	absolute	4+	4+	font-size:xx-small
	absolute	4+	4+	font-size:x-small
	absolute	4+	4+	font-size:small
	absolute	4+	4+	font-size:medium
	absolute	4+	4+	font-size:large
	absolute	4+	4+	font-size:x-large
	absolute	4+	4+	font-size:xx-large
	relative	4+	4+	font-size:smaller
	relative	4+	4+	font-size:larger
	percentage	4+	4+	font-size:75%

TEXT PROPERTIES

Despite the **font properties** listed above there are some options for defining text properties such as alignments, underlines, etc.

Property	Values	NS	IE	Example
line-height	normal	4W	4+	line-height:normal
	number	4+	4P	line-height:1.5
	length	4+	4+	line-height:22px
	percentage	4+	4P	line-height:150%
text-decoration	none	4+	4M	text-decoration:none
	underline	4+	4+	text-decoration:underline
	overline		4W	text-decoration:overline
	line-through	4+	4+	text-decoration:line-through
	blink	4+		text-decoration:blink

text-transform	none	4+	4W	text-transform:none
	capitalize	4+	4W	text-transform:capitalize
	uppercase	4+	4W	text-transform:uppercase
	lowercase	4+	4W	text-transform:lowercase
text-align	left	4+	4+	text-align:left
	right	4+	4+	text-align:right
	center	4+	4+	text-align:center
	justify	4+	4W	text-align:justify
text-indent	length	4+	4+	text-indent:20px;
	percentage	4+	4+	text-indent:10%
white-space	normal	4+		white-space:normal
	pre	4+		white-space:pre

Note:

line-height :

When using a number (such as 1.5) the number refers to the font size, where 1.5 would mean that a 1.5 lines spacing (using the current font size) will be inserted between the lines.

text-transform :

Capitalize sets the first letter of each word in uppercase.

Uppercase forces all letters to uppercase.

Lowercase forces all letters to lowercase.

text-indent :

Use this to indent the first word of a paragraph.

white-space :

If white-space is set to **pre** the browser will show all spaces in the text, rather than ignoring all occurrences of more than one space. This is similar to the **<pre>** tag in plain HTML. Since the white-space is only supported by NS you should use the **<pre>** tag instead.

COLORS

As you can see, the above CSS properties can replace all text formatting that can be done with plain HTML with one exception: the color.

The color is not part of the font collection in CSS - rather it has its own definition. If you want to add a color to the text in the above example you'd do it this way:

```
B {font:arial, helvetica 12px bold; color:red}
```

CSS has several options for defining colors of both text and background areas on your pages.

These options can entirely replace the color attributes in plain HTML. In addition, you get new options that you just didn't have in plain HTML.

For example, in plain HTML, when you wanted to create an area with a specific color you were forced to include a table. With CSS, you can define an area to have a specific color without that area being part of a table.

Or even more useful, in plain HTML when working with tables, you had to specify font attributes and colors etc. for each and every table cell. With CSS you can simply refer to a certain class in your **<TD>** tags.

COLOR PROPERTIES

Property	Values	NS	IE
color	<color>	4+	4+
background-color	transparent	4+	4+
	<color>	4+	4+
background-image	none	4+	4+
	url(<URL>)	4+	4+
background-repeat	repeat	4+	4+
	repeat-x	4+	4+
	repeat-y	4+	4+
	no-repeat	4+	4+
background-attachment	scroll		4+
	fixed		4+
background-position	<percentage>		4+
	<length>		4+
	top		4+
	center		4+
	bottom		4+
	left		4+
background	right		4+
	<background-color>	4+	4+
	<background-image>	4+	4+
	<background-repeat>	4+	4+
	<background-attachment>		4+
	<background-position>		4+

Setting colors

Basically you have three color options with CSS:

- 1: Setting the foreground color for contents
- 2: Setting the background color for an area

3: Setting a background image to fill out an area

In the next section we will list the different properties that let you do that.

In plain HTML, colors can either be entered by name (red, blue etc.) or by a hexadecimal color code (for example: #FF9900).

With CSS you have these options:

Common name

You can define colors with the use of common names, by simply enter the name of the desired color.

For example:

```
.myclass {color:red; background-color:blue;}
```

Hexadecimal value

You can define colors with the use of hexadecimal values, similar to how it's done in plain HTML.

For example:

```
.myclass {color:#000000; background-color:#FFCC00;}
```

RGB value

You can define colors with the use of RGB values, by simply entering the values for amounts of Red, Green and Blue.

For example:

```
.myclass {color:rgb(255,255,204); background-color:rgb(51,51,102);}
```

You can also define RGB colors using percentage values for the amounts of

Red, Green and Blue:

For example:

```
.myclass {color:rgb(100%,100%,81%); background-color:rgb(81%,18%,100%);}
```

Setting background colors

Background colors are defined similar to the colors mentioned above. For example you can set the background color of the entire page using the **BODY** selector:

```
BODY {background-color:#FF6666;}
```

Setting a background image

CSS lets you set a background image for both the page and single elements on the page.

In addition, CSS offers several positioning methods for background images.

You can define the background image for the page like this:

```
BODY {background-image:url(myimage.gif);}
```

You can control the repetition of the image with the **background-repeat** property.

background-repeat:repeat

Tiles the image until the entire page is filled, just like an ordinary background image in plain HTML.

background-repeat:repeat-x

Repeats the image horizontally - but not vertically.

background-repeat:repeat-y

Repeats the image vertically - but not horizontally.

background-repeat:no-repeat

Does not tile the image at all.

Positioning a background

Background positioning is done by entering a value for the left position and top position separated by a space.

In this example the image is positioned 75 pixels from the upper left corner of the page:

```
BODY {background-image:url(myimage.gif); background-position: 75px 75px;}
```

Note: Background positioning is not supported by Netscape 4 browsers.

Fixing a background

You can fixate an image at a certain position so that it doesn't move when scrolling occurs.

```
BODY {background-image:url(myimage.gif); background-attachment: fixed;}
```

Note: Background fixation is not supported by Netscape 4 browsers.

Setting multiple background values

Rather than defining each **background** property with its own property you can assign them all with the use of the **background** property.

Look at this example:

```
BODY {background:green url(myimage.gif) repeat-y fixed 75px 75px;}
```

CSS Links

Property	Values	NS	IE
A:link	<style>	4+	4+
A:visited	<style>	4+	4+
A:active	<style>	4+	4+
A:hover	<style>	6+	4+

DEFINING STYLES FOR LINKS

As mentioned in the above table, there are four different selectors with respect to links.

You can specify whatever style you'd like to each of these selectors, just like you'd do with normal text.

Here you can see a few examples on how CSS can be used to replace the traditional image based mouseover effects for links.

The **hover** style is not supported by Netscape browsers prior to version 6, but since it does no harm, you can still use it for the benefit of the +90% of visitors that arrive using MSIE).

One of the most common uses of CSS with links is to remove the underline. Typically it's done so that the underline appears only when a hover occurs. In the example below, we did just that. In addition we added a red color for hovered links.

```
<style type="text/css">
A:link {text-decoration: none}
A:visited {text-decoration: none}
A:active {text-decoration: none}
A:hover {text-decoration: underline; color: red;}
</style>
```

MULTIPLE LINKSTYLES ON SAME PAGE

The final topic deals with how to add multiple link styles that can be used on the same page.

In the above examples we addressed the HTML selector - A:link etc - and thus redefined the overall link style.

How do we define a link style that is only active in a certain area of the page?

The answer is: [context dependent selectors](#).

Rather than addressing the **A:link** selector we will address it while being dependant on a certain outer class that surrounds the area where we'd like our link style to be effective.

```
<html>
<head>
```

```

<style type="text/css">
.class1 A:link {text-decoration: none}
.class1 A:visited {text-decoration: none}
.class1 A:active {text-decoration: none}
.class1 A:hover {text-decoration: underline; color: red;}

.class2 A:link {text-decoration: underline overline}
.class2 A:visited {text-decoration: underline overline}
.class2 A:active {text-decoration: underline overline}
.class2 A:hover {text-decoration: underline; color: green;}
</style>
</head>

<body>
ONE TYPE OF LINKS
<br>
<span class="class1">
<a href="http://www.yahoo.com">YAHOO</a>
<br>
<a href="http://www.google.com">GOOGLE</a>
</span>
<br>
<br>
ANOTHER TYPE OF LINKS
<br>
<span class="class2">
<a href="http://www.yahoo.com">YAHOO</a>
<br>
<a href="http://www.google.com">GOOGLE</a>
</span>
</body>
</html>

```

Note how we use the **** to define the context.

This is smart for two reasons:

1) The obvious, that it allows us to use different link styles on the same page, rather than being limited to using a single overall link style.

2) We can define entire areas where a certain link style works for all links within that area. Thus, we don't have to add a style definition to each and every link in that area.

CSS Lists

CSS allows you to customize the lists that can be made with HTML.

The good news is that there are many powerful properties for doing so.

The bad news is that Netscape and Internet Explorer often support these properties in different ways. Both browsers have limitations in their support of list styles.

Netscape browsers only let you add the list CSS to tags - not just any tag.

Internet Explorer's support of CSS with relation to lists is only fully supported for browsers on the Windows platform.

In any case, be careful about using CSS for lists since it might not show the way you want it to on all browsers. However, most things won't ruin anything if the browser doesn't support it - it just shows as a normal list - so it will be okay to define lists that only work for the most widely used browser.

LIST PROPERTIES

Property	Values	NS	IE
list-style type	disc	4+	4W
	circle	4+	4W
	square	4+	4W
	decimal	4+	4W
	lower-roman	4+	4W
	upper-roman	4+	4W
	lower-alpha	4+	4W
	upper-alpha	4+	4W
	none		4W
list-style image	none		4W
	url(<url>)		4W
list-style position	outside		4W
	inside		4W
list-style	<list-style type>		4W
	<list style position>		4W
	<list-style image>		4w

DEFINING STYLES FOR LINKS

As mentioned in the above table, we have four unique selectors with respect to lists. The fourth selector, **list-style** is an overall selector that let you define all list

related styles at once.

The three basic selectors are

list-style type

Defines the look of the bullets used in your list.

list-style image

Let's you use a custom graphic for bullets.

list-style position

Often the text in a list is longer than one line.

list-style position:outer lets the second line align with the first line. That is: the bullet is to the left of both lines.

list-style position:inner lets the second line align with the bullet.

Assigning several properties at once

Instead of using different selectors for each list-style you can specify them all at once using the **list-style** property.

For example

```
<html>
<head>
<style type="text/css">
LI.list1 {list-style: circle outside; color:green;}
LI.list2 {list-style: square inside; color:blue}
.blacktext {color:black}
</style>
</head>

<body>

<ul>
<li class="list1"><span class="blacktext">This is one black line</span>
<li class="list1">This is another line that is much longer than the first. But it isn't a
black line since we did not specify a style for the text that goes here other than
the style we defined for the list.
</ul>
<br>
<br>
<ul>
<li class="list2"><span class="blacktext">This is one black line</span>
<li class="list2">This is another line that is much longer than the first. But it isn't a
black line since we did not specify a style for the text that goes here other than
the style we defined for the list.

</ul>
```

```
</body>  
</html>
```