Introduction to CSS

Style sheets (or to give them their full title, cascading style sheets or CSS) are becoming more and more common on the Web. They were first introduced in a limited form to Internet Explorer 3.0, and are recognised by all version 4 browsers.

This tutorial will show you what style sheets are, when to use them, and (hopefully) how to use them!

What are style sheets?

Style sheets are a way to separate style from content in Web pages. In an ideal world, you would put all your content (e.g. text and graphics) in one place, and define how that content is laid out (the style) in another.

Straight HTML mixes style with content. For example, each time you want a bold, centred paragraph to represent a quote, you have to have to use **<p align="center">** and **** tags to achieve the effect. You're mixing what you want to say, with how you want to say it.

The markup metalanguage, XML, will hopefully move us towards total content and style separation. In the meantime, we have the pretty good compromise of style sheets.

Style sheets allow you to modify the default attributes of many standard HTML tags. For example, the **** tag normally just makes text bolder, as it is doing here. However, you could use a style sheet to modify the **** tag to make text red, all in capitals, or one font size bigger. It's totally up to you.

Even better - if you make all documents use the same style sheet, you only have to change the properties for **** in the single style sheet file, and all the documents in your website will display **** in the new way! Cool, huh?

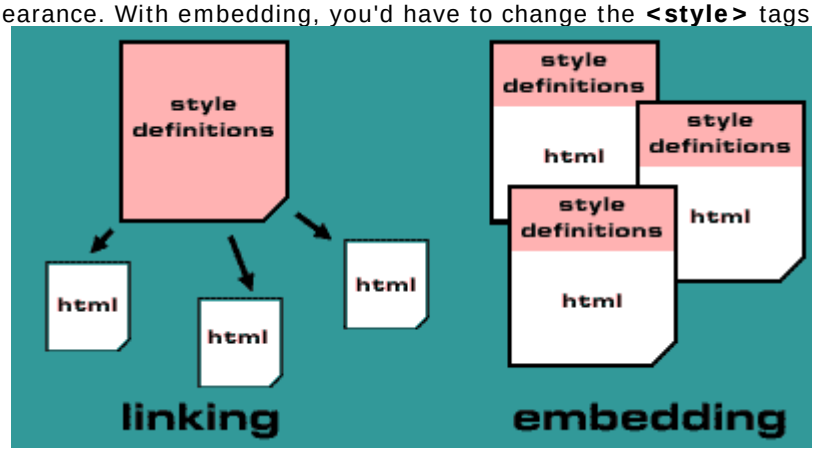
For example, if you're using IE 3+ or Netscape 4+, you'll notice that the headings on this page are in a dark green colour, and that there's a gap between each heading and the paragraph above it. You'll also notice that the links are yellow, and that if you use IE4+ or Netscape 6+ the links change from yellow to white as the mouse rolls over them.

Lovely! How do I use them?

There are two main ways to use style sheets - linked and embedded. (You can also place styles straight into the thick of the HTML itself - called inlining - but this is less frequently used and won't be covered here. There's not enough space!)

You can create a style sheet in a separate file and then link one or more web pages to it - this is called **linking**, amazingly enough. Or, you can embed style definitions directly into the **<head>** section of individual web pages, using the **<style>** tag - we humans call this **embedding**.

Now it doesn't take a genius to work out that linking is more powerful than embedding. If you link many pages to one style sheet file, you only have to change a style in that one file, and all the linked pages will change their appearance. With embedding, you'd have to change the **<style>** tags for each page...



However, embedding can be used in tandem with linking. Say you've done a perfect style sheet which makes everything in **** tags appear blue, and 100 pages link to it. But then you realise that your 99th page would look much better if the stuff in **** looked red, just for that one page! No problem - then you embed a style into just that page, using the **<style>** tag. Embedded styles override the styles defined in the linked style sheet, so for this one page the bold stuff will look red, not blue. Clever! (This is why they're called cascading style sheets, by the way.)

So you get the general idea. Let's get stuck in.

The nuts and bolts of it all

Let's examine a simple style sheet, and then see how you would link a web page to it.

```
p { text-align: justify;
   font-family: Verdana, Arial, Sans-Serif; }

h4 { color: #ff4400; margin-top: 45px; }

A:link, A:visited, A:active { text-decoration: none;
   font-weight: bold; color: #0000ff }

A:hover {color:red;text-transform:uppercase;}
```

This style sheet is divided into four sections. Each section redefines a standard HTML attribute:

The **p** section affects all paragraphs (enclosed with ** ** and ** **) and makes them justified with the left and right margins. It also specifies the preferred fonts. The left-most font is used if your computer has it. If not, the browser works through the list from left to right until it finds the font; it's a good idea to put either **serif** or **sans-serif** at the end to specify a generic serifed or non-serifed font respectively.

The **h4** section redefines level 4 headings. Any text within an **<h4></h4>** tag will now be coloured orange, and there will be a 45 pixel gap between the heading and the end of the paragraph above.

The **A:link**, **A:visited**, **A:active** section defines the attributes for the normal, visited and active (clicked on) links in the page. **text-decoration:none** removes those nasty underlines from under the link text. **font-weight:bold** and **color: #0000ff** should be self-explanatory.

The last section, A:hover, only works for IE4+ and Netscape 6+. It does a mouse rollover effect for the text links. Move the mouse over a link and it will turn red and all the letters become uppercase. Now there's no mistaking where the links are in your page! :)

That's the style sheet - now how do we link our pages it to it? Simple - with one line in the **<head>** section:

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

This assumes the style sheet file is in the same directory as the web page, and it is called **style.css**. If it was in the directory above, we would of course use **href="../style.css"**.

That's how you link to a style sheet. Suppose instead that we wanted to embed the above styles. We'd put them in the <head> section of our web page, within <style> tags:

```
<head>
<title>Web page with embedded styles</title>

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

```
<!--

   p { text-align: justify; font-family:

   Verdana, Arial, Sans-Serif; }



   h4 { color: #ff4400; margin-top: 45px; }



   A:link, A:visited, A:active { text-decoration: none;

   font-weight: bold; color: #0000ff; }



   A:hover {color:red;text-transform:uppercase;}

-->

</style>



</head>



<body>

.

.

.
```

Note that it's a good idea to put comment tags (**<!-- -->**) around the style definition, to hide it from browsers that don't understand the **<style>** tag.

That's how you embed a style sheet. Don't forget that you can link to another stylesheet as well as embed one in your page, but the embedded style definitions will override the linked ones.

Classes

For further control over the style of your web pages, it's possible to create **classes** of tags. For example, the **<p>** tag could have a class, **quote**, specifically for quotations. This class could indent both sides of the paragraph and make the text italicised.

The style sheet would look like this:

```
p.quote { margin-left: 20px; margin-right: 20px;
    text-decoration: italic; }
```

... and you'd reference the **quote** class in your HTML like this:

```
<p class="quote">
In the beginning God created the heavens and the earth.
Now the earth was formless and empty, darkness was over
the surface of the deep, and the Spirit of God was
hovering over the waters.
</p>
```

... which would make this modern translation from the Masoretic Text appear indented on both sides and italicised!

More fun with style sheets

We've covered the basics of style sheets. For a more thorough treatment of the subject, you could do worse than look at this paper from the guys who invented the thing. This covers all the different tags and stuff that you can modify with style sheets.

The End

That's the end of this tutorial. We hope you found it useful. If you're still stuck and would like further help, check out our online Help Forums, where you can get assistance from members of Elated and other webmasters.

CSS Units

This tutorial lists all the units that can be used within style sheets. The units are used to specify things like distances and colours.

The units can be grouped into four types: **length** units, **percentage** units, **colour** units and **URL's**.

Length units

There are 8 length units in the CSS specification. Of these, 3 are **relative** units, and 5 are **absolute** units.

Relative length units

These units are relative to another length property. Use relative units when you want your pages to scale well across a range of output devices - for example, if you want your page to look good on the screen and when it's printed out.

Unit	Description	Example
em	The height of the element's font	letter-spacing: 0.1em

ex	The height of the letter **x** in the element's font	margin: 1ex
px	Pixels	font-size: 12px

Absolute length units

These units are "real-world" units. They are best used for print work, or other occasions when the type of output device is known.

Unit	Description	Example
in	Inches	line-height: 0.5in
cm	Centimetres	margin-top: 2cm
mm	Millimetres	letter-spacing: 1mm
pt	Points (1pt = 1/72in)	font-size: 14pt
pc	Picas (1pc = 12pt)	font-size: 2pc

Percentage units

Percentage units are always relative to another value, usually the element's font size.
For example:

```
/* 150% of the element's font size: */

  h4 { line-height: 150% }



/* 10% of the line width: */

  p { text-indent: 10% }
```

Colour units

Colours in CSS are specified using either a colour keyword (name), or an RGB value in one of several possible formats.

Colour keywords

Colours can be specified using names such as **black**, **white**, **red** etc. Although there is no official standard for these colours, Microsoft and Netscape browsers support hundreds of these colour names.
For example:

```
body { color: white; background: black }

h1 { color: red }
```

Colour RGB values

The best way to specify colours in CSS is using RGB values, which ensures that the exact colour will be used. Colours can be specified in any of the following ways:

Unit	Description	Example
#RRGGBB	Standard HTML red-green-blue 6-digit hex number	color: #FFFF00
#RGB	Short-hand red-green-blue 3-digit hex number: each digit is replicated to produce the final colour, e.g. #123 -> #112233	color: #0FF
rgb(R, G, B)	Decimal representation, each value being an integer between 0 and 255	color: rgb(255,192,255)
rgb(R%, G%, B%)	Percentage representation, each value being a percentage between 0.0% and 100.0%	color: rgb(100%,80%,50%)

URL's

A URL is specified using the following format:

```
url("http://www.url.com/images/box.gif")
```

The double quotes around the URL can be replaced with single quotes, or omitted altogether.
For example:

```
body { background-image:

url(http://www.url.com/images/background.jpg) }
```

The End

That's the end of this tutorial. We hope you found it useful. If you're still stuck and would like further help, check out our online Help Forums, where you can get assistance from members of Elated and other webmasters.

Controlling Fonts with CSS

In this tutorial we will look at how to control font properties using style sheets. This is a really useful feature of CSS because it means that you can avoid having all those **<font>** tags in your Web pages, and it allows you to easily control all your fonts simply by editing one style sheet file.

We'll look at the different **font properties** that can be used with CSS, and explain each property with the aid of some real-life examples. Each example is shown as it renders in your browser.

There are six properties that can be used to control fonts - **font-family**, **font-style**, **font-variant**, **font-weight**, **font-size** and **font**. Let's look at each of these in turn.

1. font-family

The **font-family** property is used to set the font face used for the text (e.g. **Times** or **Arial**). The allowed values are:

Value	Example
family-name	{ font-family: Times }
generic-family	{ font-family: serif }

You can specify one or more **family names** or **generic families** for the font. **Family names** include names such as **times**, **arial**, **gill** and **helvetica**.

Generic families include the following:

- **serif** (e.g. Times)
- **sans-serif** (e.g. Helvetica)
- **cursive** (e.g. Zapf-Chancery)
- **fantasy** (e.g. Western)
- **monospace** (e.g. Courier)

It's always a good idea to include at least one **generic family**, in case the browser can't find a specific font (e.g. not all computers will have **Zapf-Chancery** installed).

Examples:

```
p { font-family: "times new roman", serif }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { font-family: helvetica, arial, sans-serif }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

2. font-style

This property controls whether the font is slanted (italicised) or not. The options are:

Value	Example
normal	{ font-style: normal }
italic	{ font-style: italic }
oblique	{ font-style: oblique }

This allows you to set the font style to **normal** (upright), or **italic / oblique** (slanted). **italic** and **oblique** are usually the same thing on most browsers.

Examples:

```
p { font-style: oblique }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { font-style: normal }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

3. font-variant

Allows you to choose between normal and small-caps lettering. Small-caps are capital letters used in place of lowercase ones. The original capitals are shown as slightly bigger capitals.
The options are:

Value	Example

normal	{ font-variant: normal }
small-caps	{ font-variant: small-caps }

Examples:

```
p { font-variant: small-caps }
```

`TAKE SOME MORE TEA,' THE MARCH HARE SAID TO ALICE, VERY EARNESTLY.

```
p { font-variant: normal }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

4. font-weight

This controls whether the font is normal weight, bold or light. The options for **font-weight** values are:

Value	Example
normal	{ font-weight: normal }
bold	{ font-weight: bold }
bolder	{ font-weight: bolder }
lighter	{ font-weight: lighter }
100, 200, ... 900	{ font-weight: 700 }

bolder and **lighter** will make the font one step bolder or lighter than the default or inherited weight. The numbers **100 - 900** are a numerical representation for weight, where **100** is the lightest and **900** is the heaviest (boldest). Usually, **normal** is represented by **400**, and **bold** by **700**.

Examples:

```
p { font-weight: bold }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { font-weight: 200 }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

5. font-size

The **font-size** value specifies the size of the font used to display the text. There are four types of values that can be used for **font-size**:

Value	Example
absolute-size	{ font-size: large }

relative-size	{ font-size: smaller }
length	{ font-size: 12pt }
percentage	{ font-size: 150% }

absolute-size values are words specifying definite font sizes. Possible values are: **xx-small, x-small, small, medium, large, x-large** and **xx-large** . Although the exact font size represented by, say, **large**, may vary from browser to browser, you can be sure that **large** will always be bigger than **medium**, for example.
relative-size values are relative to the inherited or default font size. Possible values are: **larger** and **smaller**. For example, if the inherited font size (from a parent style) is **large**, a **relative-size** of **larger** will set the font size to **x-large**.
length values are specified using the CSS length units such as **em**, **px**, **cm** and **pt**. For a full description of these, see the tutorial CSS Units.
percentage values are relative to the parent element's font size. For example, if the inherited font size is **12pt** and a percentage value of **150%** is specified, the resulting font size will be **18pt**. See the tutorial CSS Units for details.

Examples:

```
p { font-size: 10pt }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { font-size: larger }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { font-size: 10px }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { font-size: 150% }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

6. font

The final property is **font**, which can be used as a shorthand for setting all the previous properties in one line, plus the **line-height** text property. The following values may be used. Note that the values should be used in the order shown; for example, **font-size** should come after **font-weight**.

Value	Example
font-style	{ font: italic }
font-variant	{

Controlling Text Appearance with CSS

In this tutorial we will look at how to control text appearance using style sheets. CSS gives you precise control over typography in your web pages, allowing you to set parameters such as the spacing between lines, words and even letters, and the alignment and indenting of text.

We'll look at the different **text properties** that can be used with CSS, and explain each property with some real-world examples. Each example is displayed as it would render in your browser.

There are eight properties that can be used to control text appearance - **word-spacing**, **letter-spacing**, **text-decoration**, **vertical-align**, **text-transform**, **text-align**, **text-indent** and **line-height**. Let's look at each of these properties in turn.

1. word-spacing

This property controls the amount of space added to the default spacing between each word. The allowable values are:

Value	Example
normal	{ word-spacing: normal }
length	{ word-spacing: 5mm }

normal will select the default word spacing.
length will add the specified value to the default word spacing. **length** values are specified using the CSS length units such as **em**, **px**, **cm** and **pt**. For a full description of these, see the tutorial CSS Units.

Examples:

```
p { word-spacing: 1em }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { word-spacing: normal }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

2. letter-spacing

This property is similar to **word-spacing**, but controls the spacing added between each individual letter. Possible values are:

Value	Example
normal	{ letter-spacing: normal }
length	{ letter-spacing: 0.1mm }

normal will select the default letter spacing.
length will add the specified value to the default letter spacing. **length** values are specified using the CSS length units such as **em**, **px**, **cm** and **pt**. For a full description of these, see the tutorial CSS Units.

Examples:

```
p { letter-spacing: 0.1em }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { letter-spacing: 0.1cm }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

3. text-decoration

The **text-decoration** property allows you to control decorations that can be added to the text, such as strike-throughs, underlining, and (heaven forbid!) blinking. Possible values are:

Value	Example
none	{ text-decoration: none }
underline	{ text-decoration: underline }
overline	{ text-decoration: overline }
line-through	{ text-decoration: line-through }
blink	{ text-decoration: blink }

Examples:

```
p { text-decoration: underline }
```

`TAKE SOME MORE TEA,' THE MARCH HARE SAID TO ALICE, VERY EARNESTLY.

```
p { text-decoration: line-through }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

4. vertical-align

This property controls the vertical placement of the text. It's similar to the **valign** attribute in HTML. The allowable values are:

Value	Example
baseline	{ vertical-align: baseline }
sub	{ vertical-align: sub }
super	{ vertical-align: super }
top	{ vertical-align: top }
text-top	{ vertical-align: text-top }
middle	{ vertical-align: middle }
bottom	{ vertical-align: bottom }
text-bottom	{ vertical-align: text-bottom }
percentage	{ vertical-align: 25% }

The following 6 values are relative to the parent element:

baseline aligns the baseline with the parent's baseline. This is the default.

middle aligns the vertical midpoint of the element (e.g. an image) with the baseline plus half the x-height of the parent. The **x-height** is the height of the letter 'x' in the context of this line of text. See the tutorial CSS Units for details.
sub displays the element as subscript text.
super displays the element as superscript text.
text-top aligns the top of the element with the top of the font used for the parent element.
text-bottom aligns the bottom of the element with the bottom of the font used for the parent element.

The next two values are relative to the line that the element is in, as opposed to the parent element:

top causes the top of the element to be aligned vertically with the tallest element on the line.
bottom causes the bottom of the element to be aligned vertically with the lowest element on the line.
The final value, **percentage**, specifies a percentage of the element's line height (see the **line-height** property below). It raises or lowers the baseline by a percentage of the line-height above the baseline of the parent. For example, a value of **50%** will raise the baseline to halfway between the parent's baseline and the baseline of the line above. A value of **-100%** will lower the baseline to the same height as the baseline of the line below.

Examples:

```
span { vertical-align: super }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

```
span { vertical-align: sub }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

```
span { vertical-align: 100% }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

5. text-transform

text-transform controls the case of the text. You can transform all the text into capitals or lowercase, or just capitalize the first letter of each word. The options are:

Value	Example
capitalize	{ text-transform: capitalize }
uppercase	{ text-transform: uppercase }
lowercase	{ text-transform: lowercase }
none	{ text-transform: none }

capitalize uppercases the first character of each word, while **uppercase** and **lowercase** transform the whole text into all upper-case or lower-case characters respectively. **none** removes all transformations from the text and displays it as-is.

Examples:

```
p { text-transform: uppercase }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

```
p { text-transform: capitalize }
```

`Take some more tea,' the March Hare said to Alice, very earnestly.

6. text-align

This property is similar to the **align** attribute in HTML. Options are:

Value	Example
left	{ text-align: left }
right	{ text-align: right }
center	{ text-align: center }
justify	{ text-align: justify }

left, **right** and **center** perform the same actions as their HTML counterparts. **justify** creates columns of text that are aligned along their left and right edges, like the text in a book.
Examples:

```
p { text-align: center }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

```
p { text-align: justify }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

7. text-indent

This property allows you to indent the first line of text in a paragraph. The options are:

Value	Example
length	{ text-indent: 3cm }
percentage	{ text-indent: 5% }

length is specified using the CSS length units such as **em**, **px**, **cm** and **pt**. For a full description of these, see the tutorial CSS Units.
percentage is a percentage of the parent element's width.

Examples:

```
p { text-indent: 5em }
```

THE CAT ONLY GRINNED WHEN IT SAW ALICE. IT LOOKED GOOD- NATURED, SHE THOUGHT: STILL IT HAD VERY LONG CLAWS AND A GREAT MANY TEETH, SO SHE FELT THAT IT OUGHT TO BE TREATED WITH RESPECT.

8. line-height

The final text property, **line-height**, controls the distance between lines of text. Possible values are:

Value	Example
normal	{ line-height: normal }
number	{ line-height: 1.5 }
length	{ line-height: 0.5cm }
percentage	{ line-height: 125% }

normal sets the line height to the default or inherited value, while specifying a **number** results in a line height of the default value multiplied by that number.
length is specified using the CSS length units such as **em**, **px**, **cm** and **pt**. For a full description of these, see the tutorial CSS Units.
percentage is a percentage of the element's font size.
Examples:

```
p { line-height: normal }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

```
p { line-height: 2 }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

```
p { line-height: 150% }
```

The Cat only grinned when it saw Alice. It looked good- natured, she thought: still it had VERY long claws and a great many teeth, so she felt that it ought to be treated with respect.

(we can't remember from which source we found this tutorial – it is definitely not made by us)