

A BEGINNER'S TUTORIAL

C S S

M A D E E A S Y

**AN
INTRODUCTION TO
CASCADING SHEET STYLES**

By E.Cierebiej

CSS Made Easy

An Introduction To Cascading Sheet Styles

© 2012 www.EasyHtmlCode.com

Redistribution of this ebook or its contents is prohibited

CSS Made Easy

An Intro To Cascading Sheet Styles

Introduction

CSS, as *cascading sheet styles* are commonly called, turn plain looking webpages into ones that are colorful and eye catching. It does stuff like change size, color and positions elements on webpages in a way that HTML alone cannot. Best of all, CSS can change the appearance and layout of all the webpages within a site at once from a single CSS file called an *external style sheet*.

Another cool thing about CSS is that it is really easy to learn, the only thing you need to know is a basic knowledge of HTML.

Where Does CSS Go?

There's three types of CSS:

embedded - this type of CSS goes between the `<head></head>` tags of each webpage:

```
<head>
<title>Learning CSS</title>

<style type="text/css">
p {font-size:12px;}
</style>
</head>
```

Inline - You are already familiar with this type of CSS from the *HTML Made Easy* lessons, it is inserted directly into HTML tags with the **style** attribute:

```
<p style="font-size:12px;"> some text...blah, blah</p>
```

External Style Sheet - With this type of CSS it is possible to change the appearance of an entire site from just one file called an **external style sheet**, and you are about to find out how to make one. Oh, relax, it's going to be easy! Really, it is so let us get started.

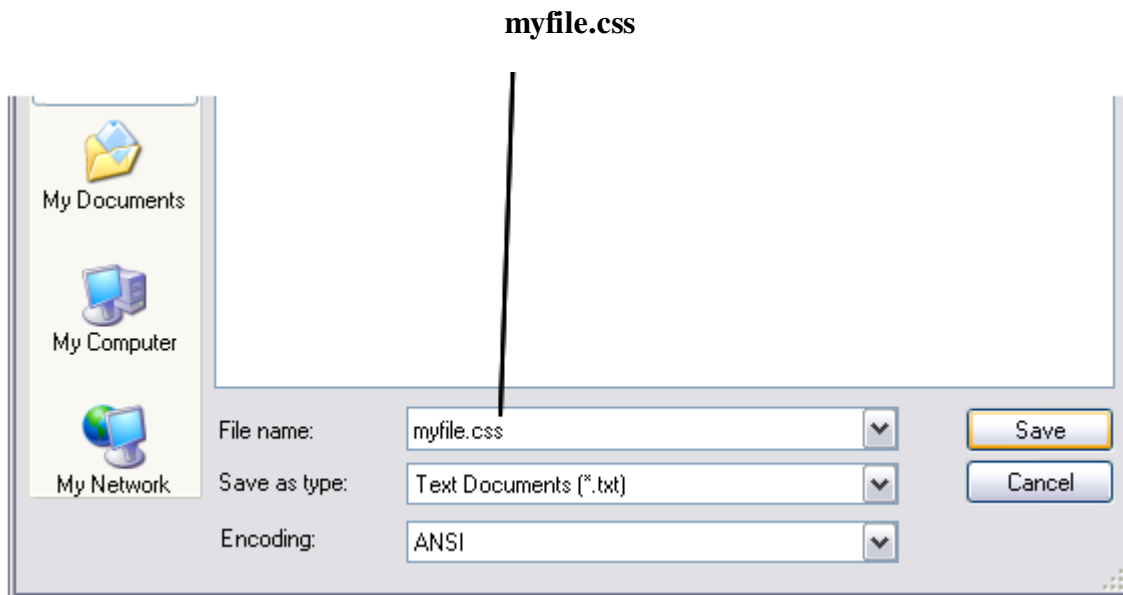
Making An External Style Sheet

What An External Style Sheet Is

An *external style sheet* is a single CSS file in which CSS code is written. This code will affect all web pages that are linked to the external style sheet. Don't worry it's not as complicated as it sounds, an external style sheet is very easy to make.

How to Make An External Style Sheet

1. Open Notepad (Simple Text for Mac)
2. Save the blank file by giving it any name you want and add the CSS extension to it like so:



The dot between **myfile** and **css** is what turns it into an external style sheet. To keep things simple, save the file in the same folder you keep your web pages.

Next the web pages are linked to the external style sheet with the following bit of code placed between the `<head></head>` tags of the pages' HTML source code:

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

Replace **myfile.css** with the name you gave your external style sheet and be sure to include the **css** extension. Whatever web pages are linked to the external style sheet will be affected by the CSS code contained in it. You are now ready to begin writing CSS!

CSS Code Structure

The external style sheet you made is where CSS code will be written, only CSS goes there and nothing else, save it each time you add some code and then refresh your web page to see the effect.

CSS code is made up of **selectors** and **properties**. A property is a command such as "hey make that background blue", and a selector determines which HTML tag will be affected by the property.

Here's an example of what CSS code looks like:

```
p { color:red; }
```

That bit of code instructed all **p** tags to display red text. In our example **p** is the *selector*, **color** is the *property*, and **red** is the property's *value*. Notice that the property and its value are enclosed in curly brackets, notice also there is a colon between the property and its value and that there is a semi colon at the end of the property's value. This is how CSS code is structured.

Selectors

There are several types of selectors, one type are *tag* selectors, these selectors are the name of HTML tags such as the **p** selector in the above example. Any HTML tag can have a corresponding selector:

```
h3 { text-align:center; }
```

That will center any text enclosed between the **<h3></h3>** tags.

Class Selectors

Another type of selector is the *class* selector. A class selector is made by giving it a name, for example:

```
.highlight { background:yellow; }
```

In the above example, **.highlight** is the name given to the selector, the name can be any made up word by you. Note there is a dot preceding the name, this dot is what turns the word into a class selector. An unlimited amount of class selectors can be made, just be sure to give each one a different name.

In order for a class selector to work, the **class** attribute along with the given name is added to any HTML tags that are to be affected by it such as in the example below:

This `word` will be affected by the class selector

Result:

This **word** will be affected by the class selector

Id Selectors

An *id* selector works in a similar way to class selectors. And as with class selectors, an id selector is also given a name by you, it could be any word. Here is what an id selector might look like:

```
#content { font-size:12px; }
```

The name given the selector is **content** and notice the number sign preceding the name (#), that's what makes it an id selector.

To get the id selector to work, the **id** attribute along with the given name needs to be inserted into the HTML tag that is to be affected by it:

```
<div id="content"> some text, etc....</div>
```

The result would be whatever the property commanded the id selector to do, in this example it would make size of text 12 pixels high. An unlimited number of id selectors can be made in the external style sheet but unlike class selectors, in which the same class selector can be used in numerous HTML tags within the same web page, each id selector should only be used once per web page.

Properties

Properties are CSS commands, and there are lots of them. They can be used with selectors or put directly into HTML tags with the use of the **style** attribute:

Property with a selector - `body { background:gray; }`

Property in HTML tag - `<body style="background:gray;">`

Properties in the **style** attribute override properties in selectors, so if the background in a body selector is set to gray but in the HTML **body** tag it is set to white, the web page will display a white background. Note Color values can either be the name of the color or its code, e.g. **#808080**

Multiple properties can be used with a selector as well as with the **style** attribute:

```
h1 {  
  color:red;  
  font-family:arial;  
  text-align:center;  
}
```

The result would be **h1** tag displaying:

Red Arial Text That's Centered

It doesn't matter how properties are organized as long as they are between curly brackets (or quotation marks when used with the **style** attribute), and each property is separated by a semi-colon at the end of its value.

Some properties will only have an effect on some HTML tags, for instance the **text-align** property will work on block tags but will have no effect on inline tags.

Some properties can contain more than one value:

```
h3 { border:1px solid blue; }
```

Note that when a property has more than one value, the values are separated by a space and only the last value has a semi colon. Result:

Blue solid border that's 1 pixel thick

Some properties have a **top**, **right**, **bottom** and **left** counterpart property:

```
h3 { border-top:2px dotted red; }
```

Result:

Red dotted top border that's 2 pixel thick

Pretty cool if I do say so. Let's see some more cool stuff CSS can do.

CSS Links

CSS has special selectors for redefining how links are displayed on web pages:

a:link - selector for links visitors have not yet clicked

a:visited - selector for links visitors have already clicked

a:hover - selector for links visitors move their mouse pointer over

Here is an example of how to use these selectors to remove the underline from links:

```
a:link { text-decoration:none; }  
a:visited { text-decoration:none; }  
a:hover { text-decoration:underline; }
```

Notice that the value for the **text-decoration** property in the **a:hover** selector is *underline*, this makes a nice mouseover effect, when the mouse pointer is moved over it the link will display an underline, but when the mouse pointer is moved off the link, the underline will disappear. Using different combinations of properties in the selectors can have some pretty cool mouseover effects.

Making different classes of link selectors allows you to choose which links will be affected by which class of link selectors. For example you may want sidebar links to have a different color from links within the main content of the web page.

To make a class, a name is given to each type of link selector:

```
a.sidebar:link { color:yellow; }  
a.sidebar:visited { color:yellow; }  
a.sidebar:hover { color:green; }
```

The name given to the set of selectors in this example is **sidebar** but it can be any word as long as it is the same for all the selectors within the class.

The next step is to insert the **class** attribute with the name into the anchor tags that are to be affected by that class of selectors:

```
<a href="file.html" class="sidebar">link here </a>
```

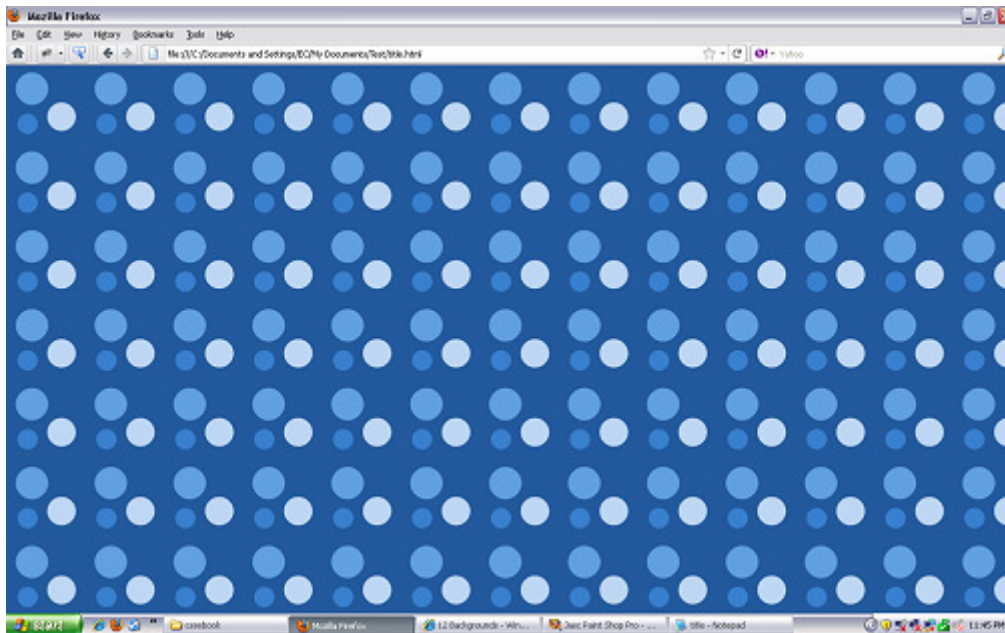
Any link that contains this code will be affected by the **"sidebar"** class of selectors. An unlimited amount of class selectors can be made for links, just be sure that the same name is given to each type of link selector within the same class.

CSS Backgrounds

The **background** property does more than just add color to a background, it can also place images as backgrounds within HTML elements. A single image such as this square graphic:



can be made to tile across and down a web page like this:



The code:

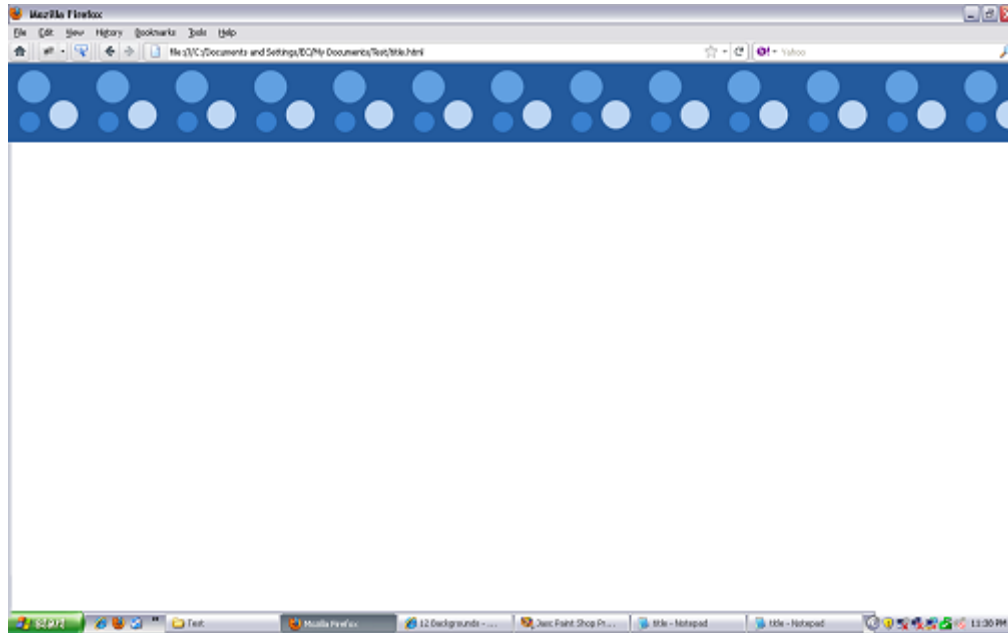
```
body { background:url(pic.gif); }
```

The **url** is telling the **background** property to look for an image with the file name **pic.gif**. Replace **pic.gif** with the file name of your graphic and be sure it is enclosed in the round brackets.

8 CSS Backgrounds

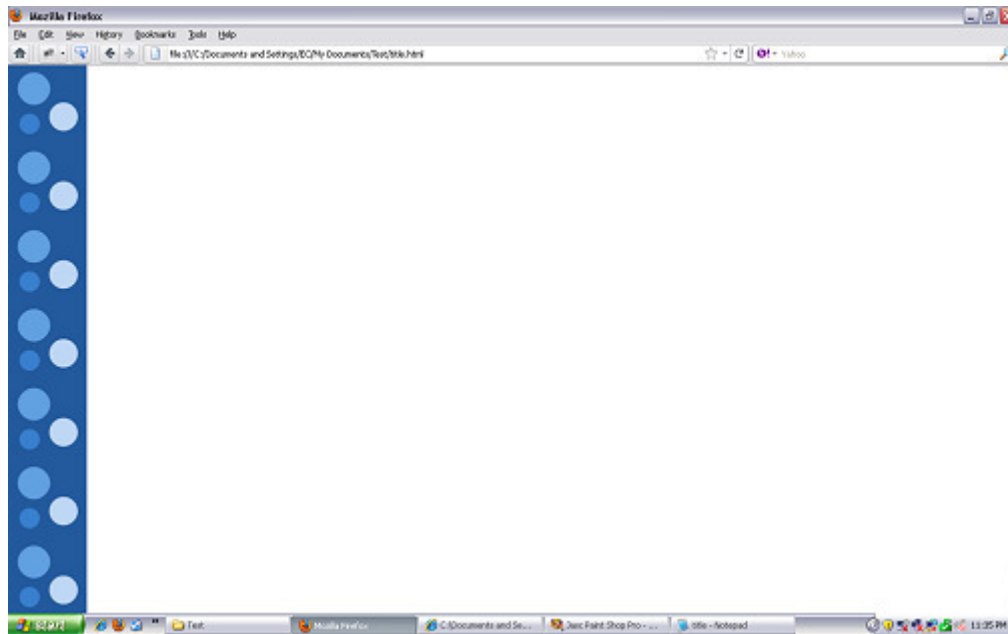
Adding a **repeat-x** or **repeat-y** value will cause a background graphic to tile either horizontally or vertically. Example for horizontal repetition:

```
body { background:url(pic.gif) repeat-x; }
```



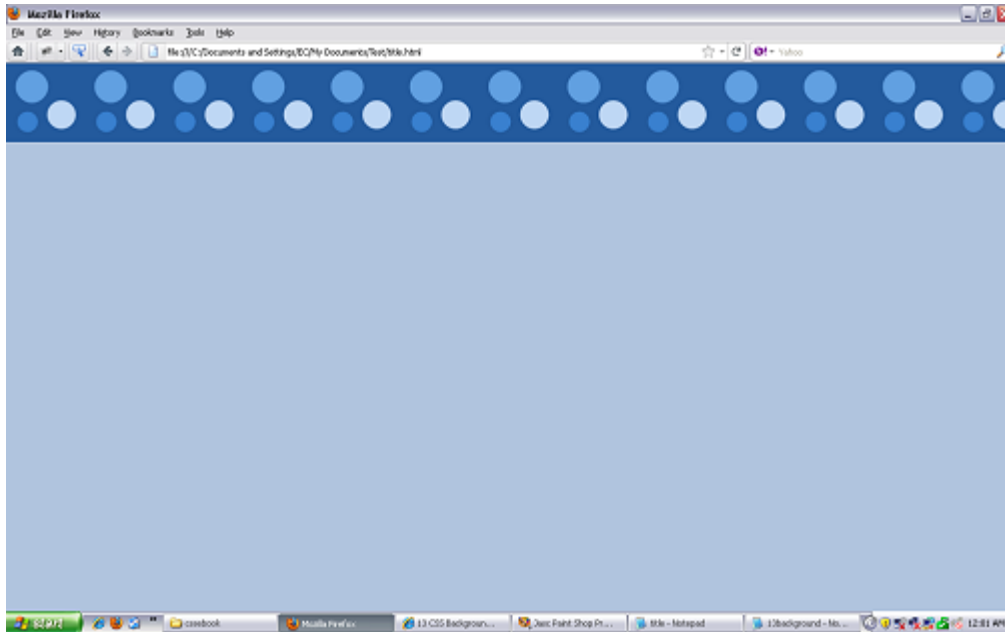
Now an example of vertical repetition:

```
body { background:url(pic.jpg) repeat-y; }
```



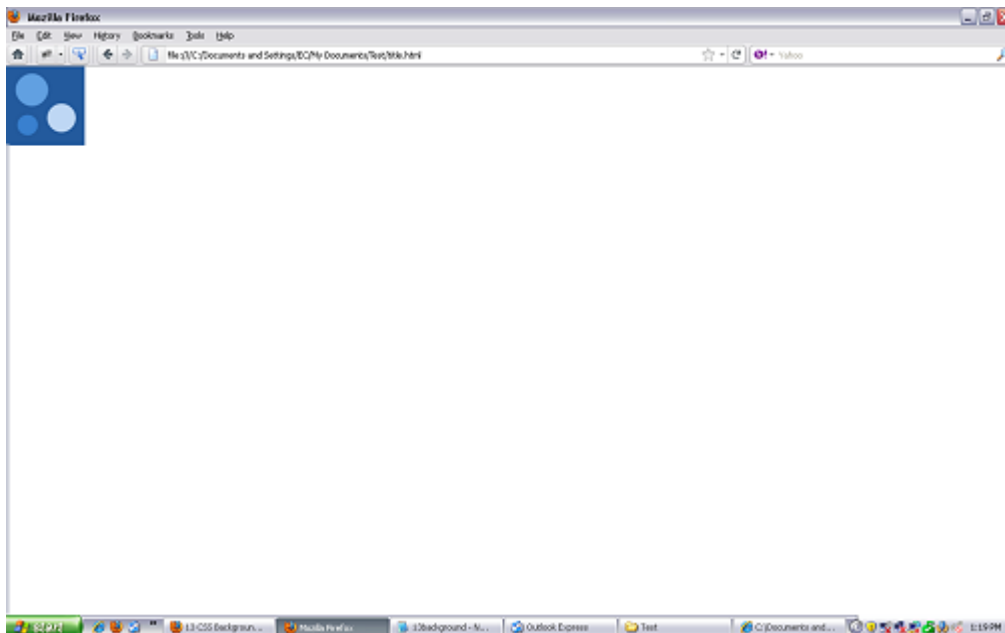
There can be both a color and an image background at the same time:

```
body { background:lightsteelblue url(pic.gif) repeat-x; }
```



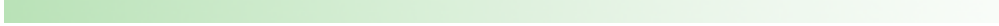
Adding a **no-repeat** value will prevent the image from repeating:

```
body { background:url(pic.gif) no-repeat; }
```



Learning CSS

This is the background graphic used:



and the CSS code:

```
h1 { background:url(hbackground.gif) repeat-y; }
```

Replace **hbackground.gif** with the name of your background graphic. Not too bad this CSS stuff, eh?

Aligning With CSS

CSS has a property that aligns elements side by side on webpages such as images and paragraphs:

```
.pic {float:left;}
```

```
.text {float:left;  
      width:100px;  
}
```

Notice a **width** property was added in the second class selector, this will set the width of the paragraph.

Code in HTML source:

```
  
<p class="text"> The text within this paragraph will line up beside the picture. Whole paragraphs  
can be lined up beside each other too, not just pictures!</p>
```

Result:



The text within this paragraph
will line up beside the picture.
Whole paragraphs can be lined
up beside each other too, not
just pictures!

Ugh! The text is too close to the picture! No problem, we will fix that by putting a **margin** property into either or both of the class selectors:

```
.pic {float:left;  
      margin-right:20px;  
}
```

```
.text {float:left;  
      width:100px;  
}
```

Result on next page:



The text within this paragraph will line up beside the picture. Whole paragraphs can be lined up beside each other too, not just pictures!

Now that is much better! The **float** property can align anything side by side on webpages, eliminating the need for tables. This makes it an indispensable tool in webpage layout using CSS.

Starting the next paragraph or picture below the "floated" elements is done with the **clear** property :

```
  
<p class="text"> The text within this paragraph will line up beside the picture. Whole paragraphs  
can be lined up beside each other too, not just pictures!</p>  
<p style="clear:left;"> Hey there! I'm starting below!</p>
```

Result:



The text within this paragraph will line up beside the picture. Whole paragraphs can be lined up beside each other too, not just pictures!

Hey there! I'm starting below!

If you wanted some more distance between the picture and bottom text a **margin-bottom** property would be added to the class selector for the picture.

Spacing With CSS

As with the **margin** property, the **padding** property also puts distance around elements on web pages, but before you see it in action let's see how something might look before a **padding** property is added so you will get a clearer understanding of what it does:

```
h3 {  
  background:yellow;  
  width:350px;  
}
```

The **width** property will determine the length of the background. Result:

Sample Text

Now the **padding** property is added to the selector:

```
h3 {  
  background:yellow;  
  width:350px;  
  padding:10px;  
}
```

Result:

Sample Text

And as with the **margin** and **border** properties, the **padding** property can also be designated **top**, **right**, **bottom** and **left**:

```
h3 {  
  background:yellow;  
  width:350px;  
  padding-left:30px;  
}
```

See the result on the next page:

Sample Text

The **padding** property adds to the total width or height of an element making the above example now 380 pixels in width. To get the desired width simply adjust the **width** property to factor in the amount of padding added, in this case the **width** property would be adjusted to 320px for a total width of 350 pixels to match the previous examples.

The **padding** property differs from the **margin** property in that it puts distance within an HTML element while the **margin** property puts distance outside an HTML element as shown in the examples below:

```
h3 {  
  background:#eeeeee;  
  width:350px;  
  padding-bottom:30px;  
}
```

Result with two **h3** elements:

Sample Text

Sample Text

As you can see, the space created is inside the background area of the **h3** elements. But when a **margin** property is used:

```
h3 {  
  background:yellow;  
  width:350px;  
  margin-bottom:30px;  
}
```

space is created outside the background area of the **h3** elements:

Sample Text

Sample Text

And that folks is the difference between a **padding** property and a **margin** property.

Display Property

CSS can make block tags behave as inline tags, and inline tags as block tags with the use of the **display** property:

h1 {display:inline;}

Code in HTML source:

<h1>Heading Tag**</h1>** Some more text

Result:

Heading Tag Some more text

As you see, even though the **h1** is a block tag, it didn't force the surrounding text to start a new line.

Here is an example of an inline tag displaying as a block tag:

.break {display:block;}

Code in HTML source:

Some text in this sentence**<em class="break">**will start **** on a new line

Result:

Some text in this sentence
will start
on a new line.

The text outside the **em** tag is not on the same line even though the **em** tag is an inline tag. Is that cool or what?

Formatting Text

There are quite a few text altering properties, here are some of the more commonly used. To change text color use either the name of the color or the color code in the **color** property:

```
.example { color:#800000; }
```

To change font type:

```
p { font-family:courier; } = Font in courier style
```

Take note that the type of font the viewer sees depends on the fonts installed on the viewer's computer. If the person does not have the font your website uses they will see the default setting of their browser.

Change text size:

```
em { font-size:20px; } = text 20 pixels high
```

For bold text:

```
.example { font-weight:bold; }
```

For italic text:

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

To indent the first sentence in a paragraph:

```
p { text-indent:15px; }
```

That would indent the first sentence by 15 pixels.

You may remember that the **text-align** property centers text with the *center* value, it can also put text to the right:

```
.mytext { text-align:right; }
```

The **text-align** property only works in block tags such as **div**, **p** and any of the heading tags.

18 Formatting Text

Earlier in the tutorial you were shown how the **text-decoration** property removes underlines from links with the *none* value, the **text-decoration** property has other values that can be used with regular text:

em { text-decoration:overline; } = *This is text with an overline value*

em { text-decoration:line-through; } = ~~*This is text with a line-through value*~~

How about an *overline* and *underline* value at the same time:

em { text-decoration:overline underline; } = *That's cool but don't get carried away*

Another cool thing that can be done with CSS is set the spacing between letters:

.mytext { letter-spacing:8px; }

Code in HTML source:

<div class="mytext">Hi There!</div>

Result:

H i T h e r e !

CSS can set the distance between lines of sentences too:

**<div style="line-height:10px;">Look at what CSS can do.
 It sets distance between lines.</div>**

Result:

Look at what CSS can do.
It sets distance between lines.

Is that cool or what! The larger the number the more distance there will be between the two lines.

CSS Lists

Both ordered and unordered lists can be styled with the **list-style** property. For instance change the type of bullets an unordered list displays:

ul {list-style:square;}

Code in HTML source:

```
<ul>
<li>Terrier</li>
<li>Beagle</li>
<li>Chihuahua</li>
</ul>
```

Result:

- Terrier
- Beagle
- Chihuahua

Change the value to **circle** for circle bullets:

ul {list-style:circle;}




Result:

- Terrier
- Beagle
- Chihuahua

Even images can be used as bullets:

ul {list-style:url(pwprint.gif);}

Result:

-  Terrier
-  Beagle
-  Chihuahua

Replace **pwprint.gif** with the name of your graphic.

With ordered lists, instead of numbers, Roman numerals or letters of the alphabet can be used to display lists:

ol {list-style:lower-roman;}

Result:

- i. Terrier
- ii. Beagle
- iii. Chihuahua

For upper case Roman numerals the **upper-roman** value is used:

ol {list-style:upper-roman;}

Result:

- I. Terrier
- II. Beagle
- III. Chihuahua

Upper case letters of the alphabet use the **upper-alpha** value:

ol {list-style:upper-alpha;}

Result:

- A. Terrier
- B. Beagle
- C. Chihuahua

Lower case letters use the **lower-alpha** value:

ol {list-style:lower-alpha;}

Result:

- a. Terrier
- b. Beagle
- c. Chihuahua

To remove bullets from both ordered and unordered lists use the **none** value:

ul {list-style:none;}

Another thing the **list-style** property can do is set the indentation of list items. By default text starting on a second line within a list item lines up with the text above it as in this example:

- The text in this list item
is on two lines
- Beagle
- Chihuahua

By giving the **list-style** property an **inside** value, the indentation of the second line of text within a list item will start under the bullet.

CSS code:

```
ul { list-style:inside; }
```

Result:

- The text in this list item
is on two lines
- Beagle
- Chihuahua

And in case you are wondering, yes this value can be used together with a bullet style value, just be sure the values are separated by a space:

```
ul { list-style:square inside; }
```

Result:

- The text in this list item
is on two lines
- Beagle
- Chihuahua

And that brings us to the end of this ebook, an expanded version of CSS Made Easy in printed book format is available at www.cafepress.com/easycss In it you will learn how to use CSS in the layout construction of web pages, and as a bonus for purchasing, the book includes a download link to three CSS based web page templates you can use to set up websites or for practice.