

Introduction to Cascading Style Sheets

Welcome to the CSS workshop! Before you begin this workshop you should have some basic understanding of HTML and building web pages. What is CSS anyway and what will it help me do? Over the course of this workshop you will learn the foundations of CSS and be able to create your own style sheet.

Objectives

- Understand what a style sheet is and how it actually styles a web page
- Know how to create a style sheet and link an html document to the style sheet
- Understand the basic building blocks of any style sheet: rules, selectors, properties and values
- Understand how to control the basic appearance of text
- Demonstrate how to customize the appearance of links
- Define background properties

Prerequisites:

You will need a basic understanding of HTML. You may want to take the HTML Basics workshop before you begin this workshop. You will need a text editor, such as Notepad and an Internet browser, such as Internet Explorer or Netscape.

Q: What is Notepad and where do I get it?

A: Notepad is the default Windows text editor. On most Windows systems, click your Start button and choose Programs then Accessories. It should be a little blue notebook.

Mac Users: SimpleText is the default text editor on the Mac. In OSX use TextEdit and change the following preferences: Select (in the preferences window) Plain text instead of Rich text and then select Ignore rich text commands in HTML files. This is very important because if you don't do this HTML codes probably won't work.

One thing you should avoid using is a word processor (like Microsoft Word) for authoring your HTML documents.

At the end of this workshop, you will create your own HTML page.

Introduction

What is CSS?

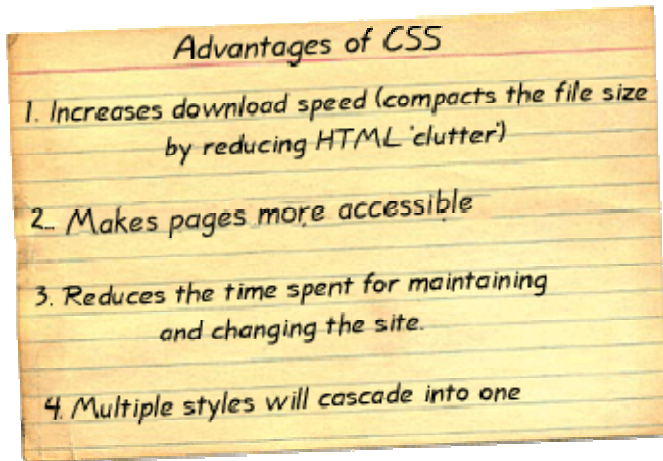
CSS is an abbreviation for Cascading Style Sheets. CSS works with HTML and other Markup Languages (such as XHTML and XML) to control the way the content is presented. Cascading Style Sheets is a means to separate the appearance of a webpage from the content of a webpage. CSS is a recommendation of the World Wide Web Consortium (the W3C). The W3C is a consortium of web stakeholders: universities, companies such as Microsoft, Netscape and Macromedia, and experts in many web related fields. The presentation is specified by styles, which are presented in a style sheet. If you're familiar with word processing programs like Microsoft Word, you have probably played around at least a little bit with styles. For example, when you want to make the headline text of your document big and bold, the hard way to do it would be to select the text, select a font face and weight, and select the color. The easier way to do it (presuming your document has more than one headline) is to create a "rule", or style, for all the headlines in your document. Then all you have to do is to make one rule, and keep on applying that to all your headers.

CSS in its most basic form works exactly like this. Instead of using tags over and over again to control little sections of your page, you can establish some rules to apply globally, to a single page or all the pages on your site. CSS is a great time saver.

What's the "Cascade" part of CSS?

The cascade part of CSS means that more than one stylesheet can be attached to a document, and all of them can influence the presentation. For example, a designer can have a global stylesheet for the whole site, but a local one for say, controlling the link color and background of a specific page. Or, a user can use her own stylesheet if she has problems seeing the page, or if she just prefers a certain look. We'll talk about cascading and hierarchy of styles in a later lesson

Why Use CSS?



Increases Download Speed

By taking the appearance markup out of the html file you decrease the html file size and significantly decrease the download time of that file.

Makes Pages More Accessible

Your pages become more accessible. By separating the styling (CSS) from the content and structure (HTML), you are well on your way to satisfying Accessibility requirements. This is an important thing to consider, if you are creating sites that might be used by the visually impaired.

Reduces Time

It is much easier to update pages. It is much faster to update a page that uses styles over using tags and the like. With CSS, you can decide how headings should appear, and enter that information once. Every heading in every page that is linked to this style sheet now has that appearance. Want to make every heading of level 3 obviously different from those of level 2? Edit the style sheet, and every such heading now has the altered appearance. Consider how much time you will save. With cascading style sheets, whole organizations can share a small number of style sheets, ensuring consistency across the site with no need for constant updating and editing to accommodate changes.

Multiple Style Sheets Cascade Into One

Style Sheets allow style information to be specified in many ways. Styles can be specified inside a single HTML element, inside the <head> element of an HTML page, or in an external CSS file. Even multiple external Style Sheets can be referenced inside a single HTML document.

How Do Style Sheets Work?

Style sheets are just text files, or text embedded in the head of an HTML document, that help separate content from appearance. The content of a page goes into an HTML file and the appearance goes into a style sheet. But how does all this end up as a web page in your browser? Think of a style sheet as a set of instructions, suggesting to a web browser how to draw a page. The style sheet suggests how the browser should display the page based on rules you define in the style sheet.

Let's Take A Look

To get an idea of what CSS can visit the website below. The examples on this site all use the same html document. The only difference between the examples is the style sheet used. Click on the examples listed on the right hand side of the page.

CSS Zen Garden - <http://www.csszengarden.com/>

Anatomy of CSS

The three types of HTML elements

An important concept to understand is that there are basically three types of HTML tags: ones that indicate a block-level element, ones that indicate an inline element, and replaced tags.

Think of block-level elements as boxes that have a line break before and after. The `<p>` tag is a good example of a box-level element: as you know, there is always a line break and space before and after each paragraph. Block-level elements are the ones you will spend most of your time applying style rules to.

Inline elements on the other hand, don't have line breaks before and after. Inline elements are used, well, inline: an example would be the `` tag that bolds a section of text.

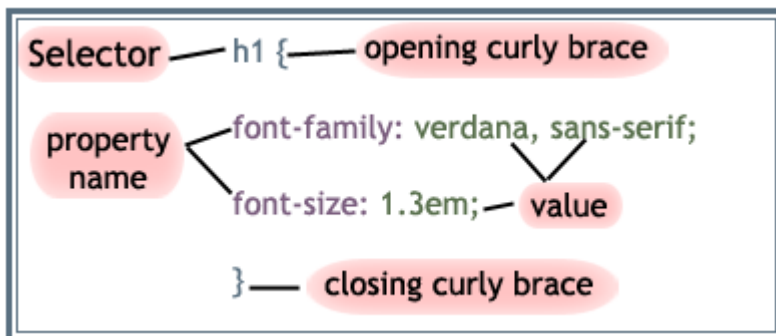
The third kind of HTML tag is called a replaced tag. I'm not sure why it's called "replaced" but what it means is simply that these elements have a set width and height. The most used replaced tag is the `` tag, which most often than not, you must specify a height and width for it.

The Parts of A Style Sheet

A style sheet consists of one or more rules that describe how document elements should be displayed. A rule in CSS has two parts: the selector and the declaration. The declaration also has two parts, the property and the value. Let's take a look at a rule for a heading 1 style:

```
h1 { font-family: verdana, "sans serif"; font-size: 1.3em }
```

This expression is a rule that says every h1 tag will be verdana or other sans-serif font and the font-size will be 1.3em. Let's take a look at the different parts of this rule.



```
Selector {  
property1: some value;  
property2: somevalue;  
}
```

The declaration contains the property and value for the selector. The property is the attribute you wish to change and each property can take a value. The property and value are separated by a colon and surrounded by curly braces:

```
body { background-color: black}
```

If the value of a property is more than one word, put quotes around that value:

```
body { font-family: "sans serif"; }
```

If you wish to specify more than one property, you must use a semi-colon to separate each property. This rule defines a paragraph that will have blue text that is centered.

```
p { text-align: center; color: blue }
```

You can group selectors. Separate each selector with a comma. The example below groups headers 1, 2, and 3 and makes them all yellow.

```
h1, h2, h3 { color: yellow}
```

Let's see how we can translate a typical tag to a CSS style. Here is the tag first:

```
<font size="3" color="green" face="verdana, arial, sans-serif">Some text</font>
```

This is the same thing (more or less) as a CSS style:

```
selector {  
font-family: verdana, arial, sans-serif;  
font-size: 1em;  
color: green;  
}
```

The selector in this case can be any tag that is on your page.

Style Sheet Syntax

Though easy to use, CSS is a very specific standard. This means that if you don't get the grammar right the browser won't know what to do. Either your style sheet will not work at all, or it will work in unexpected ways. Above you've seen what the parts of a style sheet are, and an example of how they are put together. Referring to the examples above, here are some simple rules to follow to get your style sheet right every time.

1. Every statement must have a selector and a declaration. The declaration comes immediately after the selector and is contained in a pair of curly braces.
2. The declaration is one or more properties separated by semicolons.
3. Each property has a property name followed by a colon and then the value for that property. There are many different types of values, but any given property can only take certain values as set down in the specification.
4. Sometimes a property can take a number of values, as in the font-family. The values in the list should be separated by a comma and a space.
5. Sometimes a value will have a unit as well as the actual value, as in the 1.3em. You must not put a space between the value and its unit.
6. As with HTML, white space can be used to make your style sheet easier to read and write.

Attaching Styles

There are three ways that styles can be associated with an HTML document. First, styles can be placed inline in a document. Second, a style sheet can be embedded in the head of an HTML document. The third way of associating web pages with style sheets is to place a link in the head of the HTML file to an external style sheet. With this link, when the browser begins reading the page, it sees the style sheet link, and downloads the style sheet, then uses the style sheet to draw the page.

Inline styles

Style rules can be specified right in the HTML tag, with **style="style rules"**. Here is an example:

```
<p style="font-family: trebuchet, times, serif; font-size: 20px; font-weight: 600; color: #7C3030; background-color: #B5D6A9"> The style rules above look like this! </p>
```

The style rules above look like this!

Specifying inline styles is useful when you only want to apply those styles to a specific element, as is the case above. For example, if you wanted to really emphasize a piece of text in your sentence, you could specify a local style of "color: red" to a element so that it looks like this **it looks like this**.

Embedding Style Sheets

Style sheets can be embedded into the <head> element of HTML documents. Quite simply, the style sheet itself is placed inside a <style> tag like this:

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

You would embed a style sheet within an HTML document when you want a single document to have a unique style. The head of the HTML document would look something like this:

```
<head>
<style type="text/css">
hr {color: brown}
p {font-family: arial, verdana, "sans-serif";}
body {background-color: #ffffff}
</style>
</head>
```

The browser will now read the style definitions, and format the document according to it. It's easiest for CSS beginners to start by putting CSS rules in the <head> section of the web page. This way, you can make changes, preview the page in the browser, and see immediate results. Within the <head> </head> section of your HTML page, put these tags:

```
<style type="text/css">
<!--
Your CSS rules go here!
-->
</style>
```

What are the <!-- --> tags for? That's to hide your CSS from older, non-CSS-capable browsers. If you want to add comments within your style section, put them between /* */ like this:

```
<style type="text/css">
<!-- Body {font-family: arial, sans-serif;} /* this is a css comment */ -->
</style>
```

Linking to External Style Sheets

An external style sheet is just a text file, containing **property: value** style rules. The only difference between the contents of an external style sheet and having your style rules in the document itself is that you do **not** include the <style> </style> tags, or any other HTML tags. To link a HTML document to a style sheet, you place a link to the style sheet in the head of the document. Linking to an external style sheet is ideal when applying styles to many pages. With an external style sheet, you can change the look of an entire web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

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

The browser will read the style definitions from the file mystyle.css, and format the document according to it. This can be either a relative or absolute url. The relative url is relative to the HTML document, unlike url's within style sheets which are relative to the style sheet. An external style sheet can be written in any text editor. The file should not contain any HTML tags. Your style sheet should be saved with a .css extension.

A simple css file looks like this:

```
body {  
font-family: verdana, arial, "sans serif";  
color: #000000;  
background: white;  
}
```

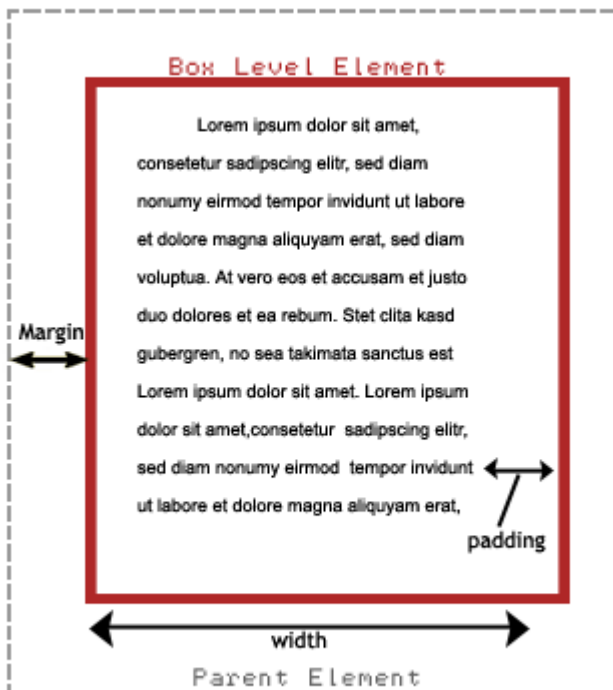
```
p {  
text-align: center;  
color: black;  
font-family: arial;  
}
```

Placing the properties on separate lines makes the style sheet easier to read. Some rules for external style sheets:

- They must be plain text files, with no HTML tags in them.
- They can have any filename you choose. Most CSS style sheets are called something.css, but that's just a naming convention.
- You can have it at any location on your site, but it's usually most convenient either to have it in your root directory or in a folder reserved for stylesheets, or for various kinds of external files you link to or include in your pages

Margins and Padding

Margin and padding are the two most commonly used properties for spacing-out elements. A margin is the space outside of the element, whereas padding is the space inside the element. While most style rules can be applied to most block level and inline HTML elements, some can only be applied to block-level elements: those are border, margin and padding. Plus, in most cases you'd only apply the text positioning styles to box level elements. Take a look at the Box Model illustration below, to see where each property is applied:



The Box Model

Do you see the difference between margin and padding? It's easier when you have a visible border, but essentially, margin applies to the space between the outside of the box and the element that

contains it (the parent element), while padding applies to the space between the border of the box and whatever is contained within it.

Note: if there is no specific containing or parent element, the Big Parent of the page is the <body> tag.

The paragraph below has this style, with a background-color and border specified so you can see what's going on:

```
This paragraph has the following margin:  
margin-left: 10px; margin-right: 20px;
```

As you see, the margin has changed, but the text inside is butting right against the borders of the box. To put some space there, add 10px of padding:

```
This paragraph has the following margin:  
margin-left: 10px; margin-right: 20px; padding: 10px;
```

To make it really stand out on the page, you can apply a border to the paragraph "box". Here is the complete style rule for the following element:

The following paragraph has these styles applied:

```
p {  
margin-left: 10px;  
margin-right: 20px;  
padding: 10px;  
background-color: #FFCC99;  
border: 5px dotted gray;  
}
```

CSS Styles

Text and Font Styles

Now that you are getting the hang of CSS rules, let's look at applying text, font, border and list styles. To apply styles, you redefine HTML elements like <p>, <h1>, <h2> and others. An example of redefining the <p> tag:

```
p {  
font-family: verdana, arial, helvetica, sans-serif;  
font-size: 12px;  
color: #000000;  
}
```

Font and text appearance are the things we most want to control with CSS. Font selector rules mostly apply to things we used to control with the tag, while text selector rules cover other things that influence the way a piece of text is presented.

Background Styles

The background property will let you set the background color, set the background as an image, repeat a background image and set a background image position on a page. Visit this sample page - http://profdevtrain.austincc.edu/css/sample_background.htm to see the background property at work.

For a list of styles that can be applied, visit the Resources page.

Border and List Styles

Border Styles

Borders can be applied to most HTML elements within the body. The Border properties allow you to specify the style, color, and width of an element's border. In HTML tables are usually used to create borders around text, but with the CSS Border properties you can create borders that can be applied to any element.

You can apply a border style to any of the four sides of an element by using top, bottom, right or left designations in your selector. The following border styles created the border around this example box:

```
p {
border-top-color: red;
border-top-style: solid;
border-bottom-color: navy;
border-bottom-style: dashed;
border-bottom-size: 5px;
}
```

List Styles

The List properties allow you to change between different list-item markers, set an image as a list-item marker, and set where to place a list-item marker. Visit this sample page to see the background property at work.

Lengths and Percentages

There are many property-specific units for values used in CSS, but there are some general units that are used in a number of properties and it is worth familiarizing yourself with these before continuing.

- **em** (such as font-size: 2em) is an element approximately equal to the height of a character.
- **px** (such as font-size: 12px) is the unit for pixels.
- **pt** (such as font-size: 12pt) is the unit for points.
- **%** (such as font-size: 80%) is the unit for... wait for it... percentages.
- Other units include **pc** (picas), **cm** (centimeters), **mm** (millimeters) and **in** (inches).

When a value is zero, you do not need to state a unit. For example, if you wanted to specify no border, it would be border: 0.

Note: A web page is not a static, absolute medium. It is meant to be flexible. Because of this, it is generally accepted that 'em' or '%' are the best units to use for font-sizes, rather than 'px', which leads to non-re-sizable text in most browsers, and should be used sparingly, for border sizes for example.

Colors

CSS brings 16,777,216 colors to your disposal. They can take the form of a name, an 'rgb' (red/green/blue) value or a hex code. Defining your font-color as red is the same as rgb(255,0,0), which is the same as rgb(100%,0%,0%), which is the same as #ff0000, which is the same as #f00.

There are 16 valid predefined color names. These are aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. Transparent is also a valid value.

The three values in the rgb value are from 0 to 255, 0 being the lowest level (for example no red), 255 being the highest level (for example full red). These values can also be a percentage.

Hexadecimal (or hex) is a base-16 number system. We are generally used to the decimal number system (base-10, from 0 to 9), but hexadecimal has 16 digits, from 0 to f.

The hex number is defined by a hash character (#) and can be three or six digits in length. Basically, the three-digit version is a compressed version of the six-digit (#f00 becomes #ff0000, #c96 becomes #cc9966 etc.). The three-digit version is easier to decipher (the first digit, like the first

value in rgb, is red, the second green and the third blue) but the six-digit version gives you more control over the exact color.

To assist you with hexadecimal colors and color schemes, visit these two sites:

<http://www.december.com/html/spec/color.html>

<http://wellstyled.com/tools/colorscheme2/index-en.html>

ID's, Classes, and Pseudo-classes

You can only have one p tag per paragraph. Or only one headline tag per paragraph. This can make things difficult if you are trying to highlight just a few words in selected paragraphs. You might also want to define an entire area of a page with a style, so that borders go around an area, rather than a paragraph. That's what ID's and class styles can do. You can add them to any tag, including HTML tags, like the p or h1, or table tags like td or tr, or just all by themselves in a span tag.

Classes

The class attribute lets you apply style rules to a group of elements. Class stands for Classify. A class selector is a name preceded by a period (.) Here is an example:

```
p.class {color: red}
```

This is the code used to turn this text red.

The html is written:

```
<p class="class"> This is the code used to turn this text red.</p>
```

So for every paragraph that you want in red text you would use the class defined above.

Another way to specify a class is simply like this:

```
.someclass {color: red;}
```

This omits the element in front, so the class "someclass" can be applied to **any element on the page**.

When to use which? It all depends the situation. When the class rules are really specific to an HTML element, then you can add the class name to the element. If you want a general class selector, then use one on its own.

ID's

An ID attribute identifies one element on a page. An ID selector is a name preceded by a hash character (#). Here is an example:

```
#class {  
color: navy;  
border: 1px black solid;  
}
```

This html is written:

```
<p id="class">#class {  
color: navy;  
border: 1px black solid;  
}</p>
```

As this example is an ID, you would use this only once. If you wanted to have more than one paragraph with navy text and a solid black border you would make this a class.

ID's, Classes, and Pseudo-classes

Pseudo-classes

Pseudo-classes are used in CSS to add different effects to some selectors, or to a part of some selectors. Pseudo-classes are attached to selectors to specify a state or relation to the selector. They

take the form of selector:pseudo class { property: value; }, simply with a colon in between the selector and the pseudo class. The most common pseudo-class is the link style.

Links can have four different states.

- link - this is the normal state
- visited - when a browser has visited that destination recently
- hover - while the cursor is over the link
- active - while the link is being clicked

We can create rules that apply to links in any of these states. The definition order **must** be a:link, a:visited, a:hover and a:active in order to be effective. The following links are created with these style definitions :

```
a:link { color: #336699; text-decoration: none; }
a:visited { color: #99CCFF; text-decoration: none; }
a:hover { color: #660000; text-decoration: none; }
```

What if you want the links in a certain section of your page to be different, like for menus? Pseudo-classes can be combined with other classes.

```
a.sample1:link { color: #336699; text-decoration: none; padding: 1px;}
a.sample1:visited { color: #660000; text-decoration: none; padding: 1px; }
a.sample1:hover {color: #660000; text-decoration: none; padding: 1px; font-size: 110%; }
a.sample1:active {font-size: 120%;}
```

The html is written:

```
<p><a class="sample1" href="#">This is an example of a pseudo-class and CSS class
combined.</a></p>
```

Cascade and inheritance

Cascade Order

So now you know how to set up stylesheets externally or on the page. But with so many style rules, how is it decided what takes precedence? What if there two identical selectors with different style rules? This is where the Cascade part of CSS (Cascading Style Sheets) comes into play.

Order of preference: different stylesheets

A document can have several stylesheets involved with it. Besides the ones you create, all browsers have a built-in stylesheets (where you set the default font, body and link colors). Unless the user changes the default settings, here's how the stylesheets are ranked:

1. Author (Designer) stylesheets take top priority
2. Then user stylesheets
3. Finally, browser default styles

Author stylesheets

You as the author (designer) of the web pages can create several stylesheets. Here's how they are ranked.

1. Inline styles and embedded styles (styles on the HTML page itself). Inline and embedded styles are called **Persistent** styles, and they usually have the top weight; in other words, persistent styles override any styles in external stylesheets.
2. The main or **Preferred** external stylesheet, which is linked with the attribute **rel=** in the **LINK** tag, follows in the hierarchy.
3. **Alternate** stylesheets are not used until the user selects them explicitly. If an alternate stylesheet is selected, the preferred one is turned off, but the persistent styles remain.

When there's only one stylesheet

When there is only one stylesheet, here is how style rules are ranked:

1. It's possible to push a style rule up to the "front of the line", by giving it an **!important** label, **regardless of which stylesheet it's in**. For instance, if there are two conflicting rules for the same selector:

```
H1 {font-family: verdana;}  
H1 {font-family: times !important;}
```

Then the second rule will prevail and the H1 element will be in Times.

2. The more specific a rule is, the more weight it has. This means for example that style rules applied to an ID selector or a class selector have more weight than style rules applied to generic type selectors.
3. Finally, when all other rules are exhausted, the style rules that are listed **last** take higher precedence. (That's why **persistent** styles (the ones actually on the HTML page) take higher precedence over linked external stylesheets, because they get processed last.)

Inheritance

Inheritance describes how style rules are passed on from the **parent element** to the **child element**. What parents and children? Well, any element that is **nested** within another element on an HTML page is a child, and the surrounding element is the parent

Take a look at a typical HTML web page structure:

```
<body> <p> .....  
<em> .....</em> ..... <b> .....</b>  
..... <span> .....</span> </p>  
</body>
```

The <p> element is a child of the <body> element; the and elements are children of the <p>. The big parent here is the <body>.

This means that in a browser, any style rules that are applied to a parent element gets inherited by the child elements **unless the child elements have style rules of their own**. For instance, if you have a style rule for the BODY like so:

```
body {font-family: "Comic Sans MS"}
```

Then all of the children of the BODY should also display with the Comic Sans MS font, unless specified otherwise. Note that this **parent and child** concept is very important.

A child element inherits the color and background-color values from the parent, but if there is a conflict between rules, and an incompatible rule wins out for the parent element, there could be problems. Let's say a the BODY style rule that "wins out" had this style:

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

However, the p style rule that "won out" was this:

```
p {color: black;}
```

With no background-color value specified, the paragraphs on that page are going to be pretty unreadable! In any case, the more complicated your CSS styles get, the more you have to test everything to see that things are working together. But not to worry - practice does make things easier.

What Now?

Now you're ready to venture out on your own. It's time to put things together and start creating your own CSS style sheets. Take the online quiz and then complete the following assignments:

Take the quiz! - <http://www.austincc.edu/ITdocs/css/quiz/popquiz.htm>

Assignment

- Make a list of the HTML tags you use all the time. Group them into **block-level**, **inline** and **replaced** tags.
- Download the practice page and turn it into a simple HTML page, with **no table tags**. If you can, stick to <P>, <H1>, <H2>, etc. tags. Then, put the tags that indicate that CSS rules will go there in the <HEAD> section.

You can use a text editor, Dreamweaver or other web creation software to edit the file. Visit the Resources pages for a detailed list of CSS properties. Post this document to your web site and email the url to the instructor at lniles@austincc.edu, or e-mail the document as an attachment to the instructor.

Download - <http://www.austincc.edu/hr/profdev/eworkshops/css/practice.html>

CSS Properties

Background Properties

Property	Description	Values
background-color	Sets the background color of an element	color-rgb color-hex color-name transparent
background-image	Sets an image as the background	url none
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	scroll fixed
background-position	Sets the starting position of a background image	top left top center top right center left center center center right bottom left bottom center bottom right <i>x-% y-%</i> <i>x-pos y-pos</i>
background-repeat	Sets if/how a background image will be repeated	repeat repeat-x repeat-y no-repeat

Text Properties:

Property	Description	Values
color	Sets the color of the text.	color
letter-spacing	Increase or decrease the space between characters	normal length
text-align	Aligns the text in an element	left right center justify
text-decoration	Adds decoration to text	none underline overline line-through blink
text-indent	Indents the first line of text in an element	length %
text-transform	Controls the letters in an element	none capitalize uppercase lowercase

Font Properties:

Property	Description	Values
font-family	A prioritized list of font family names and/or generic family names for an element	family-name generic-family
font-size	Sets the size of a font	xx-small x-small small medium large x-large xx-large smaller larger length %
font-style	Sets the style of the font	normal italic oblique
font-weight	Sets the weight of a font.	normal bold bolder lighter 100 200 300 400 500 600 700 800 900

Border Properties

Property	Description	Values
border-color	Sets the color of the four borders, can have from one to four colors	color
border-style	Sets the style of the four borders, can have from one to four styles	none hidden dotted dashed solid double groove ridge inset outset
border-width	A shorthand property for setting the width of the four borders in one declaration, can have from one to four values	thin medium thick length

List Properties:

Property	Description	Values
list-style-image	Sets an image as the list-item marker	url none
list-style-position	Places the list-item marker in the list	inside outside
list-style-type	Sets the type of the list-item marker	none disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-alpha upper-alpha lower-greek lower-latin upper-latin

Resources

Here are links to resources we have used throughout this workshop. Also included are some resources for you to refer to when you are out creating style sheets on your own.

W3Schools Examples - http://www.w3schools.com/css/css_examples.asp

CSS Reference - http://www.w3schools.com/css/css_reference.asp

CSS Zen Garden - <http://www.csszengarden.com/>

Color Selector - <http://www.warpspeedhelp.com/colors.html>

CSS Property Index - <http://www.blooberry.com/indexdot/css/propindex/all.htm>

Everything CSS - <http://www.d.umn.edu/itss/support/Training/Online/webdesign/css.html>