Introduction to Cascading Style Sheet (CSS)

Digital Media Center

129 Herring Hall

http://dmc.rice.edu/
dmc-info@rice.edu
(713) 348-3635

Introduction to Cascading Style Sheets

1. Overview

Cascading Style Sheets (CSS) are a method of web design that formats web page content according to a presentation style specified by the web page author. There are several advantages to using CS to format the presentation elements of a web page. CSS essentially separates document content from the manner in which it is presented, thus allowing for more fluid transitions between various browser platforms. CSS also provides for more precise control for spacing, alignment, and positioning of content without relying on the need for layout tables or frames. Font style, color, and font size can all be manipulated using CSS as well.

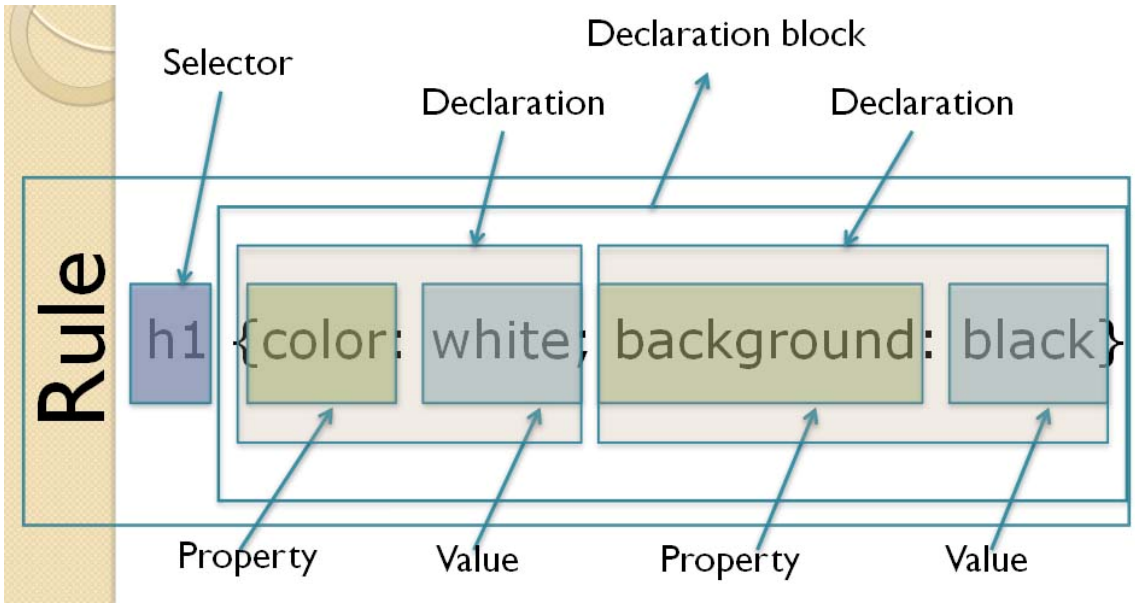
2. The basic CSS Syntax

What is a CSS rule?

A CSS rule is simply a statement that consists of a selector and a declaration.

- **Selector:** is the hook used to choose what part(s) of your HTML to apply the CSS to. It indicates the element to which the rule is applied. Following the selector is the…
- **Declaration Block:** Everything within the curly brackets, "{" and "}", is called the declaration block
- **Declaration:** Inside a declaration block you can have as many declarations as you want and each declaration is a combination of a CSS Property and a value.
- **Property:** is one of the CSS Properties used to tell what part of the selector will be changed (or styled). It specifies a characteristic, such as color, font-family, position, and is followed by a colon (:)
- **Value:** assigns a value to the property.

The property is always followed by a colon, and each declaration is separated with a semicolon



3. Types of selectors

CSS has several types of selectors. Explaining all of them in detail would take too long. We will just introduce you with two most commonly used selectors. For a really in depth study, you should look into the W3C specification for selectors.

Class and ID selectors

These two types have one thing in common: they can be used independently of HTML elements. They can be used on their own or in combination with an element selector. Suppose you want only certain paragraphs to have blue text. P {color: blue;} will not do because that will turn all paragraphs in blue text. So, what you need is a selector you can use whenever you want, regardless of the HTML element. Enter the class and ID selector.

3.1 Class selector

With the class selector you can define different styles for the same type of HTML element.In our example, the class selector would look like this:

```
P.blue {color: blue;}
```

But in order for that rule to work, you will need to modify your HTML like this: this text in this paragraph needs to be blue

What this means is: any paragraph whose CLASS attribute has a value of “blue” will apply the following rule. What if I want other parts of the document to have blue text and not just paragraphs? You can make a class selector without a preceding element: just plain .blue {color: blue;}. Now every element that has a CLASS attribute with a value of blue will have blue text.

The way a class selector works is that it references directly a value that is found in the CLASS attribute of an element. For the styles of a class selector to work, they must be associated with the element in question. How? By having the CLASS attribute of that element set to the appropriate value.

A class selector is always preceded by a period (.). The period is necessary for one reason: it helps to keep the class selector separate from anything you would like to combine it with, for instance an HTML element like we had in our P.blue {color: blue;} rule.

3.2 ID selector

The W3C defines CSS ID as "a unique identifier to an element". IDs are commonly used for Layout and Uniqueness. Standards specify that any given ID name can only be referenced once within a page or document. From our experience, IDs are most commonly used correctly in CSS layouts. This makes sense because there are usually only one menu per page, one banner, and usually only one content pane.

CSS IDs are preceded not by a period but by a pound sign (#). Our “blue” rule would look like this:

#blue {color: blue;}

The big difference between Class and ID

ID = A person's Identification (ID) is **unique** to one person.
Class = There are **many** people in a class.

Use IDs when there is only one occurrence per page or in case you have info that is unique in the document (footer, title, ...); Use classes when there are one or more occurrences per page. In HTML, we must use "id=name" instead of "class=name" to reference it!

4. Grouping

You can group selectors. Separate each selector with a comma.

You can group selectors, declarations or everything. Let's say you want all your heading levels to have the same color. Instead of making a rule for each heading separate, you can group them like this:

H1, H2, H3, H4, H5, H6 {color: blue;}

You group them together by separating each selector with a comma.

If you want H1 to have blue text, Arial font, font-size of 18 pixels and with a red background. Now what? Do you have to write three times those rules with H1? Of course not. You'll group your declarations like this:

H1 { font : 18px Arial; color: blue; background: red;}

5. Positioning

The CSS positioning properties allow you to position an element.

There are five different layout properties available in CSS: position: static, position: relative, position: absolute, and float.

5.1 position: static

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

Static positioned elements are not affected by the top, bottom, left, and right properties.

5.2 position: fixed

An element with fixed position is positioned relative to the browser window.

It will not move even if the window is scrolled.

```
p.pos_fixed
{
position: fixed;
top:30px;
right:5px;
}
```

5.3position: relative

If you specify *position: relative*, then you can use *top* or *bottom*, and *left* or *right* to move the element relative to where it would normally occur in the document.
Let's move div down 20 pixels, and to the left 40 pixels:

```
div { position: relative; top:20px; left:-40px;}
```

5.4position: absolute

When you specify *position: absolute*, the element is removed from the document and placed exactly where you tell it to go.

Let's move div to the top right of the page:

```
div { position:absolute; top:0; right:0; width:200px
```

5. 5 float

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.Float is very often used for images, but it is also useful when working with layouts.

```
img {float:right;}
```

Turning off Float - Using Clear

Any element that you set the clear property for will appear below an element that is floated that direction. You can clear left floats, right floats or both: Add a text line into the image gallery, using the clear property:

```
.text_line { clear:both; }
```

6. Three ways to insert CSS

6.1 Inline CSS

The first way is called Inline Cascading Style Sheets. You can add CSS directly into the elements in your markup with the “style” attribute.

```
<div style=”color:red;”><p style=”font-size:12px;”>
```

“Style” is a “Core HTML Attribute” so you can apply it to any visual HTML/XHTML elements. This may seem to be the easiest way of using CSS, but it is not recommended as it completely ignores the best parts of Cascading Style Sheets.

6.2 Internal Style Sheets

The second way adds CSS to a single document or web page by adding the following code into the <head> of your document using the <style> element.

```
<style type="text/css">
hr {color: sienna}
p {margin-left:20px}
body {background-image:url("images/back40.gif")}
</style>
```

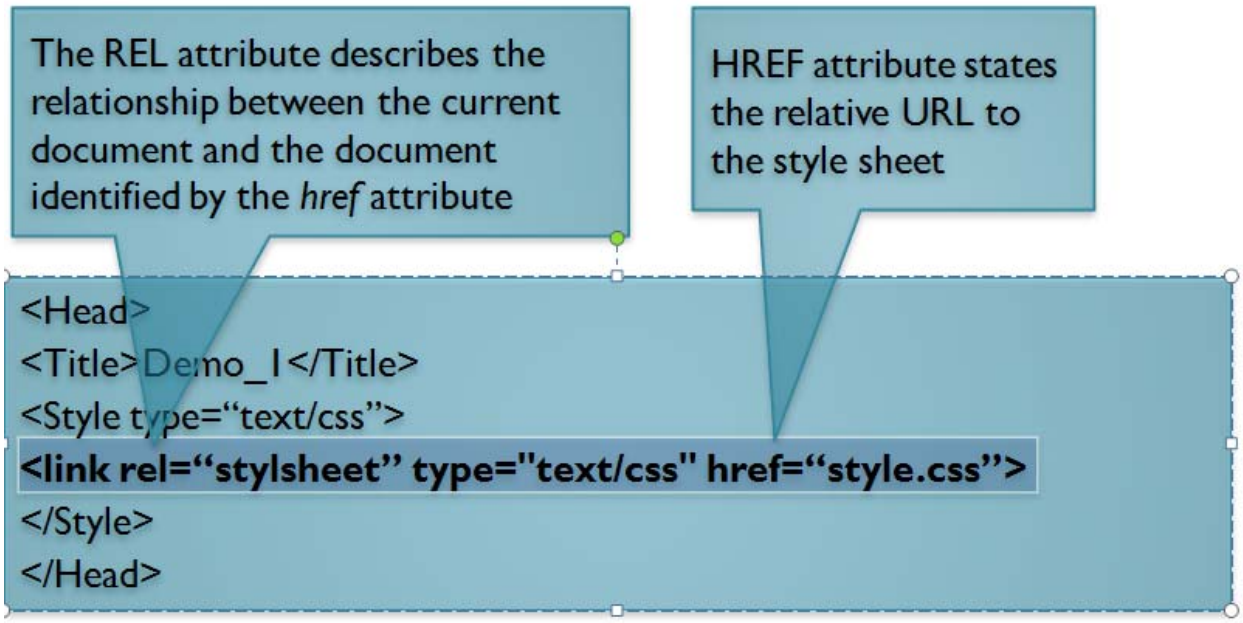
As you can see, <style> is an HTML element that is opened and closed. Within the <style> element is the CSS that will be applied to that page. There is no limit to the amount of CSS you can put inside the <style> element. You would enter this code in the <head> area of your web page (anywhere after <head> and before </head> in your HTML document) and that would apply this CSS to the elements of that web page.

6.3 External Style Sheets

The third and most advantageous way to embed CSS into a document uses **External Cascading Style Sheets** (or External Style Sheets). An external CSS file is simply a text file with a “.css” extension. This file can then be included into many different pages. This allows you to make one document that has the styling for your entire website. Then you include that file into every page of your website. This is the recommended way of using CSS in your website as it lets you make global visual changes (to an entire website) by only tweaking one file.

For Example, let’s say you have a website where all the paragraphs are black, and you want to change them to grey. Since you have an external Style Sheet, you just open up that one CSS file and change the style for paragraphs to: color:grey; Now all the paragraphs on your website are grey!

Here is the code you need to link the css fileto your webpages”:
<link rel=“stylsheet” type="text/css" href=“style.css”>



7. Media Types

There are about 10 different media types in CSS, but most common types are ***Screen*** and ***Print***. CSS today allows you to link to more than one style sheet in your web page.

The simple reason you would want to do this is so that you could have the HTML page 'change' its appearance automatically when someone visits your page with a particular type of device. These types of devices can include a typical desktop computer, a PDA (Windows CE device, Palm Pilot, etc.) and a printer among other devices!

Basic ideas:

- People will print your pages typically to be able to read the content, not to see pictures. You need to hide all images from the printer and this is done in the print style sheet.
- Navigational elements are not needed in the printed document as well, so all navigational elements should be removed from the printed page.
- Let's say you designed your pages with a black background with white text. You don't want peoples printers printing all that black, they won't be too happy with the price of ink these days! To solve this problem, in our print style sheet we will set the page color to 'white' and the text color to 'black'.

It works like this; when you link a style sheet to your HTML page, there is an attribute that you can specify in the CSS link tag that tells the device reading your page if it should use the style sheet specified in the link.

Screen media type:

<link href="style.css" rel="stylesheet" type="text/css" media="screen">
It (‘media=”screen”’) is set up for computer screens.

Print media type:

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

It ('media="print"') points to the style sheet that has been set up for printing.

Nowadays most browsers know that if someone tries to print the page the style marked with: 'media="print"' should be used.

For more information, visit
http://www.killersites.com/articles/newsletterArchive/Newsletter_Nov3_2003.htm

8. Resources

http://chrispederick.com/work/webdeveloper/ (Firefox Web Developer Extension)
http://www.w3schools.com/css/ (highly recommended)
http://www.yourhtmlsource.com/stylesheets/
http://www.meyerweb.com/eric/css/
http://www.autisticcuckoo.net/archive.php?id=2004/12/07/relatively-absolute
http://www.barelyfitz.com/screencast/html-training/css/positioning/
http://www.killersites.com/articles/newsletterArchive/Newsletter_Nov3_2003.htm

One more thing…

Browser Testing

It is suggested to test using standards compliant browsers that support CSS well (Firefox, Mozilla, Safari, Opera) and then test as you go in Internet Explorer and try to overcome any problems while keeping your code valid. Most important, forget the idea that you are trying to create a page that looks the same in all browsing environments.
Focus instead on creating a page that looks good in all browsers you care to target, and supports the standards so it will continue to look good in the future.