Style Sheets

1 Why Cascadeing Style Sheet?

1.1 Compatibility

HTML tags were originally designed to define the content of a document. They were supposed to say “This is a header”, ”This is a paragraph”, ”This is a table”, by using tags like `<h1>`, `<p>`, `<table>`, and so on. Unfortunately, once you have been playing with HTML for a while, you would find out a same HTML document is very likely to have different appearance in different web browsers.

- Due to the limitation of HTML, you may realize how little control you have over formatting of a page. HTML does not provide mechanisms for indentation, justification, line spacing, character spacing, absolute position, or other layout attributes. All these are supposed to be taken care of by the browsers themselves.
- To provide more control over HTML, the two major browsers - Netscape and Internet Explorer - turned to add proprietary tags into HTMLs, i.e. tags that are only recognized by a single kind of products, which made the situation even worse.

Thus, sometimes HTML designers have to detect which browser on the client side is used and different versions of HTML files are deployed on the server side, each for one type of browser. This causes a heavy burden upon HTML designers.

1.2 Reusability

What’s more, beyond the incompatibility of different browsers, it is hard to maintain a whole bunch of HTML files which use the same style. For example, we want that all of them use a same color for the heading text, or Arial font is used for text in tables. As we know from the last class, we may use the `<font>` elements embedded in the HTMLs to achieve the goal, but how about later on we want another color or face used instead? Imaginably, we have to update each appearance of those elements to reflect our new goal. This is obviously awkward.

If you have learned object orientation theory, or database theory, you surely know now that redundancy is one of the major issues they deal with. In a program, you should by any means avoid the repeated appearance of code segments that have the same or similar functions. In designing a database, each data item had better not appear more than once in the database. If those situations exist, you will have to take higher costs to maintain the program or the database. The process is also error-prone. The less you do, the fewer mistakes you may make.

The same law holds for HTML design. We should find a way to separate the content of HTML from its style so that we may define a style once and reuse it wherever we want. The answer is *CSS*, short for *Cascadeing Style Sheet*.

2 What CSS can do?

CSS basically has the following advantages:

- You can control layout like never before.
- You can separate content from formatting.
- You can make smaller, faster pages.
- You can maintain or update many pages at once, faster and easier than before.

With CSS, we may define styles in a separate file and then apply them to all of the pages of concern. This gives us much more control and helps to provide a consistent look throughout all of the pages.

3 Cascading Style Sheet Basics

Ever use the style feature of Microsoft Word? It allows you to apply a specific formatting style to text in your Word document. There are styles such as Heading 1, Bullet, and Normal. If you want to have a certain look for some of the text in the document, you can define the font, paragraph style, and other attributes and then save this set of attributes as a named style. You can then apply this style to text in the document and that text is displayed with the defined font, paragraph style, and other attributes. CSS provides the same type of functionality in your web pages.

3.1 Style Sheets

To define the formatting styles and rules with CSS, you should use style sheets. A *style sheet* is a set of style rules that describe how HTML document elements are to be displayed on the page. Each style rule in the style sheet looks similar to this:

```
p {font-family:arial; font-size:20pt; color:red}
```

Or for better readability,

```
p {
  font-family:arial;
  font-size:20pt;
  color:red
}
```

Each style rule is comprised of two parts, a selector and a declaration. The *selector* part of the style rule specifies which HTML elements are to be affected by the rule. The style rule above is defined for `p` (paragraph) elements. This rule is applied automatically to every `p` element in every HTML document that uses this style sheet.

A selector can specify more than one element by separating the elements with commas. The following style rule applies to all `h1` and `h2` heading elements:

```
h1 h2 {font-family:arial; color:navy}
```

The *declaration* part of a style rule, contained within braces (`{}`), defines the style properties to be applied to the specified elements. The properties are comprised of name and value pairs. Each name is followed by a colon (`:`) and one or more values. The property name defines the attribute, such as `color` or `font-family`. The property value defines the specification of that attribute. Each name and value pair is separated by a semicolon (`;`). As a result of the first example of style rule, all of the `p` elements on the web page are displayed in red, 20-point Arial font.

At this point, you may be thinking that you can achieve the same result using HTML elements and their associated attributes. For this particular example, you may use:

```
<p><font face="arial" size="20pt" color="red">Some text</p>
```

But what if you have 20 paragraphs? Do you want to repeat that font definition within every paragraph? What if you decide later that you really want that text to be blue? Apparently, style sheets bring many benefits to us.

4 Defining Styles

There are 3 basic ways to define styles for web pages: inline, embedded, and external.

4.1 Inline Styles

You can specify a style rule directly within an HTML element using the `style` attribute of the element. This is useful when you want a style applied only for a particular element on your web page. For example,

```
<p style="font-family:arial; font-size:20pt; color:red">Some text</p>
```

Since the style rule is defined within the element's tag, there is no need for a selector, only the declaration.

Using an inline style is better than using the traditional individual element attributes because many of the formatting attributes have been deprecated and are supposed not to be used any more. In addition, there are more style properties than element attributes, giving you more control over the look of the element. For example, you may use `margin-left` style property allows you to set a left margin for the paragraph. There is no comparable paragraph attribute.

Nevertheless, as you can see, we cannot benefit more from CSS besides the above, especially from the viewpoint of reusability. For more benefits, we need to use embedded and external styles.

4.2 Embedded Styles

You can define a consistent style for all elements of a specific type using embedded styles. This style is automatically applied to that element type throughout the page, giving you a standard look on the page.

You declare an embedded style with a `style` element in the header section of the HTML document. The `style` element contains one or more style rules. For example:

```
<head>
<title>Welcome to Internet Programming</title>
<style type="text/css">
  <!--
  p {font-family:arial; font-size:15pt; color:navy}
  h1 {color:navy}
  body {background-color:#D2B48C}
  -->
</style>
</head>
```

The `style` element has a `type` attribute that indicates the types of style rules to use; in this case, it is CSS. This attribute is required for strict conformance to XHTML. The comment markers within the `style` element are not required but highly recommended so that older browsers that don't understand the `style` element ignore the style definition.

4.3 External Styles

Above the style rules are defined in the HTML document that uses the styles. To reuse the styles throughout multiple pages of your web application, we can instead create the styles in an external file and link that file to all of HTML documents.

Later, if you want to change the way all of your pages look, simply change the external style sheet file and all of the pages in your application linked to the style sheet file reflect the change.

To create an external style sheet, simply create a text file for the style rules. This file is called a style sheet file and is normally given a `.css` extension. Add your styles to that file. The style rules are the same as for the embedded styles.

Any HTML document that wants to use the style from the style sheet file can link to that style sheet file. Use the `link` element in the header section of the HTML document to link to the desired external sheet file:

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

The `rel` attribute of the `link` element defines the nature of the relationship between the linked resources; in this case, the link is to a style sheet. The `type` attribute defines the type of the linked file (always `text/css` for cascading style sheets). The name of the style sheet file is defined with the `href` attribute using relative or absolute paths.

5 Declaring Style Classes for Flexibility

The styles defined so far have been for specific elements or specific types of elements. How if we want different styles applied respectively to different groups of elements of the same type? CSS has the mechanism called *style classes* to allow us to declare named styles that can be applied to specific elements on a web page.

Remember when you specify a particular HTML element as the selector of a style rule, the style properties are automatically applied to all occurrences of that particular type of element. With style classes, you may define different styles for different types of the same element. For example, you can define two types of paragraphs: key points and normal information. If you want the key points to stand out when your page is displayed, you could make the key point paragraph font bigger and red. You can define two different styles for the two different types of paragraphs by defining two style classes as follows:

```
<style type="text/css">
  <!--
  p {margin-left:0.5in; font-family:arial}
  p.key {font-size:20pt; color:red}
  p.normal {font-size:15pt; color:navy}
  .margin {margin-left:0.5in}
  -->
</style>
```

As the example shows, you declare a style class using a dot (`.`) and a style class name in the selector part of the style rule. The style class names can be any name that provides a good description of the usage of the style. If the style class is only to be used for a particular type of element, the element name precedes the style class definition.

In the example, both the `key` and `normal` style classes can only be used with `p` elements, while the `margin` style class is not associated with any particular element type, so it can be used with any relevant element.

In the example, the `p` element style is applied automatically to every `p` element on the web page. But to assign a style class to a particular HTML element, you must set the `class` attribute of that element to the style class name. For example:

```
<body>
<h1 class="margin">Welcome to Internet Programming</h1>
<p class="key">Announcement</p>
<p class="normal">We won't have class tomorrow.</p>
</body>
```

And any `p` element without a `class` attribute uses the standard `p` element style.

Style and style class declarations assume that you want to apply a style to several elements. There may be situations where you want to define a unique style for an individual element. You can do this with inline styles by setting the `style` attribute of the element as discussed previsouly. However, you may want to keep all of your style information together. In this case, you can define a style rule using the value of the element's `id` attribute.

To apply a style to a specific element, there must be a way to identify that element. To do so, you have to define the `id` attribute of that element and then use a pound sign (#) and the value of the element's `id` attribute as the style selector. For example:

```
#LastName {font-size:15pt; color:brown}
```

6 Cascading Style Sheets

A key feature of CSS is that style rules can *cascade*. That is several style rules may work together for the same set of elements and all of the rules can influence the presentation of those elements.

As we discussed above, you may define inline, embedded, or external style rules. They may overlap with each other when the same type of elements is used as selector. In this case, all the style attributes are applied.

But how if there is a conflict, e.g. different colors assigned in different style rules? CSS assigns different priorities to the style rules defined differently:

- $Inline > Embedded > External$
- Specific selectors always take priority over general selectors. A style rule with a class defined in the selector beats a style rule without a class. And a style rule with an element ID beats a style rule with a class.
- When multiple style rules apply to the same element, the style rule defined last takes precedence.

The cascading effect can be very useful when you define a general style for a large range of elements and then define a different style for a subset of them.