HTML

This document explains the major artifacts of HTML, which though probably you won't deal with directly with graphical HTML editors available, you'd better know to some extent what HTML files are and how they work. With years of programming experience, there should be no difficulty for us to do so in a short period.

1 HTML Basics

As we know, HTML files make it possible for people to share information and resources. With a web client, a user may request a resource, in most cases in the form of HTML, from a web server.

HyperText Markup Language (*HTML*) is the basic language for the Web. As its name implies, it defines a standard syntax for marking up a document. It allows you to specify formatting, layout, and style for the document. Web browsers then interpret the markup syntax and display your document as a web page.

1.1 Tags

To distinguish between the markup syntax and the effective content, *tags* are used. Each tag is surrounded by angle brackets (<>) with its *name* and possible *attributes* specified inside. It's these tags that tell the browser how to display the web pages.

HTML is not case-sensitive, so tag names and attributes can be uppercase, lowercase, or proper case. For compatibility with the current standards (i.e. XHTML, which comes later), lowercase is recommended.

1.2 Defining Elements

With tags, HTML *elements* may be defined to specify the formatting, layout, and style of web pages.

An HTML element is defined with a beginning tag and an ending tag. The starting tag consists of a left angle bracket (<), the element name, and a right angle bracket (>) with no spaces

between the angle bracket and the element name. the ending tag consists of a left angle bracket (<), a forward slash (/), the element name, and a right angle bracket (>). The text between the starting tag and the ending tag is the content of the element. For example:

```
<h1>Welcome to Internet Programming!</h1>
```

defines a level 1 heading.

1.2.1 Empty Elements

If an element has no content, it is *empty*. For example, the *br* (line break) element has no content; it simply breaks to a new line. For example:

```
View a list of our course.<br></br>Or, sign up for our mailing list.
```

With empty elements, we may combine the ending tags with the beginning one as follows:

```
View a list of our course.<br />Or, sign up for our mailing list.
```

1.2.2 Nesting Elements

Like control structures in programming languages, elements can contain other elements. Between the beginning and ending tag of one element you can insert the beginning and ending tag of another element. For example:

```
<head>
<title>Welcome to Internet Programming</title>
</head>
```

Notice how the *title* element tags are nested entirely within the head element tags. This capability to nest allows you to nest fonts within paragraphs, rows within tables, and controls within forms.

Nevertheless, HTML tags should not overlap. That is, if a tag begins within another tag, it should also end within that tag. If all of the elements in your HTML document are defined with beginning and ending tags and the tags do not overlap, the HTML document is well-formed. For example, the following is not well-formed:

```
<h1>Welcome to our <font color="red">Training Page</h1>
Our training courses are the best in the industry.</font>
```

which should actually be written in the following way:

```
<h1>Welcome to our <font color="red">Training Page</font></h1>
<font color="red">Our training courses are the best in the industry.</font>
```

1.2.3 Element Types

Based on their roles, HTML elements can be divided into two groups: *inline* and *block-level*.

Basically block-level elements define the HTML document structure while inline elements do not. Inline elements are used to change the content's attributes but don't display the content on a new line. The *font* element is an example of this type. Inline elements can only contain text or other inline elements, but block-level elements can contain other block-level elements, inline elements, or text. The *h1* element is an example of this kind.

1.3 Specifying Attributes

Attributes provide a greater level of control over how the element behaves or looks. An attribute is defined by an attribute name followed by an equal sign (=) and a value. The attribute value should be quoted using either double quotes (" ") or single quotes (' '). Quotation marks are not required unless the attribute value contains special characters, though you are recommended to use quotation marks around all attribute values for compatibility with the new XHTML standards.

Attribute name and value pairs are defined within the beginning tag for the element, after the tag name and before the right angle bracket (>). You can specify multiple attributes for an element by separating them with a space. For example:

```
<font face="arial" size="4">Our college is one of the best in this country.</font>
```

2 The Anatomy of an HTML Document

Most HTML documents follow the same basic structure shown below:

```
<html>
<!-- Welcome Page -->
<head>
<title>Welcome to Internet Programming</title>
</head>
<body bgcolor="#D2B48C">
<h1> Welcome to CSc31800: Internet Programming</h1>
<font face="arial" size="4">Our college is one of the best in this country.
  </font>
</body>
</html>
```

Every well-formed web page includes the tags for the *html* element, which is the root element of the page and defines that the content of the file is HTML. the rest of the page is divided into two main sections, the header and the body.

2.1 Comments

Notice that a *comment* line, like line 2 in the above example, can exist anywhere in the page. Any text between the beginning comment marker (`<!--`) and the ending marker (`-->`) is considered to be a comment and won't be displayed in the browser window. However when the users open the HTML document to read the source, the comments do appear, so don't put anything in the comments if you don't want the users to read.

2.2 Header

The *header* section contains the preparatory information for the page, and style definition and scripting as well. But it doesn't contain any content to be displayed on the web page. This section at least includes the page title (line 4 in the example), which should provide a good description of the page.

The page title is important because it is used in many ways. The browser displays this title in the browser's title bar, search engines list this title when the page is found by a search, the browser uses the title by default when the user saves the page as a favorite link, and the title is used in the history list.

2.3 Body

The *body* section of an HTML document (lines 6 through 9) contains the information to be displayed in the browser, including the content for the page and the elements that define the format of the page.

In the example, line 7 defines a line of text to appear on the page using the level 1 heading style. The heading style is different from the title in that the *title* element defines the logical name of the page and *h1* element simply defines a style of text on the page.

The second element within the body defines text to appear with the defined font and size.

Any text in the HTML document that is not associated with an element is displayed in the browser as is.

You can freely indent or add whitespace to the HTML document as needed for readability without impacting the display of the page. Tabs, carriage returns, and all extra space in the text of your HTML document are ignored. For example,

```
<font face="arial" size="4">
    Our college is one of the best in this country.
</font>
```

If you do want to start a new line, use the *br* element, or start a new paragraph with the *p* element. To insert white space, use the ` ` (non-breaking space) character entity. There is no HTML equivalent for a tab, but you can set margins using styles, which we will talk about next time.

2.4 Displaying Special Characters

As we know, some characters are used to define the HTML syntax. Then how about displaying those characters as text? To do so, HTML defines several character entities, each starting with `&` and ending with `;`.

`&`

: &, ampersand. Since the ampersand has special meaning in HTML, this entity allows you to define an ampersand in the text.

`&apos;`

: ', apostrophe.

`&copy;`

: ©, copyright symbol.

`>`

: >, greater than.

`<`

: <, less than.

` `

: Nonbreaking space.

`&quot;`

: Quotation mark.

2.5 Working with URI

Some HTML elements, such as anchors and images, require specification of an URL or URI, which gives the specific location of another HTML document, image, or any other file on the Web.

URIs can be *relative* or *absolute*. An absolute URI defines the entire path to a file including the protocol, the web server name, the complete path, and the filename. The following defines an `a` (anchor) element that uses an absolute URI and defines a hyperlink to a Course Catalog HTML file:

```
<a href="http://www.somewhere.com/CourseCatalog.html">
  View the course catalog.
</a>
```

A relative URL describes the location of the desired file with reference to the location of the current page. For example:

```
<a href="CourseCatalog.html">
  View the course catalog.
</a>

<a href="../home/catalog/CourseCatalog.html">
  View the course catalog.
</a>

<a href="../home/catalog/CourseCatalog.html">
  <img src="../images/icon.gif" alt="Catalog" />
  View the course catalog.
</a>
```

2.6 Working with Colors

Many HTML elements have attributes, such as `bgcolor`, which requires specification of a color. A color can be defined by a logical name or by its red, green, and blue (RGB) color values.

The logical names include simple colors such as “black” and “navy” and fancier color names like “cornflowerblue” and “sandybrown”.

An RGB color value is comprised of three hexadecimal numbers that specify the intensity of each of the three colors: red, green, and blue. The values range from 00, defining none of that color, to FF, defining the maximum amount of that color. For example, #FF0000 is max red, no green, and no blue, resulting in red.

2.7 Working with Fonts

When you have a lot of information to display on a web page, varying fonts can help the readers to recognize the logical difference between different parts of text. Font definition includes the style or face, size, and color. For example:

```
<font face="arial" size="4" color="red">
  Our college is the best in the metropolitan area.
</font>
```

3 XHTML

The "X" in XHTML represents the application of the eXtensible Markup Language (XML) to HTML. The World Wide Web Consortium (W3C) has recommended the XHTML specification as the newest version of HTML and calls it "a reformulation of HTML 4 in XML 1.0."

XML, which will be covered in the future classes, allows the definition of a precise syntax. By applying XML to HTML, you can develop HTML that conforms to a rigorous standard so that the HTML document may have a more consistent interpretation and appearance in all browsers.

3.1 Specifying the HTML Version

To follow XHTML standards, you need declare the version of HTML you used in your HTML files. The definition, for example, should go in the following way:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "DTD/xhtml1-transitional.dtd">
```

This specifies the version of HTML in your HTML document, in this case, XHTML 1.0.

The DOCTYPE declaration also identifies the *Document Type Definition* (DTD) for the HTML document. A DTD is basically an XML file, defining the set of elements and attributes that are valid for the HTML version you specified. The DTD can be used by a browser to validate your HTML.

With XHTML 1.0, you have 3 DTD choices:

- Validate your HTML based on the strict XHTML standard. Use this when you want clean structural HTMl without any tags that have been replaced by style sheets.

  ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "DTD/xhtml1-strict.dtd">
  ```

- Validate your HTML based on the transitional standard between HTML 4 and XHTML. Use this when you want to use the new XHTML features but also support older browsers.

  ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "DTD/xhtml1-transitional.dtd">
  ```

- Validate your HTML based on using frames. Use this only when you want to use HTML frames.

  ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "DTD/xhtml1-frameset.dtd">
  ```

4 Exercise