Introduction to HTML

HTML is the HyperText Markup Language. It is used to describe how text, images, and multimedia are displayed by Web browsers. In this lab you will learn a little about HTML so that you can create a web page containing headings, interesting fonts, lists, and links as well as applets.

HTML uses *tags* to describe the layout of a document; the browser then uses these tags to figure out how to display the document. Tags are enclosed in angle brackets. For example, <title> is a tag that indicates that this section contains the title of the document. Many tags, including <title>, have corresponding end tags that indicate where the section ends. The end tags look just like the start tags except that they start with the character /, e.g., </title>. So the following text indicates that the title of the document is `Introduction to HTML`:

```
<title>Introduction to HTML</title>
```

There are a few tags that almost every document will have: <html>, <head>, <title>, and <body>. Here is an example of a simple HTML document:

```
<HTML>
  <HEAD>
    <TITLE>Introduction to HTML</TITLE>
  </HEAD>

  <BODY>
  In this lab you will learn about HTML, which is lots of fun to
  use.  In particular, you will learn how to use fonts,
  paragraphs, lists, links and applets in a web page.  Now you
  can make your own web page for your friends to visit!
  </BODY>
</HTML>
```

To see what this looks like, open the file in the web browser. Change the size of the browser window (click and drag any corner) and see how the text is reformatted as the window changes. Note that the title appears on the window, not as part of the document.

The HEAD of a document (everything between <HEAD> and </HEAD>) contains the introduction to the document. The title goes in the head, but for now we won't use the head for anything else. The BODY of a document (everything between <BODY> and </BODY>) contains everything that will be displayed as part of the document. Both the HEAD and the BODY are enclosed by the HTML tags, which begin and end the document.

This document contains only plain text, but an HTML document can have much more structure: headings, paragraphs, lists, bold and italic text, images, links, tables, and so on. Here is a document containing a heading, two paragraphs, and some fancy fonts:

```
<HTML>
  <HEAD>
    <TITLE>Introduction to HTML</TITLE>
  </HEAD>

  <BODY BGCOLOR="lightgreen">
  <H1 align="center">Introduction to HTML</H1>

  <P>In this lab you will learn about <I>HTML</I>, which
  is lots of fun
  to use.  In particular, you will learn how to use fonts,
  paragraphs, lists, links, and colors in a web page.  Now you
  can make your <B>own</B> web page for your friends to visit!</P>

  <P>Later in this lab you will do some fancier stuff with
  applets and graphics and include an applet on your web page.
  Can't you just feel the job offers start rolling in?</P>
  <U>Yippee!</U>
  </BODY>
</HTML>
```

Run the HTML document to see what this looks like in the browser.

In this document the <H1> tag creates a level 1 heading. This is the biggest heading; it might be used at the beginning of the document or the start of a new chapter. Level 2 through level 6 headings are also available with the <H2> through <H6> tags.

The <P> tag creates a new paragraph. Most browsers leave a blank line between paragraphs. The <B> tag creates bold text, the <I> tag creates italic text, and the <U> tag creates underlined text. Note that each of these tags is closed with the corresponding end tag. The BGCOLOR attribute on the BODY tag sets the background color.

Note that line breaks and blank lines in the HTML document do not matter—the browser will format paragraphs to fit the window. If it weren't for the <P> tag, the blank line between the paragraphs in this document would not show up in the displayed document.

Exercise #1: For a file to be visible on the Web, it must be where the web server knows how to find it. *** Instruct students how to create and/or access the public html directory on the local system. ***

Open a new file called MyPage.html in a directory where it will be accessible from the web. Write a simple web page about things that interest you. Your page should contain at least the following:

- A title (using the <TITLE> tag)
- Two different levels of headings
- Two paragraphs
- Some bold, italic, or underlined text

Your name should appear somewhere in the document.

When you are done, view your document from the browser. Just type in the URL, don't use File | Open Page.

More HTML

Lists We often want to add a list to a document. HTML provides two kinds of lists, *ordered* (e.g., 1, 2, 3) and *unordered* (e.g., bulleted). A list is introduced with the <OL> or <UL> tag, depending on whether it is ordered or unordered. Each list item is introduced with a <LI> tag and ended with the </LI> tag. The entire list is then ended with </OL> or </UL>, as appropriate. For example, the code below creates the list shown; replacing the <OL> and </OL> tags with <UL> and </UL> would produce the same list with bullets instead of numbers.

```
Things I like:
<OL>
<LI>chocolate
<LI>rabbits
<LI>chocolate rabbits
</OL>
```

Things I like:

1. chocolate
2. rabbits
3. chocolate rabbits

Exercise #2: Add a list, either ordered or unordered, of at least three elements to your document.

Links Links connect one document to another. Links are created in HTML with the <A> (anchor) tag. When creating a link you have to specify two things:

- The URL of the document to go to when the link is clicked. This is given as the HREF attribute of the A element.
- How the link should be displayed (that is, what text or image to click on to go to the linked document). This appears between the <A> and </A> tags.

For example, the code below creates the link shown, which goes to a page about the history of computing:

```
Learn more about <A HREF="http://ei.cs.vt.edu/~history">the history of
computing.</A>
```

Learn more about the history of computing.

Exercise #3: Add at least one link that ties in to the material on your page.

Drawing Shapes

The following is a simple applet that draws a blue rectangle on a yellow background.

```
// ********************************************************
//   Shapes.java
//
//   The program will draw two filled rectangles and a
//   filled oval.
// ********************************************************

import javax.swing.JApplet;
import java.awt.*;

public class Shapes extends JApplet
{
    public void paint (Graphics page)
    {
      // Declare size constants
      final int MAX_SIZE = 300;
      final int PAGE_WIDTH = 600;
      final int PAGE_HEIGHT = 400;

      // Declare variables
      int x, y;    // x and y coordinates of upper left-corner of each shape
      int width, height; // width and height of each shape

      // Set the background color
      setBackground (Color.yellow);

      // Set the color for the next shape to be drawn
      page.setColor (Color.blue);

      // Assign the corner point and width and height
      x = 200;
      y = 150;
      width = 100;
      height = 70;

      // Draw the rectangle
      page.fillRect(x, y, width, height);
    }
}
```

Study the code, noting the following:

- The program imports javax.swing.JApplet because this is an applet, and it imports java.awt.* because it uses graphics.
- There is no *main* method—instead there is a *paint* method. The paint method is automatically invoked when an applet is displayed, just as the main method is automatically invoked when an application is executed.
- Most of the methods that draw shapes (see the list in Figure 2.12) require parameters that specify the upper left-hand corner of the shape (using the coordinate system described in Section 2.9) and the width and height of the shape. You can see this in the calls to *fillRect,* which draws a rectangle filled with the current foreground color.
- This applet will be drawn assuming the window for drawing (the Graphics object - named page here) is 600 pixels wide and 400 pixels high. These numbers are defined in constants at the beginning of the program. (They currently have no use but you will use them later). The width and height of the applet are actually specified in the HTML file that instructs the Web browser to run the applet (remember applets are executed by Web browsers and Web browsers get their instructions from HTML documents—note that the code executed by the browser is the *bytecode* for the program, the *Shapes.class* file). The code in the HTML document is as follows:

```
<html>
<applet code="Shapes.class" width=600 height=400>
</applet>
</html>
```

Save the files *Shapes.java* and *Shapes.html* to your directory. Now do the following:

1. Compile Shapes.java, but don't run it—this is an applet, so it is run through a browser or a special program called the Applet Viewer.

2. Run the program through your browser. You should see a blue rectangle on a yellow background.

3. Now run the program through the Applet Viewer by typing the command

   ```
   appletviewer Shapes.html
   ```

   You should see a new window open displaying the rectangle.

4. Now open the program in your text editor and change the x and y variables both to 0. Save and recompile the program, then view it in the Applet Viewer (this is generally less trouble when making lots of changes than using the browser). What happened to the rectangle?

5. Now change the width to 200 and the height to 300. Save, recompile and run to see how this affects the rectangle.

6. Change x to 400, y to 40, width to 50 and height to 200. Test the program to see the effect.

7. Modify the program so that it draws four rectangles in all, as follows:
   - One rectangle should be entirely contained in another rectangle.
   - One rectangle should overlap one of the first two but not be entirely inside of it.
   - The fourth rectangle should not overlap any of the others.

8. One last touch to the program... Change the colors for at least three of the shapes so the background and each of the three shapes are different colors (a list of colors is in Figure 2.10 of the text). Also change two of *the fillRect* methods to *fillOval* so the final program draws two rectangles and two ovals. Be sure that the overlap rules are still met.

The Java Coordinate System

The Java coordinate system is discussed in Section 2.7 & 2.9 of the text. Under this system, the upper left-hand corner of the window is the point (0,0). The X axis goes across the window, and the Y axis goes down the window. So the bigger the X value, the farther a point is to the right. The bigger the Y value, the farther it is down. There are no negative X or Y values in the Java coordinate system. Actually, you can use negative values, but since they're off the screen they won't show up!

1. Save files *Coords.java* and *Coords.html* to your directory. File Coords.java contains an applet that draws a rectangle whose upper lefthand corner is at 0,0. Use the applet viewer to run this applet. Remember that you have to do it through the html file: *appletviewer Coords.html.*

2. Modify the applet so that instead of 0,0 for the upper lefthand corner, you use the coordinates of the middle of the applet window. This applet is set up to be 600 pixels wide and 400 pixels high, so you can figure out where the middle is. Save, compile, and view your applet. Does the rectangle appear to be in the middle of the screen? Modify the coordinates so that it does appear to be in the middle.

3. Now add four more rectangles to the applet, one in each corner. Each rectangle should just touch one corner of the center rectangle and should go exactly to the edges of the window.

4. Make each rectangle be a different color. To do this, use the setColor method of the Graphics class to change the color (this is already done once). **Do not change the background color once it has been set!** Doing so causes the screen to flicker between colors.

```
// ********************************************************
//   Coords.java
//
//   Draw rectangles to illustrate the Java coordinate system
//
// ********************************************************

import javax.swing.JApplet;
import java.awt.*;

public class Coords extends JApplet
{
    public void paint (Graphics page)
    {
      // Declare size constants
      final int MAX_SIZE = 300;
      final int PAGE_WIDTH = 600;
      final int PAGE_HEIGHT = 400;

      // Declare variables
      int x, y;    // x and y coordinates of upper left-corner of each shape
      int width, height; // width and height of each shape

      // Set the background color
      setBackground (Color.yellow);

      // Set the color for the next shape to be drawn
      page.setColor (Color.blue);

      // Assign the corner point and width and height
      x = 0;
      y = 0;
```

```
      width = 150;
      height = 100;

      page.fillRect(x, y, width, height);
    }
}
```

Coords.html

```
<html>
<applet code="Coords.class" width=600 height=400>
</applet>
</html>
```

Drawing a Face

Write an applet that draws a smiling face. Give the face eyes with pupils, ears, a nose, and a mouth. Use at least three different colors, and fill in some of the features. Name this file Face.java, and create a corresponding .html file. View your applet using the applet viewer.

Now add your face to a web page you created (you may have created one in an earlier lab exercise). You will do this using the <APPLET> tag that is in the .html file you are using with the applet viewer—you can just copy it out of that file and paste it into your other file. The applet will appear wherever you insert the applet tag.

Creating a Pie Chart

Write an applet that draws a pie chart showing the percentage of household income spent on various expenses. Use the percentages below:

Rent and Utilities	35%
Transportation	15%
Food	15%
Educational	25%
Miscellaneous	10%

Each section of the pie should be in a different color, of course. Label each section of the pie with the category it represents— the labels should appear outside the pie itself.

Embed your applet in an HTML document about managing expenses. It should contain a heading, some text, a relevant list, and at least one link to a relevant page. Your name should also appear somewhere on the page. Embed the applet so that it fits in nicely.

Colors in Java

The basic scheme for representing a picture in a computer is to break the picture down into small elements called *pixels* and then represent the color of each pixel by a numeric code (this idea is discussed in section 1.6 of the text). In most computer languages, including Java, the color is specified by three numbers—one representing the amount of red in the color, another the amount of green, and the third the amount of blue. These numbers are referred to as the *RGB value* of the color. In Java, each of the three primary colors is represented by an 8-bit code. Hence, the possible base 10 values for each have a range of 0-255. Zero means none of that color while 255 means the maximum amount of the color. Pure red is represented by 255 for red, 0 for green, and 0 for blue, while magenta is a mix of red and blue (255 for red, 0 for green, and 255 for blue). In Java you can create your own colors. So far in the graphics programs we have written we have used the pre-defined colors, such as Color.red, from the Color class. However, we may also create our own Color object and use it in a graphics program. One way to create a Color object is to declare a variable of type Color and instantiate it using the constructor that requires three integer parameters—the first representing the amount of red, the second the amount of green, and the third the amount of blue in the color. For example, the following declares the Color object *myColor* and instantiates it to a color with code 255 for red, 0 for green, and 255 for blue.

```
Color myColor = new Color(255, 0, 255);
```

The statement*page.setColor(myColor)* then will set the foreground color for the page to be the color defined by the *myColor* object. The file *Colors.java* contains an applet that defines *myColor* to be a Color object with color code (200, 100, 255) - a shade of purple. Save the program and its associated HTML file *Colors.html* to your directory, compile and run it using the appletviewer. Now make the following modifications:

1. Change the constructor so the color code is (0,0,0) — absence of color. What color should this be? Run the program to check.
2. Try a few other combinations of color codes to see what you get. The description of the Color class in Appendix M contains documentation that gives the codes for the pre-defined colors.
3. Notice in the Color class in Appendix M there is a constructor that takes a single integer as an argument. The first 8 bits of this integer are ignored while the last 24 bits define the color—8 bits for red, 8 for green, and the last 8 bits for blue. Hence, the bit pattern

   ```
   00000000000000001111111100000000
   ```

   should represent pure green. Its base 10 value is 65280. Change the declaration of the *myColor* object to

   ```
   Color myColor  =  new Color   (65280);
   ```

   Compile and run the program. Do you see green?

4. The Color class has methods that return the individual color codes (for red, green, and blue) for a Color object. For example,

   ```
   redCode = myColor.getRed();
   ```

   returns the code for the red component of the myColor object (redCode should be a variable of type int). The methods that return the green and blue components are *getGreen* and *getBlue,* respectively. Add statements to the program, similar to the above, to get the three color codes for *myColor* (you need to declare some variables). Then add statements such as

   ```
   page.drawString("Red: " + redCode, ____ , ____ );
   ```

   to label the rectangle with the three color codes (fill in the blanks with appropriate coordinates so each string is drawn inside the rectangle—you also need to set the drawing color to something such as black so the strings will show up). Compile and run the program; for the pure green color above, you should get 0 for red and blue and 255 for green. Now change the integer you use to create the color from 65280 to 115 and see what you get (can you predict?), then try it again with 2486921 (harder to predict!).

```
// ********************************************************
//   Colors.java
//
//   Draw rectangles to illustrate colors and their codes in Java
// ********************************************************

import javax.swing.JApplet;
import java.awt.*;

public class Colors extends JApplet
{
    public void paint (Graphics page)
    {
      // Declare size constants
      final int PAGE_WIDTH = 600;
      final int PAGE_HEIGHT = 400;

      // Declare variables
      int x, y;    // x and y coordinates of upper left-corner of each shape
      int width, height; // width and height of each shape

      Color myColor = new Color (200, 100, 255);

      // Set the background color and paint the screen with a white rectangle
      setBackground (Color.white);
      page.setColor(Color.white);
      page.fillRect(0, 0, PAGE_WIDTH, PAGE_HEIGHT);

      // Set the color for the rectangle
      page.setColor (myColor);

      // Assign the corner point and width and height then draw
      x = 200;
      y = 125;
      width = 200;
      height = 150;

      page.fillRect(x, y, width, height);

    }
}
```

Colors.html

```
<html>
<applet code="Colors.class" width=600 height=400>
</applet>
</html>
```