

### > Project Summary and Outcomes

The ultimate objective of this project will be to enable and enliven existing web sites—school web sites, for instance—with Java applets. Using HTML alone, it is not possible to add animation, time keeping, and other special effects to web sites. There are several vehicles available for creating and implementing these added features in web pages. One of these is the Java applet. Through this project, student programmers will become more comfortable with the interaction between Java and web pages.

This project will include a series of activities, each involving a Java applet and a corresponding web page. The first applet will display a digital clock. Once that applet is functioning, students will be able to incorporate the compiled [class] version directly into a web page. The next activity will focus on passing parameters, in this case simple colors, to a Java applet from an HTML file. The applet will supply a foreground and background color to be utilized when displaying time. The HTML file will set the desired parameters and call the applet. The third applet will use parameter passing to add a text message to a ticker tape, specifying font name, size, and colors. Students will generate the Java program to create a moving banner. Once that is working satisfactorily, they will parameterize the message, font name, size, and perhaps color, and set those parameters in a corresponding HTML file. A fourth applet will include animation. To implement this applet, students will need to know how to declare and start threads, declare and initialize an array of images, and use an ImageTracker. Once the applet is implemented, it is an easy transition to an HTML file, where parameters can be passed to control the images displayed, the location where they are displayed, and a delay time.

### > Student Assessment (see rubric)

**How well do students perform each of the following tasks:**

- Implement a Java applet to display time
- Incorporate the applets into a web page
- Employ parameter passing to make the applets more flexible

## > Prerequisite Knowledge and Skills

### Java:

- Applet construction
- Timer implementation – `javax.swing.Timer`
- Action listeners
- Parameter passing to applets
- Calendaring objects: Calendar methods have replaced many similarly named Time methods – `javax.util.Calendar`
- Defining and implementing threads using the `Runnable` interface
- Declaring and defining arrays of Images
- Using a `MediaTracker` object

## > This Project Targets the Following Subject Areas(s):

Pre-Java	Java Programming
<input type="checkbox"/> Hardware Basics	<input checked="" type="checkbox"/> Applet Programming
<input type="checkbox"/> Software Basics	<input type="checkbox"/> Subroutine Programming
<input type="checkbox"/> Networks and Servers	<input type="checkbox"/> Application Programming
<input type="checkbox"/> HTML	<input type="checkbox"/> Class Programming
<input type="checkbox"/> Action Scripting	<input checked="" type="checkbox"/> Method Programming
<input type="checkbox"/> Java Scripting	<input type="checkbox"/> Full Scale Programming

## > Project-Framing Questions

<b>Essential Question</b>	<ul style="list-style-type: none"> <li>• How does the introduction of Java applets help expand web design options</li> </ul>
<b>Activity Questions</b>	<ul style="list-style-type: none"> <li>• How are applets incorporated into web pages?</li> <li>• How is a clock implemented in Java?</li> <li>• How are parameters transferred from HTML to Java?</li> <li>• How is a process started using a thread?</li> </ul>
<b>Sample Content Questions</b>	<ul style="list-style-type: none"> <li>• What is included in the Date object? How can you access these elements?</li> <li>• Which methods of Time have now been replaced by corresponding Calendar methods and why?</li> <li>• What methods of the Runnable interface must be implemented?</li> </ul>

## > Targeted Content Standards, Benchmarks, or State Frameworks

### Massachusetts Curriculum Frameworks:

#### Geometry:

- G.G.11 – Using rectangular coordinates, calculate midpoints of segments, slopes of lines and segments, and distances between two points, and apply the results to the solutions of problems.

#### School-to-Career

##### Competencies:

- Reasoning
- Making decisions
- Thinking creatively
- Solving problems

#### International Society of Technology

##### Education (ISTE):

- TL-IV.A – Apply technology in assessing student learning of subject matter using a variety of assessment techniques.

## > Materials and Resources Required for Project

<b>Equipment</b>	<ul style="list-style-type: none"> <li>• A computer with Java installed on it</li> </ul>
<b>Consumable Supplies</b>	<ul style="list-style-type: none"> <li>• Diskettes or CDs</li> <li>• Graph paper</li> </ul>
<b>Textbooks/Lesson Guides</b>	<ul style="list-style-type: none"> <li>• <b><i>Swing, Second Edition</i></b> by Matthew Robinson, et al. Chapter 5 for treatment of Java / html parameter passing is a great example of using Java to highlight links.</li> <li>• <b><i>Core Web Programming</i></b> by Marty Hall and Larry Brown. Chapter 9 for discussion and examples of applets used in web programming</li> </ul>
<b>Technology</b>	<ul style="list-style-type: none"> <li>• <code>javax.swing.Timer</code></li> <li>• <code>javax.swing.Applet</code></li> <li>• <code>java.awt.event.ActionListener</code></li> <li>• <code>java.lang.Thread</code></li> </ul>
<b>Internet Resources</b>	<ul style="list-style-type: none"> <li>• Numerous websites on the internet show applets in action</li> </ul>

## > Activity/Lesson Plan Outline

**Overview:** Students will engage in a series of web applet activities, each of which will demonstrate the functionality that Java can add to the web development process.

**Purpose:** To expand the features available on web sites. Understanding Java applets gives the web developer one more useful tool.

**Assessment:** How well do students achieve each of the objectives? See attached rubric for details. The first objective involves using the Java Calendar class to get the current date and time and then updating and displaying that data within an applet. A related objective involves incorporating the applet into a standard HTML document. A third objective relates to the ability to include parameters within an applet. Parameters make the applet development process much more dynamic and, consequently, do the same for web development. The fourth objective entails animating the web document by means of an array of images. Finally, students should be able to parameterize the animation process to modify what is being displayed, as well as the timing of that display.

### Activity 1: Implement a Clock Using a Java Applet

Begin by creating a Java applet to continuously update the time. To do this, make use of an object of type `Calendar`. Many of the methods formerly associated with the `Time` class have now been deprecated and they should be replaced by calls to and from a `Calendar` object. To create a `Calendar` object, use the `getInstance()` method, for example, `Calendar cal = Calendar.getInstance();` You also need to use an object of type `ActionListener` to listen for `Timer` events. Each time the `Timer` fires, the `ActionListener` should generate a call to `repaint`. To generate an `ActionListener`, the applet must implement the `ActionListener` interface. The class declaration must indicate that intention:

```
public class CurrentTime extends JApplet implements
ActionListener
```

Additionally, because the `JApplet` now intends to implement `ActionListener`, it must overwrite the `actionPerformed` method, illustrated as follows. Each time the timer fires, it will generate a repaint request:

Additionally, because the `JApplet` now intends to implement `ActionListener`, it must overwrite the `actionPerformed` method, illustrated as follows. Each time the timer fires, it will generate a repaint request:

```
public void actionPerformed(ActionEvent e) {
    repaint();
}
```

You need a third object—a `Timer`—to continuously update the system time. When instantiating the timer object, you need to the firing interval in the constructor.

```
t = new javax.swing.Timer(1000, this);
```

**Tips, Hints, and Tricks:** The clock applet depends on the use of a timer. In Java, there are actually two timer classes: one is defined in `java.util`; the other in `javax.swing`. To avoid ambiguity, when instantiating a `Timer` object, use a fully qualified object definition, for instance, `javax.swing.Timer`.

Given a `Timer`  $t$  and an action listener `action`, the constructor looks as follows:

```
t = new javax.swing.Timer(1000, action);
```

## **Activity 2: Incorporate the CurrentTime Applet Into a Web Page**

Some Java compilers automatically create a web page when compiling an applet. If

yours does not, create a simple web page incorporating the .class file generated by the compiler. Note the following code:

```
<applet code="CurrentTime.class"
        width=500
        height=500
        codebase="."
        alt="Your browser understands the <APPLET> tag but
isn't running the applet, for some reason.">
    Your browser is ignoring the <APPLET> tag!
</applet>
```

You can modify width and height to fit the needs of the HTML file. Refer to the complete HTML sample following the rubric chart.

### **Activity 3: Pass Parameters**

When an applet runs, you can determine its parameters at run time by passing them into the program, either through the command line, specifying parameters in an IDE like Bluej, or loading them from an HTML file. This activity focuses on the latter option. In this activity, you can modify the time applet to set a foreground and background color using the `init()` method.

Tips, Hints, and Tricks: Because there is no conversion in Java between Strings and color information, use conditionals to convert from String parameters to actual colors. For example, you can set the background to `Color.blue` if the String background is equal to "blue". Refer to the complete HTML sample following the rubric chart.

### **Activity 4: Create an Animated Ticker Tape**

Use parameter passing in the third applet to add a text message to a ticker tape, specifying font name, size, and color. Generate the Java program to create a moving banner. Once that is working satisfactorily, parameterize the message, font name, size, and perhaps color. Then set the parameters in a corresponding HTML file. Refer to the complete HTML sample following the rubric chart. You can produce this animation with repeated calls to paint a message using a decreasing x-coordinate.

## **> Pacing/Timeline**

- Activity 1: 2 days
- Activity 2: 1 day

- Activity 3: 2 days
- HTML passing parameters to Java applet – 2 days
- Activity 4: 2 days
- Activity 5: 1 day

**> Teacher Reflection** (For example, what worked well in this project? What would you change if you were to teach it again?)

- Were the students able to make the clock work properly?
- How easy was it to add the applet code to a web page?
- How comfortable were the students with the interaction between Java and web pages?
- What other applet applications were the students interested in exploring?

## > Enhancing the School Web Site With Applets - Rubric

Attribute	Needs Improvement	Proficient	Advanced	Maximum Pts.
<i>Implements Time and Displays, and Updates in One-Second Intervals</i>	Displays time once or not at all	Displays time, which updates every second. Correctly uses Calendar class for methods. Able to display Hours, Minutes, and Seconds	Adds graphics to enhance the display	
<i>Incorporates Applet into HTML Document</i>	Cannot incorporate applet	Successfully incorporates applet	Successfully and appropriately incorporates applets into area	
<i>Adds Parameters Within HTML for Applet</i>	Cannot add parameters	Successfully adds parameters for size and text	Adds a wide range of parameters, including color and font. Includes conditionals for string to Color conversion	
<i>Implements and Displays an Array of Images</i>	Does not use an array of images	Successfully implements animation with images	Implements animation with good allowance for delay between displays of images. Does error checking on parameters	
<i>Understands and Implements Parameters for Animation</i>	Passes parameters only once or twice or not at all	Understands and implements parameters, using them to make the application as flexible as possible	Implements additional parameters by making the HTML file simply a listing of parameters and by passing the bulk of the functionality to the applet	

Sample 1**> Addendum 1: Sample Code for Building Applets and Applet-Infused Web Pages**

currentTime.java - an applet displaying the current time that refreshes every second

```
import javax.swing.JApplet;
import java.awt.Graphics;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*;
import java.text.DateFormat;

/**
 * Class CurrentTime - simple applet to display current time
 *
 * @author Ginny Lombard
 * @version 1.0
 */
public class CurrentTime extends JApplet implements ActionListener
{
    // instance variables - replace the example below with your own
    javax.swing.Timer t;
    Calendar cal;
    int width = getWidth();
    int height;
    String backgroundColor;
    String currentTime;
    DateFormat df;
    /**
     * Called by the browser or applet viewer to inform this JApplet
     that it
     * has been loaded into the system. It is always called before
     the first
     * time that the start method is called.
     */
    public void init()
    {

        width = getWidth();
        height = getHeight();

        cal = Calendar.getInstance();
        df = DateFormat.getTimeInstance();

        t = new javax.swing.Timer(1000, this);
        t.start();
```

```

    }
    public void actionPerformed(ActionEvent e) {
        repaint();
    }

    /**
     * Called by the browser or applet viewer to inform this JApplet
that it
     * should start its execution. It is called after the init
method and
     * each time the JApplet is revisited in a Web page.
     */
    public void start()
    {
        // provide any code required to run each time
        // web page is visited
    }

    public void stop()
    {
        // provide any code that needs to be run when page
        // is replaced by another page or before JApplet is
destroyed
    }

    /**
     * Paint method for applet.
     *
     * @param g    the Graphics object for this applet
     */
    public void paint(Graphics g)
    {

        g.setColor(Color.white);
        g.fillRect(0,0,width, height);
        g.setColor(Color.black);
        cal.setTime(new Date());
        Date d = cal.getTime();

        currentTime = df.format(d);

        g.drawString(currentTime, width-80, 20);

    }

}
}

```

## Sample 2

CurrentTime.html - a Web page incorporating the compiled .class file from CurrentTime

```
<html>
  <head>
    <title>CurrentTime</title>
  </head>
  <body>

    <h1>CurrentTime</h1>
    <hr>
    <applet code="CurrentTime.class"
      width=500
      height=500
      codebase="."
      alt="Your browser understands the <APPLET> tag but
isn't running the applet, for some reason."
    >

      Your browser is ignoring the <APPLET> tag!
    </applet>
    <hr>
  </body>
</html>
```

### Sample 3

HelloColor.java - a simple Java applet displaying time with a background and foreground color

```
import javax.swing.JApplet;
import java.awt.Graphics;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*;
import java.text.DateFormat;

/**
 * Write a description of the applet class HelloColor here.
 *
 * @author Ginny Lombard
 * @version 1.0
 */

public class HelloColor extends JApplet implements ActionListener
{
    javax.swing.Timer t;
    Calendar cal;
    int width;
    int height;
    String currentTime;
    Color back;
    Color fore;
```

```

DateFormat df;

/**
 * Called by the browser or applet viewer to inform this Applet that
it
 * has been loaded into the system. It is always called before the
first
 * time that the start method is called.
 */
public void init()
{
    setFont(new Font("Arial", Font.BOLD, 14));
    String background = getParameter("background");
    String foreground = getParameter("foreground");

    System.out.println(background + " " + foreground);

    back = Color.white;
    fore = Color.black;

    if (background != null)
    {
        if (background.equalsIgnoreCase("red"))
            back = Color.red;
        else if (background.equalsIgnoreCase("blue"))
            back = Color.blue;
        else if (background.equalsIgnoreCase("green"))
            back = Color.green;
    }
    if (foreground != null)
    {
        if (foreground.equalsIgnoreCase("white"))
            fore = Color.white;
        else if (foreground.equalsIgnoreCase("black"))
            fore = Color.black;
        else if (foreground.equalsIgnoreCase("gray"))
            fore = Color.gray;
    }
    setBackground(back);
    setForeground(fore);
    width = getWidth();
    height = getHeight();

    cal = Calendar.getInstance();
    df = DateFormat.getInstance();

    t = new javax.swing.Timer(1000, this);
    t.start();

}

public void start()
{
    // provide any code required to run each time

```

```

        // web page is visited
    }

    public void stop()
    {
        // provide any code that needs to be run when page
        // is replaced by another page or before JApplet is destroyed
    }

    /**
     * Paint method for applet.
     *
     * @param g    the Graphics object for this applet
     */
    public void paint(Graphics g)
    {

        g.setColor(back);
        g.fillRect(0,0,width, 40);
        g.setColor(fore);
        cal.setTime(new Date());
        Date d = cal.getTime();
        currentTime = df.format(d);

        g.drawString(currentTime, width-150, 20);

    }
    public void actionPerformed(ActionEvent e)
    {
        repaint();
    }
}

```

#### Sample 4

ColorfulWorld.html - a simple HTML page to call the color applet

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML>
<HEAD>
  <TITLE>Colorful Applet</TITLE>
</HEAD>
<BODY>
<H1>Colorful Applet</H1>
<P>
<APPLET CODE="HelloColor.class" WIDTH=400 HEIGHT=40>
  <PARAM NAME="background" VALUE="red">
  <PARAM NAME="foreground" VALUE="white">
  <B>Error! You must use a Java-enabled browser.</B>
</APPLET>
<P>
</BODY>

```

---

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java Project was developed for The Sun Foundation by Youth Tech Entrepreneurs 2004.

</HTML>

### Sample 5

TickerTape.java is a simple HTML page to run a ticker-tape message across the screen.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/**
 * TickerTape.java
 *
 * @author Ginny Lombard
 * @version 1.0
 */
public class TickerTape extends JApplet implements ActionListener
{
    private int xPos, yPos;
    private String message;
    private String sSize;
    private int fontSize;
    private String fontName;

    public void init()
    {
        Container c = getContentPane();
        c.setBackground(Color.white);
        xPos = c.getWidth();
        yPos = c.getHeight();
        message = "Hello World";
        fontName = "TimesRoman";
        sSize = "32";
        fontSize = Integer.parseInt(sSize);

        javax.swing.Timer t = new javax.swing.Timer(30, this);
        t.start();
    }
    public void actionPerformed(ActionEvent e)
    {
        repaint();
    }

    public void paint(Graphics g)
    {
        super.paint(g);
        g.setFont(new Font(fontName,Font.ITALIC, fontSize));
    }
}
```

```

        xPos--;
        if (xPos < -100)
        {
            xPos = getWidth();
        }
        yPos = getHeight() / 2;
        g.drawString(message, xPos, yPos);
    }

}

```

### Sample 6

SpinningGlove.java - an animated logo

```

import java.applet.*;
import javax.swing.*;
import java.awt.*;
import java.awt.image.*;
import java.io.*;

public class SpinningGlobe extends JApplet
    implements Runnable
{
    private boolean init = false;
    Image globe[] = null;
    Thread animation = null;
    int xLocation = 0;           // x Coordinate of drawing
    int yLocation = 20;         // y Coordinate of drawing
    int current = 0;

    /**
     * Standard initialization method for an applet
     */
    public void init()
    {
        if ( init == false )
        {
            init = true;
            globe = new Image[4];
            for (int i = 0; i < 4; i++)
            {
                globe[i] = getImage(getCodeBase(), "globe" + i + ".gif");
            }
        }
    }

    /**
     * Standard paint routine for an applet.
     * @param g contains the Graphics class to use for painting
     */
    public void paint(Graphics g)

```

```

    {
        g.drawImage(globe[current], xLocation, 20, this);
    }

    public void start()
    {
        if ( animation == null )
        {
            animation = new Thread(this);
            animation.start();
        }
    }

    public void run()
    {
        while (true)
        {
            repaint();
            current++;
            if ( current == 4 ) current = 0;
            try
            {
                Thread.currentThread().sleep(150);
            }
            catch (InterruptedException e)
            {
            }
        }
    }
}

```

### Sample 7

SpinningGlove.java - an animated logo, parameterized version

```

import java.applet.*;
import javax.swing.*;
import java.awt.*;
import java.awt.image.*;
import java.io.*;

public class SpinningGlobe2 extends JApplet
    implements Runnable
{
    private boolean init = false;
    Image myImage[] = null;
    Thread animation = null;
    int xLocation = 0;           // x Coordinate of drawing
    int yLocation = 20;         // y Coordinate of drawing
    int current = 0;
    String xLoc, yLoc;
    String filename;

```

---

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java Project was developed for The Sun Foundation by Youth Tech Entrepreneurs 2004.

```

String sImages;
int numImages = 4;

/**
 * Standard initialization method for an applet
 */
public void init()
{
    if ( init == false )
    {
        init = true;
        filename = getParameter("filename");
        if (filename == null)
            filename = "globe";
        sImages = getParameter("numImages");
        if (sImages != null)
            numImages = Integer.parseInt(sImages);
        xLoc = getParameter("xlocation");
        yLoc = getParameter("ylocation");
        if (xLoc != null)
            xLocation = Integer.parseInt(xLoc);
        if (yLoc != null)
            yLocation = Integer.parseInt(yLoc);
        tracker = new MediaTracker(this);
        myImage = new Image[numImages];
        for (int i = 0; i < numImages; i++)
        {
            myImage[i] = getImage(getCodeBase(), filename + i +
".gif"); // get image
        }
    }
}

/**
 * Standard paint routine for an applet.
 * @param g contains the Graphics class to use for painting
 */
public void paint(Graphics g)
{
    g.drawImage(myImage[current], xLocation, yLocation, this);
}

public void start()
{
    if ( animation == null )
    {
        animation = new Thread(this);
        animation.start();
    }
}

```

```
public void run()
{
    while (true)
    {
        repaint();
        current++;
        if ( current == 4 ) current = 0;
        try
        {
            Thread.currentThread().sleep(150);
        }
        catch (InterruptedException e)
        {
        }
    }
}
}
```