CSC 551: Web Programming

Spring 2004

Combining Java & JavaScript

- integrating Java with JavaScript
  - calling Java routines from JavaScript
  - controlling an applet from JavaScript
  - accessing JavaScript & HTML elements from an applet

- related topics
  - Java archives (JARs), JavaBeans

JavaScript vs. Java

recall: JavaScript is very good for simple tasks, GUI layout

- flexible data typing, primitive object types fine for quick development
- integration with HTML makes layout & control of GUI elements easy

- not much library support, only primitive data structuring capabilities
- not well-suited to multi-file projects, OO approach

recall: Java is better at complex tasks, especially graphics

- full-featured, more robust, extensive libraries of classes/routines
- can support large projects, interacting objects

- GUI layout is difficult, integration with HTML not obvious

IDEAL: make use of the the strengths of each language

- include applets in a page when needed (e.g., graphics)
- allow communication between applet & JavaScript code

Calling Java routines from JavaScript

Netscape Communicator allows direct calls to Java routines

- specify full package name of routine, then call as in Java
  - useful for more esoteric routines that are not supported directly in JavaScript
- this feature is NOT supported by Internet Explorer

```
<html>
<!--  Dave Reed    java.html    3/15/04  -->
<!--  Note: works in Netscape only.      -->

<head>
  <title>Java+JavaScript Demo</title>
</head>

<body>
  <script language="JavaScript">
    document.write(java.lang.Math.random());
  </script>
</body>
</html>
```

view page in browser

Calling applet methods

more commonly, want to include an applet in a page, control via HTML events & JavaScript

consider MontePI example from last week

- want to draw dots inside a square (with an inscribed circle)
- could build GUI interface into applet, but required tricky layout manager
- instead, leave graphics up to the applet, controlled via JavaScript

to call a Java applet method from JavaScript

```
document.appletName.methodCall(…)
```

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

public class Monte6 extends Applet
{
    private static Random randy;
    private int SIZE;
    private Image offScreenImage;
    private Graphics offScreenGraphics;

    private int randomInRange(int low, int high) {…}
    private double distance(int x1, int y1, int x2, int y2) {…}

    public void init()
    {
          randy = new Random();
          Dimension dim = getSize();
          SIZE = dim.width;

          drawCircle();
    }

    public void drawCircle() {
        // DRAWS CIRCLE ON BOTH getGraphics() AND
        // offScreenGraphics

    }

    public void drawDots(int numPoints)
    {
        // DRAWS numPoints RANDOM DOTS ON BOTH getGraphics()
        // AND offScreenGraphics
    }

    public void paint(Graphics g)
    {
          g.drawImage(offScreenImage, 0, 0, null);
    }
}
```

MontePI revisited

`init` creates the random number generator & gets applet size

`drawDots` draws the dots on the screen and to the off-screen buffer

`paint` redraws the screen using the buffer

MontePI example (cont.)

```
<html>
<!-- Dave Reed    Monte6.html    3/15/04 -->

<head>
  <title>Monte Carlo Darts Page</title>
</head>

<body bgcolor="gray">
  <div style="text-align:center">
    <applet code="Monte6.class" name="MonteApplet"
            height=300 width=300>
      You must use a Java-enabled browser to view this applet.
    </applet>

    <br /><br />

    <form name="MonteForm">
      <input type="button" value="Generate points"
        onClick="document.MonteApplet.drawDots(1000);">
    </form>
  </div>
</body>
</html>
```

here, HTML button controls the applet (via JavaScript)

view page in browser

Example (cont.)

better interface:

allow user to specify number of dots in text box

each click adds new dots, have separate button to clear

```
<html>
<!-- Dave Reed    Monte6a.html    3/15/04 -->

<head>
  <title>Monte Carlo Darts Page</title>
</head>

<body bgcolor="gray">
  <div style="text-align:center">
    <applet code="Monte6.class" name="MonteApplet"
            height=300 width=300>
      You must use a Java-enabled browser to view this applet.
    </applet>

    <br /><br />

     <form name="MonteForm">
          <input type="button" value="Generate"
                 onClick="numDots = parseFloat(document.MonteForm.numPoints.value);
                          document.MonteApplet.drawDots(numDots);">
          <input type="text" name="numPoints" size=6 value=100> points
          <br /><br />
          <input type="button" value="Clear the screen"
                 onClick="document.MonteApplet.drawCircle();">
     </form>
  </div>
</body>
</html>
```

view page in browser

Dividing control

where the control lies affects the efficiency/usability of an applet

- want the applet to be as self-contained as possible,
  take advantage of speed advantage, more advanced features
- but if GUI controls are in HTML, then JavaScript needs overall control

consider adding counters for number of dots inside & outside circle

- have the applet keep track of the dots in instance variables

1. call method to draw all dots, then JavaScript accesses counts & display
   → *fast, but only see counts when done*

2. could return more control to the page, applet draws one dot at a time
   → *repetition is handled by JavaScript, can update boxes after each dot*
   → *slower, but more flexible (and can see counts change in Netscape)*

3. could have applet update the HTML text boxes itself
   → *tricky, ties the applet to the page*

JavaScript in control

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

public class Monte7 extends Applet
{
    . . .

    public int numInside, numOutside;

    public void drawCircle()
    {
        numInside = 0;  numOutside = 0;
        . . .
    }

    public void drawDots(int numDots)
    {
        . . .
        for (int i = 0; i < numPoints; i++) {
          int x = randomInRange(0, SIZE);
          int y = randomInRange(0, SIZE);
          if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
            offScreenGraphics.setColor(Color.white);
            g.setColor(Color.white);
            numInside++;
          }
          else {
            offScreenGraphics.setColor(Color.black);
            g.setColor(Color.black);
            numOutside++;
          }
        }
        . . .
    }
    . . .
}
```

have applet keep track of number inside & out

- instance variables `numInside` and `numOutside` are initialized in `drawCircle`, updated in `drawDots`

since public, these instance variables can be accessed in the page

Example (cont.)

```
<!-- Dave Reed    Monte7.html    3/15/04 -->

<head>
  <title>Monte Carlo Darts Page</title>
  <script type="text/javascript">
    function drawAll()
    {
        var numDots = parseFloat(document.MonteForm.numPoints.value);
        document.MonteApplet.drawDots(numDots);
        document.MonteForm.numIn.value = document.MonteApplet.numInside;
        document.MonteForm.numOut.value = document.MonteApplet.numOutside;
    }

    function clearAll()
    {
        document.MonteApplet.drawCircle();
        document.MonteForm.numIn.value = 0;
        document.MonteForm.numOut.value = 0;
    }
  </script>
</head>

<body bgcolor="gray">
  <form name="MonteForm">
  <table align="center">
    <tr><td><applet code="Monte7.class" name="MonteApplet" height=300 width=300>
              You must use a Java-enabled browser to view this applet.
            </applet>
        <td><input type="button" value="Generate" onClick="drawAll();">
          <input type="text" name="numPoints" size=6 value=100> points
          <p><hr>
          <p><input type="text" name="numIn" size=6 value=0> points inside
          <p><INPUT TYPE="text" name="numOut" size=6 value=0> points outside
          <p><hr>
          <p><input type="button" value="Clear the screen" onClick="clearAll()">
  </table>
  </form>
</body>
</html>
```

Note: can utilize HTML table to achieve desired layout of elements

view page in browser

Example (cont.)

```
<!-- Dave Reed    Monte7a.html    3/15/04 -->

<head>
  <title>Monte Carlo Darts Page</title>
  <script type="text/javascript">
     function drawAll()
    {
        var numDots = parseFloat(document.MonteForm.numPoints.value);
        for (var i = 0; i < numDots; i++) {
            document.MonteApplet.drawDots(1);
            document.MonteForm.numIn.value = document.MonteApplet.numInside;
            document.MonteForm.numOut.value = document.MonteApplet.numOutside;
        }
    }

    function clearAll()
    {
        document.MonteApplet.drawCircle();
        document.MonteForm.numIn.value = 0;
        document.MonteForm.numOut.value = 0;
    }
  </script>
</head>

<body bgcolor="gray">
  <form name="MonteForm">
  <table align="center">
    <tr><td><applet code="Monte7.class" name="MonteApplet" height=300 width=300>
              You must use a Java-enabled browser to view this applet.
            </applet>
        <td><input type="button" value="Generate" onClick="drawAll();">
          <input type="text" name="numPoints" size=6 value=100> points
          <p><hr>
          <p><input type="text" name="numIn" size=6 value=0> points inside
          <p><INPUT TYPE="text" name="numOut" size=6 value=0> points outside
          <p><hr>
          <p><input type="button" value="Clear the screen" onClick="clearAll()">
  </table>
  </form>
</body>
</html>
```

Alternatively: could draw each dot individually, display counts after each dot

view page in browser

Accessing HTML/JavaScript from the applet

it is possible for the applet to access elements in the page

- requires the `JSObject` class from the `netscape.javascript` package

```
import netscape.javascript.JSObject;
```

- use `getWindow` and `getMember` methods to access components

```
JSObject jsWin = JSObject.getWindow(this);                  // GETS WINDOW
JSObject jsDoc = (JSObject) jsWin.getMember("document");    // GETS DOCUMENT

JSObject MonteForm = (JSObject) jsDoc.getMember("MonteForm");  // GETS FORM

numInside = (JSObject) MonteForm.getMember("numIn");        // GETS TEXT BOX
```

- use `getMember` and `setMember` methods to access component attributes

```
int num = Integer.parseInt( (String)numInside.getMember("value") );

numInside.setMember("value", ""+(num+1));
```

Java in control

```
import java.awt.*;
import java.applet.*;
import java.util.Random;
import netscape.javascript.JSObject; // need plugin.jar in classpath

public class Monte8 extends Applet
{
    . . .
    private JSObject numDots, numInside, numOutside;

    public void init()
    {
      . . .
      try {
        JSObject jsWin = JSObject.getWindow(this);
        JSObject jsDoc = (JSObject) jsWin.getMember("document");
        JSObject MonteForm = (JSObject) jsDoc.getMember("MonteForm");
        numDots = (JSObject) MonteForm.getMember("numDots");
        numInside = (JSObject) MonteForm.getMember("numIn");
        numOutside = (JSObject) MonteForm.getMember("numOut");
      }
      catch (netscape.javascript.JSException jse) { }
      drawCircle();
    }
    public void drawDot()
    {
      . . .
      int inCount = 0; outCount = 0;
      if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
        . . .
        inCount++;
      }
      else {
        . . .
        outCount++;
        }
    }
    int newIn = Integer.parseInt((String)numInside.getMember("value"))+ inCount;
    numInside.setMember("value", ""+newIn);
    int newOut = Integer.parseInt((String)numOutside.getMember("value"))+ outCount;
    numOutside.setMember("value", ""+newOut);
}
```

```
public void drawCircle()
{
  numInside.setMember("value", "0");
  numOutside.setMember("value", "0");
  . . .
}
```


Example (cont.)

```
<html>
<!-- Dave Reed    Monte8.html    3/15/04 -->

<head>
  <title>Monte Carlo Darts Page</title>
</head>

<body bgcolor="gray">
  <form name="MonteForm">
  <table align="center">
    <tr>
      <td><applet code="Monte9.class" name="MonteApplet"
                  height=300 width=300 mayscript>
          You must use a Java-enabled browser to view this applet.
          </applet>
      <td><input type="button" value="Generate"
                 onClick="document.MonteApplet.drawDots();">
        <input type="text" name="numDots" size=6 value=100> points
        <br /><br />
        <hr>
        <br /><br />
        <input type="text" name="numIn" size=6 value=0> points inside
        <br /><br />
        <input type="text" name="numOut" size=6 value=0> points outside
        <br /><br />
        <hr>
        <br /><br />
        <input type="button" value="Clear the screen"
               onClick="document.MonteApplet.drawCircle();">
  </table>
  </form>
</body>
</html>
```

MAYSCRIPT attribute must be specified in the APPLET tag to allow access to HTML & JavaScript in the page

view page in browser

Non-graphical example

```
import java.applet.Applet;
import java.awt.*;
import java.net.*;
// This appears in Core Web Programming from
// Prentice Hall Publishers, and may be freely used
// or adapted. 1997 Marty Hall, hall@apl.jhu.edu.

public class GetHost extends Applet {
  private String host;

  public void init() {
    setBackground(Color.white);
    try {
      host = InetAddress.getLocalHost().toString();
    } catch(UnknownHostException uhe) {
      host = "Unknown Host";
    }
  }

  public String getHost() {
    return(host);
  }
}
```

uses the Java `InetAddress` class to get the client's host name

returns via `getHost` method

Example (cont.)

```
<html>
<!-- resume.html: based on an example in Core Web Programming, 1997 -->
<!-- Original author: Marty Hall (hall@apl.jhu.edu.)                -->

<head>
  <title>WonderWidget</title>

  <script type="text/javascript">
  function showResume()
  // Results: sets location of page based on user's host name
  {
    if ((document.gethost.getHost()).indexOf("widgets-r-us.com") != -1) {
      location = "ResumeLoyal.html";
    }
    else {
      location = "ResumeReal.html";
    }
  }
  </script>
</head>

<body bgcolor="white">
<h1>WonderWidget</h1>

<applet code="GetHost.class" width=10 height=10 name="gethost">
</applet>

Description:
<ul>
  <li>Name: Wonder Widget
  <li>Serial Number: 1544X
  <li>Cost: $7.95 (plus 22.50 shipping and handling)
  <li>Designer:  <a href="javascript:showResume();">
                 J. Random Hacker</a>
</body>
</hmtl>
```

applet provides access to `getHost` method

here, the link is conditional based on the host name

view page in browser

Related topics

JAR files

- for applets that are comprised of multiple classes, can bundle all necessary files into a Java Archive (JAR) file
- uses the popular ZIP file format
- download using ARCHIVES attribute, automatically unzipped by browser

JavaBeans

- reusable components (e.g., buttons, menus) that can be packaged and reused
- requires special tools for compiling and packaging (e.g., BDK)
- downloaded with an applet using the `ARCHIVES` attribute

```
<applet code="javaApp.class" archives="jarfile.jar">
```