

CSC 551: Web Programming

Fall 2001

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
 - JavaBeans, Java archives (JARs)

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   11/1/01  -->
<!-- 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(...)
```

MontePI example revisited

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

public class Monte7 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;
    }

    public void drawDots(int numPoints)
    {
        // DRAWS TO BOTH getGraphics() AND offScreenGraphics
    }

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

init creates the random number generator & get 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   Monte7.html   11/1/01 -->

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

<body bgcolor="gray">
<div align="center">
  <applet code="Monte7.class" name="MonteApplet"
          height=300 width=300>
    You must use a Java-enabled browser to view this applet.
  </applet>
  <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

[view page in browser](#)

Adding features

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

public class Monte8 extends Applet
{
    . . .

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

        clear();
    }

    public void clear()
    {
        // CLEARS SCREEN AND DRAWS RED CIRCLE
    }

    public void drawDots(int numPoints)
    {
        // DRAWS TO BOTH getGraphics() AND offScreenGraphics
    }

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

can add a finer granularity to the applet to allow greater control from the Web page

Example (cont.)

```
<html>
<!-- Dave Reed Monte8.html 11/1/01 -->

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

<body bgcolor="gray">
<div align="center">
<applet code="Monte8.class" name="MonteApplet"
height=300 width=300>
You must use a Java-enabled browser to view this applet.
</applet>

<p>
<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
<p>
<input type="button" value="Clear the screen" onClick="document.MonteApplet.clear();">
</form>
</div>
</body>
</html>
```

allow user to specify number of dots in text box

each click adds new dots, have separate button to clear

[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. after drawing all dots, JavaScript could access counts & display
→ *can't see counts in progress (in Netscape)*
 2. could have applet update the HTML text boxes itself
→ *tricky (example later)*
 3. 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*

JavaScript in control

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

public class Monte9 extends Applet
{
    . . .
    public int numInside, numOutside;
    . . .

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

    public void drawDot()
    {
        . . .
        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++;
        }
        . . .
    }
    . . .
}
```

change applet so that method only draws a single dot (repetition to be controlled by JavaScript)

have applet keep track of number inside & out (can access & display with JavaScript)

Example (cont.)

```
<html>
<!-- Dave Reed   Monte9.html   11/1/01 -->
<head>
<title>Monte Carlo Darts Page</title>
<script language="JavaScript">
  function doAll()
  {
    var numDots = parseFloat(document.MonteForm.numPoints.value);
    for (var i = 0; i < numDots; i++) {
      document.MonteApplet.drawDot();
      document.MonteForm.numIn.value = document.MonteApplet.numInside;
      document.MonteForm.numOut.value = document.MonteApplet.numOutside;
    }
  }

  function clearAll()
  {
    document.MonteApplet.clear();
    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="Monte9.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="doAll();">
<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()">
</td>
</tr>
</table>
</form>
</body>
</html>
```

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

[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;

public class Montel0 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) { }
        clear();
    }
    public void drawDot()
    {
        . . .
        if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
            int num = Integer.parseInt((String)numInside.getMember("value"));
            numInside.setMember("value", ""+(num+1));
        }
        else {
            . . .
            int num = Integer.parseInt((String)numOutside.getMember("value"));
            numOutside.setMember("value", ""+(num+1));
        }
        . . .
    }
}

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

Example (cont.)

```
<html>
<!-- Dave Reed   Montel0.html   11/1/01  -->

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

<body bgcolor="gray">
<form name="MonteForm">
<table align="center">
<tr>
<td><applet code="Montel0.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
<p>
<hr>
<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>
<input type="button" value="Clear the screen"
onClick="document.MonteApplet.clear();">
</td>
</tr>
</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>
<!-- This appears in Core Web Programming from -->
<!-- Prentice Hall Publishers, and may be freely used -->
<!-- or adapted. 1997 Marty Hall, hall@apl.jhu.edu. -->

<head>
<title>WonderWidget</title>

<script language="JavaScript">
function showResume() {
    if ((document.getHost().getHost()).indexOf("widgets-r-us.com") != -1)
        location = "ResumeLoyal.html";
    else
        location = "ResumeReal.html";
    return(false);
}
</script>
</head>

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

<applet code="GetHost" 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="ResumeLoyal.html"
onClic="return(showResume())">
J. Random Hacker</a>
</ul>
</body>
</html>
```

applet provides
access to getHost
method

here, the link is
conditional based
on the host name

[view page in browser](#)

Related topics

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">
```

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

Next week...

Server-side programming via CGI

- server-side vs. client-side
- Common Gateway Interface
 - CGI via C++

read Chapters 24 & 25

as always, be prepared for a quiz on

- today's lecture (moderately thorough)
- the reading (superficial)