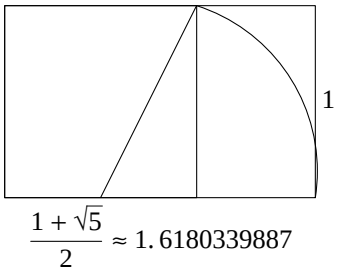
Fall 2004 Handout 11

The golden ratio ϕ (the Greek lowercase letter phi)

See Edward R. Tufte, *The Visual Display of Quantitative Information* (Graphics Press, 1983), pp. 186−190; and Davis Bergamini, *Mathematics* (Time-Life Books, 1970), pp. 94−97.



$$\frac{1+\sqrt{5}}{2} \approx 1.6180339887$$

For **&{};** entities, see Flanagan pp. 166−167; Wagner p. 41. They can be used only within an attribute of an HTML tag, e.g., the **WIDTH** attribute of the **IMG** tag.

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>The golden ratio</TITLE>
 4 <META http-equiv = "Content-Script-Type" content = "text/javascript">
 5 </HEAD>
 6 <BODY>
 7 <H1>The golden ratio</H1>
 8
 9 <SCRIPT TYPE = "text/javascript">
10 var height = 100;                  //height of rectangle, in pixels
11 var phi = (1 + Math.sqrt(5)) / 2;  //the golden ratio, approx 1.61803
12 </SCRIPT>
13
14 <TABLE>
15
16 <TR>
17 <TD>
18 <IMG SRC = "gold.gif" WIDTH = &{Math.round(height * phi)}; HEIGHT = &{height};>
19 </TD>
```



```
20 <TD>1</TD>
21 </TR>
22
23 <TR>
24 <TD ALIGN = CENTER>
25 <SCRIPT TYPE = "text/javascript">document.write(phi);</SCRIPT>
26 </TD>
27 </TR>
28 </SCRIPT>
29
30 </TABLE>
31 </BODY>
32 </HTML>
```

The raw output is a 2×2 table:

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>The golden ratio</TITLE>
 4 </HEAD>
 5 <BODY>
 6 <H1>The golden ratio</H1>
 7 <TABLE>
 8
 9 <TR>
10 <TD><IMG SRC = "gold.gif" WIDTH = 162 HEIGHT = 100></TD>
11 <TD>1</TD>
12 </TR>
13
14 <TR>
15 <TD ALIGN = CENTER>1.618033988749895</TD>
16 </TR>
17
18 </TABLE>
19 </BODY>
20 </HTML>
```

Draw an image/x-xbitmap flag

I had to write the number 8 in many places in the program. Sometimes it stood for the number of bits in a byte; sometimes it stood for the number of pixels in the height of a horizontal stripe. To make it clear which 8 stood for what, I created the two variables in lines 9 and 10 and wrote whichever one was appropriate instead of 8.

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>An image/x-xbitmap flag</TITLE>
 4 </HEAD>
 5 <BODY>
 6 <H1>An image/x-xbitmap flag</H1>
 7 <SCRIPT TYPE = "text/javascript">
 8
 9 var bits_per_byte = 8;
10 var bits_per_stripe = 8;            //height of each horizontal stripe, in bits
11 var height = 13 * bits_per_stripe;  //thirteen colonies
12 var width = 2 * height;             //must be multiple of bits_per_byte
```

```
13
14 var w2 = window.open("", "window2",
15     "innerWidth=" + (width + 16) + ",innerHeight=" + (height + 16));
16 w2.document.open("image/x-xbitmap");
17
18 w2.document.write(
19     "#define flag_width ", width, "\n",
20     "#define flag_height ", height, "\n",
21     "static unsigned char flag_bits[] = {\n"
22 );
23
24 for (var row = 0; row < height; row = row + 1) {
25     for (var col = 0; col < width / bits_per_byte; col = col + 1) {
26         if (row < height / 2 && col < width / (2 * bits_per_byte)) {
27             //Black union jack in upper left corner.
28             w2.document.write("0xFF, ");
29         } else if (row % (2 * bits_per_stripe) < bits_per_stripe) {
30             //black stripe
31             w2.document.write("0xFF, ");
32         } else {
33             //white stripe
34             w2.document.write("0x00, ");
35         }
36     }
37
38     //at end of each row
39     w2.document.write("\n");
40 }
41
42 w2.document.write("};\n");
43 w2.document.close();
44
45 </SCRIPT>
46 </BODY>
47 </HTML>
```

To see the raw output of rows and columns of hexadecimal numbers for debugging, change the **"image/x-xbitmap"** in line 16 to **"text/plain"**:

```
1 #define flag_width 208
2 #define flag_height 104
3 static unsigned char flag_bits[] = {
4 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, etc.
5 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, etc.
6 etc.
```

Sort an array:
Flanagan pp. 89−90, 127−128, 342−343; Wagner pp. 600, 829

** ** is the HTML ‘‘non-breaking space’’. It prevents empty table cells from disappearing.

```
1 <HTML>
2 <HEAD>
3 <TITLE>Sort the MIME types</TITLE>
4 </HEAD>
5 <BODY>
```

```
 6 <H1>Sort the MIME types</H1>
 7 <TABLE BORDER>
 8 <TH> </TH>
 9 <TH>type</TH>
10 <TH>description</TH>
11 <TH>suffixes</TH>
12 <TH>enabledPlugin</TH>
13
14 <SCRIPT TYPE = "text/javascript">
15
16 function compare(a, b)
17 {
18     var aType = a.type.toLowerCase();
19     var bType = b.type.toLowerCase();
20
21     if (aType < bType) {
22         return -1;   //a should come before b
23     } else if (aType > bType) {
24         return 1;    //b should come before a
25     } else {
26         return 0;    //doesn't matter which comes first
27     }
28 }
29
30 //Make a copy of the array navigator.mimeTypes.
31 var mt = new Array(navigator.mimeTypes.length);
32 for (var i = 0; i < navigator.mimeTypes.length; i = i + 1) {
33     mt[i] = navigator.mimeTypes[i];
34 }
35
36 mt.sort(compare);
37
38 for (var i = 0; i < navigator.mimeTypes.length; i = i + 1) {
39     var m = mt[i];
40     document.write(
41         "<TR>",
42         "<TD ALIGN = RIGHT>", i + 1,                                    "</TD>",
43         "<TD>", m.type,                                                 "</TD>",
44         "<TD>", m.description   == ""   ? " " : m.description,     "</TD>",
45         "<TD>", m.suffixes      == ""   ? " " : m.suffixes,        "</TD>",
46         "<TD>", m.enabledPlugin == null ? " " : m.enabledPlugin.name, "</TD>",
47         "</TR>"
48     );
49 }
50 </SCRIPT>
51
52 </TABLE>
53 </BODY>
54 </HTML>
```

Without the **?:** operator (Flanagan pp. 63−64, 136−137; Wagner pp. 126−127), we'd have to write lines 40−48 as

```
 1     document.write(
 2         "<TR>",
```



```
 3          "<TD ALIGN = RIGHT>", i + 1, "</TD>",
 4          "<TD>", m.type, "</TD>",
 5          "<TD>"
 6      );
 7
 8      if (m.description == "") {
 9          document.write(" ");
10      } else {
11          document.write(m.description);
12      }
13
14      document.write("</TD><TD>");
15
16      if (m.suffixes == "") {
17          document.write(" ");
18      } else {
19          document.write(m.suffixes);
20      }
21
22      document.write("</TD><TD>");
23
24      if (m.enabledPlugin == null) {
25          document.write(" ");
26      } else {
27          document.write(m.enabledPlugin.name);
28      }
29
30      document.write("</TD></TR>");
31  );
```

With the **?:** operator, we can write lines 21−27 as

```
return aType < bType ? -1 : aType > bType;
```

since the numerical value of the subexpression **aType > bType** will be 1 for true, 0 for false (Flanagan p. 136; Wagner p. 98).

Call a method of a Java applet:
Flanagan p. 297; Wagner p. 661

Java can draw a picture much faster than JavaScript can draw a **image/x-xbitmap**.

The **paint** method that the Java class **Applet** inherits from the Java class **Component** paints only the applet's foreground. The **repaint** method that the Java class **Applet** inherits from the Java class **Component** paints the applet's background and foreground.

NEXT TIME: show them what would happen if the JavaScript event handler accidentally called **paint** instead of **repaint.**

NEXT TIME: Let the name of the JavaScript applet object be the same as the name of the Java class.

```
<APPLET NAME "Disk">
```

This file is **Disk.java**:

```
 1 import java.applet.*;   //Applet Package
 2 import java.awt.*;      //Abstract Windowing Toolkit Package
 3
 4 public class Disk extends Applet {
 5     public int radius;   //in pixels
```

```
 6
 7      public void init()
 8      {
 9          radius = 25;
10      }
11
12      public void paint (Graphics g)
13      {
14          g.fillArc(0, 0, 2 * radius, 2 * radius, 0, 360);
15          g.drawString("radius == " + String.valueOf(radius),
16              2 * radius + 10, radius);
17      }
18  }
```

```
 1  <HTML>
 2  <HEAD>
 3  <TITLE>Call a method of a Java applet</title>
 4  </HEAD>
 5  <BODY>
 6  <H1>Call a method of a Java applet</H1>
 7  <HR>
 8
 9  <APPLET NAME = "disk" CODE = "Disk.class" width = 150 height = 100>
10  This browser does not understand the APPLET tag and does not support Java.
11  </APPLET>
12
13  <HR>
14  <FORM NAME = "myform">
15  Input the radius in pixels:
16  <INPUT TYPE = TEXT NAME = "r" SIZE = 5>
17  <P>
18  <INPUT TYPE = SUBMIT VALUE = "Call the method." onClick = "
19      var i = parseInt(document.myform.r.value);
20      if (isNaN(i)) {
21          alert (’\’’ + document.myform.r.value + ’\’ is not a number.’);
22      } else {
23          document.disk.radius = i;
24          document.disk.repaint();
25          reset();
26          return false;    //Don’t go anywhere.
27      }
28  ">
29
30  <INPUT TYPE = RESET VALUE = "Start again." onClick = "
31      document.disk.radius = 25;
32      document.disk.repaint();
33      return true; //Reset the TEXT box.
34  ">
35  </FORM>
36  <HR>
37  </body>
38  </html>
```

A document with three layers:
Wagner pp. 445−459

Layers will eliminate the need for the ‘‘single-pixel GIF trick’’, described in the first edition of *Creating Killer Websites*, **http://www.killersites.com**.

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>A document with three layers</TITLE>
 4 <SCRIPT TYPE = "text/javascript">
 5
 6 function f()
 7 {
 8     defaultStatus =
 9         "document.layer1: " +
10         "left == " + document.layer1.left + ", " +
11         "top == " + document.layer1.top + ", " +
12         "visibility == " + document.layer1.visibility
13 }
14
15 </SCRIPT>
16 </HEAD>
17
18 <BODY onLoad = "f();">
19 <H1>A document with three layers</H1>
20
21 <LAYER NAME = "layer0" BGCOLOR = RED
22     TOP = 50 LEFT = 25 WIDTH = 162 HEIGHT = 100>
23 <H1>Layer 0</H1>
24 This is layer 0.
25 </LAYER>
26
27 <LAYER NAME = "layer1" BGCOLOR = ORANGE
28     TOP = 100 LEFT = 87 WIDTH = 162 HEIGHT = 100>
29 <BR>
30 <FORM>
31 <INPUT TYPE = CHECKBOX>Check here in layer 1.
32 </FORM>
33 </LAYER>
34
35 <LAYER NAME = "layer2" BGCOLOR = YELLOW
36     TOP = 150 LEFT = 149 WIDTH = 162 HEIGHT = 100>
37 <IMG SRC = "ans210.jpg" ALIGN = CENTER>Layer 2
38 </LAYER>
39
40 </BODY>
41 </HTML>
```

Make a layer pop up and pop down

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>Are you currently taking any medication?</TITLE>
 4 <SCRIPT TYPE = "text/javascript">
 5
 6 function hideMiddle()
```

```
 7 {
 8     document.middleLayer.visibility = "hide";
 9     document.bottomLayer.top = document.topLayer.clip.height;
10 }
11
12 function showMiddle()
13 {
14     document.bottomLayer.top = document.topLayer.clip.height +
15         document.middleLayer.clip.height;
16     document.middleLayer.visibility = "inherit";
17 }
18 </SCRIPT>
19 </HEAD>
20 <BODY onLoad = "
21     document.middleLayer.top = document.topLayer.clip.height;
22     document.bottomLayer.top = document.topLayer.clip.height;
23     document.bottomLayer.visibility = 'inherit';
24 ">
25
26 <LAYER NAME = "topLayer" LEFT = 0 TOP = 0>
27 <HR>
28 <FORM NAME = "topForm">
29 <H2>Medical history</H2>
30 <INPUT TYPE = CHECKBOX NAME = "medicated" onClick = "
31     if (this.checked) {
32         parentLayer.showMiddle();
33     } else {
34         parentLayer.hideMiddle();
35     }
36 ">Are you currently taking any medication?
37 </FORM>
38 </LAYER>
39
40 <LAYER NAME = "middleLayer" LEFT = 0 VISIBILITY = HIDE>
41 <FORM NAME = "middleForm">
42 <P>
43 Please describe it:
44 <BR>
45 <TEXTAREA NAME = "medication" ROWS = 4 COLS = 40>
46 </TEXTAREA>
47 </FORM>
48 </LAYER>
49
50 <LAYER NAME = "bottomLayer" LEFT = 0 VISIBILITY = HIDE>
51 <FORM NAME = "bottomForm" METHOD = POST
52     ACTION = "/cgi-bin/cgiwrap/~mm64/medication">
53
54 <INPUT TYPE = HIDDEN NAME = "medicated">
55 <INPUT TYPE = HIDDEN NAME = "medication">
56 <P>
57 <INPUT TYPE = SUBMIT VALUE = "Submit history." onClick = "
58     var d = parentLayer.document;
59     var f = this.form;
60
```

```
61      //Copy the data from the two other forms into the hidden elements
62      //of this form.
63
64      if (d.topLayer.document.topForm.medicated.checked) {
65          f.medicated.value = 'on';
66          f.medication.value = d.middleLayer.document.middleForm.medication.value;
67      } else {
68          f.medicated.value = '';
69          f.medication.value = '';
70      }
71
72      return true;
73  ">
74
75  <INPUT TYPE = RESET VALUE = "Start again." onClick = "
76      parentLayer.hideMiddle();
77      parentLayer.document.topLayer.document.topForm.reset();
78      parentLayer.document.middleLayer.document.middleForm.reset();
79      return true;
80  ">
81  </FORM>
82  <HR>
83  </LAYER>
84
85  </BODY>
86  </HTML>
```

bottomForm runs the same gateway **medication** as the form in Handout 10, pp. 4−6:

```
\&#!/bin/perl
\&require 'cgi-lib.pl';
\&ReadParse();
\&
\&print 'Content-type: text/html
\&
\&<HTML>
\&<HEAD>
\&<TITLE>Confirmation of medication</TITLE>
\&</HEAD>
\&<BODY>
\&<H1>Confirmation of medication</H1>
\&';
\&
\&print $in{medicated} ? $in{medication} : 'No medication.';
\&
\&print '
\&</BODY>
\&</HTML>
\&';
\&
\&exit 0;
```


Move a layer:
Wagner pp. 452−459

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>Move a layer</TITLE>
 4 <SCRIPT TYPE = "text/javascript">
 5
 6 //Radius and coordinates of center of the orbit.
 7 var radius = 120;
 8 var sunX = radius;
 9 var sunY = 90;
10
11 //Current position along orbit: 0 <= degrees < 360.
12 var degrees = 0;
13
14 function f()
15 {
16     document.earth.visibility = 'show';
17
18     if (degrees < 180) {
19         document.earth.moveBelow(document.sun);
20     } else {
21         document.earth.moveAbove(document.sun);
22     }
23
24     if (degrees < 90) {
25         status = "La Primavera " + degrees;
26     } else if (degrees < 180) {
27         status =  "L'Estate " + degrees;
28     } else if (degrees < 270) {
29         status = "L'Autunno " + degrees;
30     } else {
31         status = "L'Inverno " + degrees;
32     }
33
34     var radians = degrees * Math.PI / 180;
35     document.earth.moveTo(
36         Math.round(sunX + radius * Math.cos(radians)),
37         Math.round(sunY - radius * Math.sin(radians) / 5)
38     );
39
40     degrees = degrees + 2;
41     if (degrees >= 360) {
42         degrees = degrees - 360;
43     }
44 }
45
46 </SCRIPT>
47 </HEAD>
48
49 <BODY BGCOLOR = BLACK TEXT = WHITE LINK = YELLOW VLINK = ORANGE onLoad = "
50     setInterval('f();', 80);
51 ">
52
```

```
53 <H1>Move a layer</H1>
54 JavaScript
55 <A HREF = "">X52.9755</A>
56 salutes
57 <A HREF = "http://www.srv.net/~roxtar/valli_frankie.html">Frankie Valley
58 and the fabulous Four Seasons</A>!
59
60 <LAYER NAME = "sun" LEFT = 120 TOP = 90>
61     <IMG SRC = "sun.gif">
62     <LAYER LEFT = 15 TOP = 20>
63         <BODY TEXT = BLACK>Sun</BODY>
64     </LAYER>
65 </LAYER>
66
67 <LAYER NAME = "earth" VISIBILITY = HIDE>
68     <IMG SRC = "earth.gif">
69     <LAYER LEFT = 15 TOP = 20>
70         <BODY TEXT = BLACK>Earth</BODY>
71     </LAYER>
72 </LAYER>
73
74 </BODY>
75 </HTML>
```

If we had added this array of four strings at line 13,

```
var season = new Array("La Primavera", "L'Estate", "L'Autunno", "L'Inverno");
```

then we could have reduced the nine lines 24−32 to the single line

```
status = season[Math.floor(degrees / 90)] + " " + degrees;
```

Line 50 may be written

```
 1     setInterval(f, 80);
```

Create a class, and five objects of that class

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>Create a class, and five objects of that class</TITLE>
 4 <SCRIPT TYPE = "text/javascript">
 5
 6 //The move method of class Rabbit allows them to move only within a
 7 //golden rectangle.
 8
 9 function Rabbit_move()
10 {
11     var dx = 10 * (random(3) - 1);   //either +10, 0, or -10
12     var dy = 10 * (random(3) - 1);   //either +10, 0, or -10
13
14     var sum = this.layer.left + dx;
15     if (sum < 0 || sum > 200 * 1.61803) {
16         dx = -dx;
17     }
18
19     sum = this.layer.top + dy;
```

```
20      if (sum < 0 || sum > 200) {
21          dy = -dy;
22      }
23
24      this.layer.offset(dx, dy);
25  }
26
27  //constructor function for class Rabbit:
28  function Rabbit(layer, x, y)
29  {
30      this.move = Rabbit_move;
31
32      this.layer = layer;
33      this.layer.moveTo(x, y);
34      this.layer.visibility = "inherit";
35  }
36
37  //Return a random integer in the range 0 to n - 1 inclusive:
38  function random(n)
39  {
40      return Math.floor(n * Math.random());
41  }
42
43  function moveAll()
44  {
45         redRabbit.move();
46      orangeRabbit.move();
47      yellowRabbit.move();
48       greenRabbit.move();
49        blueRabbit.move();
50
51      status = "redRabbit's position: " +
52          redRabbit.layer.left + ", " +
53          redRabbit.layer.top;
54  }
55
56  function init()
57  {
58      //Create new Rabbits.
59      redRabbit    = new Rabbit(document.redLayer,    100, 100);
60      orangeRabbit = new Rabbit(document.orangeLayer, 100, 100);
61      yellowRabbit = new Rabbit(document.yellowLayer, 100, 100);
62      greenRabbit  = new Rabbit(document.greenLayer,  100, 100);
63      blueRabbit   = new Rabbit(document.blueLayer,   100, 100);
64
65      setInterval(moveAll, 200);
66  }
67  </SCRIPT>
68  </HEAD>
69
70  <BODY onLoad = "init();">
71  <H1>Create a class, and five objects of that class</H1>
72
73  <LAYER WIDTH = 20 HEIGHT = 20 VISIBILITY = HIDDEN
```



```
74      NAME = "redLayer" BGCOLOR = RED>
75  </LAYER>
76
77  <LAYER WIDTH = 20 HEIGHT = 20 VISIBILITY = HIDDEN
78      NAME = "orangeLayer" BGCOLOR = ORANGE>
79  </LAYER>
80
81  <LAYER WIDTH = 20 HEIGHT = 20 VISIBILITY = HIDDEN
82      NAME = "yellowLayer" BGCOLOR = YELLOW>
83  </LAYER>
84
85  <LAYER WIDTH = 20 HEIGHT = 20 VISIBILITY = HIDDEN
86      NAME = "greenLayer" BGCOLOR = GREEN>
87  </LAYER>
88
89  <LAYER WIDTH = 20 HEIGHT = 20 VISIBILITY = HIDDEN
90      NAME = "blueLayer" BGCOLOR = BLUE>
91  </LAYER>
92
93  </BODY>
94  </HTML>
```

Create a class, and an array of objects of that class

A variable that will be used only in one function should be declared in that function (line 16). A variable that will be used in more than one function should be declared above the functions (lines 7−9).

```
 1  <HTML>
 2  <HEAD>
 3  <TITLE>Create a class, and an array of objects of that class</TITLE>
 4  <SCRIPT TYPE = "text/javascript">
 5
 6  //The background color of each Rabbit:
 7  var colors = new Array ("red", "orange", "yellow", "green", "blue");
 8  var nRabbits = colors.length;   //number of Rabbits
 9  var a;                          //array of Rabbits
10
11  //The move method of class Rabbit allows them to move only within a
12  //golden rectangle.
13
14  function Rabbit_move()
15  {
16      var dx = 10 * (random(3) - 1);    //either +10, 0, or -10
17      var dy = 10 * (random(3) - 1);    //either +10, 0, or -10
18
19      var sum = this.layer.left + dx;
20      if (sum < 0 || sum > 200 * 1.61803) {
21          dx = -dx;
22      }
23
24      sum = this.layer.top + dy;
25      if (sum < 0 || sum > 200) {
26          dy = -dy;
27      }
28
```

```
29      this.layer.offset(dx, dy);
30  }
31
32  //constructor function for class Rabbit:
33  function Rabbit(layer, color, x, y)
34  {
35      this.move = Rabbit_move;
36
37      this.layer = layer;
38      this.layer.moveTo(x, y);
39      this.layer.bgColor = color;
40      this.layer.visibility = "inherit";
41  }
42
43  //Return a random integer in the range 0 to n - 1 inclusive:
44  function random(n)
45  {
46      return Math.floor(n * Math.random());
47  }
48
49  function moveAll()
50  {
51      for (var i = 0; i < nRabbits; i = i + 1) {
52          a[i].move();
53      }
54
55      status = "a[0]'s position: " + a[0].layer.left + ", " + a[0].layer.top;
56  }
57
58  function init()
59  {
60      //Create array of new Rabbits.
61      a = new Array(nRabbits);
62      for (var i = 0; i < nRabbits; i = i + 1) {
63          a[i] = new Rabbit(document.layers[i], colors[i], 100, 100);
64      }
65
66      setInterval(moveAll, 200);
67  }
68  </SCRIPT>
69  </HEAD>
70
71  <BODY onLoad = "init();">
72  <H1>Create a class, and an array of objects of that class</H1>
73
74  <SCRIPT TYPE = "text/javascript">
75  for (var i = 0; i < nRabbits; i = i + 1) {
76      document.write(
77          "<LAYER WIDTH = 20 HEIGHT = 20 VISIBILITY = HIDDEN></LAYER>"
78      );
79  }
80  </SCRIPT>
81
82  </BODY>
```



```
83 </HTML>
```

Unnecessarily complicated function

Here's a simpler version of the function **`moveBall`** that does exactly the same thing as the one in Wagner pp. 455 and 458. The greatest (and shortest) book on structured programming is

`http://mcgraw-hill.inforonics.com/cgi/getarec?mgh3370`

```
 1 function moveBall(ball, offsetX, offsetY)
 2 {
 3     if (ball.left <= 0 || ball.right >= 400) {
 4         return "leftBoundary";
 5     }
 6
 7     if (ball.top <= 0 || ball.top >= 200) {
 8         return "topBoundary";
 9     }
10
11     ball.offset(offsetX, offsetY);
12     return "ok";
13 }
```

JavaScript Party
— after the last class —

∫∫

The Embassy of Westphalia
99 East Fourth Street, between First & Second Avenues
Apartment 6A
(212) 674−1154
mark.meretzky@i5.nyu.edu

□

