Fall 2004 Handout 10

Change the gross anatomy of a document

The JavaScript programs that we've made so far leave the gross anatomy of the document suspiciously unchanged:

(1) We can **`document.write`** into the document, *but only as the document is loaded* (i.e., inside a **`<SCRIPT>`**). We can't do an additional **`document.write`** later (i.e., inside an event handler).

(2) We can change the **`document.bgColor`** property (Flanagan pp. 219−220; Wagner pp. 320−326).

(3) We can change the **`status`** property of the window that displays the document (Flanagan pp. 200−203; Wagner pp. 264−266).

(4) We can change the **`src`** property of an **`<IMG>`** in the document, but we can't change its width or height (Flanagan pp. 239−241; Wagner pp. 335, 429−437).

(5) We can change the **`checked`** property of a **`CHECKBOX`** or **`RADIO`** tag, or the **`value`** property of a **`TEXT`** or **`TEXTAREA`** tag. But we can't change their width or height. (For **`CHECKBOX`** and **`RADIO`**, see Handout 8, p. 10, line 37 and Handout 9, p. 9, line 12. For **`TEXT`**, see Wagner pp. 426−429, where a scrolling marquee is displayed in a **`TEXT`** instead of in the status line as in Handout 9, p. 8.)

A document that tries (unsuccessfully) to add to itself:
Flanagan p. 221; Wagner pp. 317−320

Without the surrounding **`<FORM>`** and **`</FORM>`** in lines 8 and 15, the **`CHECKBOX`** in lines 9−14 will not appear. Without the **`return true;`** in line 13, the check does not appear in the checkbox when you click.

Unfortunately, the following **`document.write`** will overwrite the entire document. The **`document.open`** in line 10 is not necessary—even without it, the first **`document.write`** would automatically open the document. The argument **`'text/html'`** in line 10 is not necessary—it's the default (Flanagan pp. 223, 381; Wagner p. 313). The **`document.close`** in line 12, however, is necessary because the output of **`document.write`** is buffered. Without the **`document.close`**, the output of **`document.write`** might not be flushed out to the document.

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>A document that tries (unsuccessfully) to add to itself</TITLE>
 4 </HEAD>
 5 <BODY>
 6 <H1>A document that tries (unsuccessfully) to add to itself</H1>
 7
 8 <FORM>
 9 <INPUT TYPE = CHECKBOX onClick = "
10     document.open('text/html');
11     document.write('One more line.');
12     document.close();
13     return true;
14 ">Click here to add one more line to this document.
15 </FORM>
16
```

```
17 </BODY>
18 </HTML>
```

One frame of a window can write into another frame:
Flanagan pp. 203−207; Wagner pp. 281−282

Without the `NAME = "frame1"`, in line 8, we'd have to change the `window.frame1.f();` in line 6 to `window.frames[1].f();` (Flanagan p. 179; Wagner pp. 289−290).

Without the `SRC = "emptyframe.html"` in line 7, the browser wouldn't add a `frame0` property to the current window.

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>One frame of a window can write into another frame</TITLE>
 4 </HEAD>
 5
 6 <FRAMESET ROWS = "*,*" onLoad = "window.frame1.f();">
 7     <FRAME SRC = "emptyframe.html" NAME = "frame0">
 8     <FRAME SRC =     "frame1.html" NAME = "frame1">
 9 </FRAMESET>
10
11 </HTML>
```

This file is `emptyframe.html`. I put tags into it only to avoid the browser's "document contains no data" warning (Wagner p. 282):

```
 1 <HTML>
 2 </HTML>
```

This file is `frame1.html`:

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>One frame can write into another</TITLE>
 4
 5 <SCRIPT TYPE = "text/javascript">
 6
 7 //This function is called as soon as all the frames in the frameset have been
 8 //loaded.
 9
10 function f()
11 {
12     parent.frame0.document.open("text/html");
13     parent.frame0.document.write(
14         "<HTML>",
15         "<BODY>",
16         "This is frame0.",
17         "</BODY>",
18         "</HTML>"
19     );
20     parent.frame0.document.close();
21 }
22 </SCRIPT>
23
24 </HEAD>
25 <BODY>
```

```
This is frame1.
</BODY>
</HTML>
```

To save typing, the above function could be written

```
function f()
{
    var d = parent.frame0.document;
    d.open("text/html");
    d.write(
        "<HTML>",
        "<BODY>",
        "This is frame0.",
        "</BODY>",
        "</HTML>"
    );
    d.close();
}
```

For `with`, see Flanagan pp. 77−79, 185; Wagner pp. 154−155.

```
function f()
{
    with (parent.frame0.document) {
        open("text/html");
        write(
            "<HTML>",
            "<BODY>",
            "This is frame0.",
            "</BODY>",
            "</HTML>"
        );
        close();
    }
}
```

Make sure that the other frame exists:
Flanagan pp. 169−170; Wagner pp. 275−278

```
function f()
{
    if (parent == null || parent.frame0 == null) {
        return;
    }

    var d = parent.frame0.document;
    d.open("text/html");
    d.write(
        "<HTML>",
        "<BODY>",
        "This is frame0.",
        "</BODY>",
        "</HTML>"
    );
    d.close();
}
```

What could go wrong if you shorten the **`if`** in the above function to

```
    if (parent.frame0 == null) {
        return;
    }
```

Use the above technique to see if the current document is being viewed in a frame:

```
<SCRIPT TYPE = "text/javascript">
if (parent == null) {
    document.write("This document is not being viewed in a frame.");
} else {
    document.write("This document is being viewed in a frame.");
}
</SCRIPT>
```

A form that changes its shape as you type:
Flanagan pp. 204−205, 386

```
<HTML>
<HEAD>
<TITLE>Are you currently taking any medication?</TITLE>
</HEAD>

<FRAMESET ROWS = "100%,*" onLoad = "window.hiddenframe.writeform();">
    <FRAME NAME =   "formframe" SRC =  "emptyframe.html" FRAMEBORDER = 0>
    <FRAME NAME = "hiddenframe" SRC = "hiddenframe.html" FRAMEBORDER = 0>
</FRAMESET>
</HTML>
```

`emptyframe.html` is the tiny file shown above. This file is **`hiddenframe.html`**:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<SCRIPT TYPE = "text/javascript">

//Assume that they take no medication unless they say so.
var medicated = false;
var medication = "";

function writeform(checked)
{
    var d = parent.formframe.document;
    d.open("text/html");

    d.write(
        '<HTML>',
        '<BODY>',
        '<HR>',
        '<FORM NAME = "medform" METHOD = POST ',
            'ACTION = "/cgi-bin/cgiwrap/~mm64/medication">',
        '<H2>Medical history</H2>',

        ’<INPUT TYPE = CHECKBOX NAME = "medicated" onClick = "’,
            ’parent.hiddenframe.medicated = this.checked;’,
            ’if (!this.checked) {’,
                ’parent.hiddenframe.medication = ’,
                    ’document.medform.medication.value;’,
            ’}’,
            ’parent.hiddenframe.writeform();’,
            ’return true;’,
        ’"’
    );

    if (medicated) {
        d.write(’ CHECKED’);
    }

    d.write(
        ’>Are you currently taking any medication?’
    );

    if (medicated) {
        d.write(
            ’<P>’,
            ’Please describe it:’,
            ’<BR>’,
            ’<TEXTAREA NAME = "medication" ROWS = 4 COLS = 40>’,
            medication,
            ’</TEXTAREA>’
        );
    }

    d.write(
        ’<P>’,
        ’<INPUT TYPE = SUBMIT VALUE = "Submit history.">’,

        ’<INPUT TYPE = RESET VALUE = "Start again." onClick = "’,
            ’parent.hiddenframe.medicated = false;’,
            ’parent.hiddenframe.medication = \’\’;’,
            ’parent.hiddenframe.writeform();’ +
            ’">’,

        ’</FORM>’,
        ’<HR>’,
        ’</BODY>’,
        ’</HTML>’
    );

    d.close();
}
</SCRIPT>
</BODY>
</HTML>
```

When the frameset is loaded, the raw output of **`writeform`** is

```
<HTML>
```

```
 2 <BODY>
 3 <HR>
 4 <FORM NAME = "medform" METHOD = POST
 5     ACTION = "/cgi-bin/cgiwrap/˜mm64/medication">
 6
 7 <H2>Medical history</H2>
 8
 9 <INPUT TYPE = CHECKBOX NAME = "medicated" onClick = "
10     parent.hiddenframe.medicated = this.checked;
11     if (!this.checked) {
12         parent.hiddenframe.medication =
13             document.medform.medication.value;
14     }
15     parent.hiddenframe.writeform();
16     return true;
17 ">Are you currently taking any medication?
18
19 <P>
20 <INPUT TYPE = SUBMIT VALUE = "Submit history.">
21
22 <INPUT TYPE = RESET VALUE = "Start again." onClick = "
23     parent.hiddenframe.medicated = false;
24     parent.hiddenframe.medication = ’’;
25     parent.hiddenframe.writeform();
26 ">
27
28 </FORM>
29 <HR>
30 </BODY>
31 </HTML>
```

But when the user first clicks on the checkbox, the raw output of **`writeform`** now includes lines 19−23, as well as the **`CHECKED`** attribute in line 17:

```
 1 <HTML>
 2 <BODY>
 3 <HR>
 4 <FORM NAME = "medform" METHOD = POST
 5     ACTION = "/cgi-bin/cgiwrap/˜mm64/medication">
 6
 7 <H2>Medical history</H2>
 8
 9 <INPUT TYPE = CHECKBOX NAME = "medicated" onClick = "
10     parent.hiddenframe.medicated = this.checked;
11     if (!this.checked) {
12         parent.hiddenframe.medication =
13             document.medform.medication.value;
14     }
15     parent.hiddenframe.writeform();
16     return true;
17 " CHECKED>Are you currently taking any medication?
18
19 <P>
20 Please describe it:
21 <BR>
```

```
<TEXTAREA NAME = "medication" ROWS = 4 COLS = 40>
</TEXTAREA>

<P>
<INPUT TYPE = SUBMIT VALUE = "Submit history.">

<INPUT TYPE = RESET VALUE = "Start again." onClick = "
    parent.hiddenframe.medicated = false;
    parent.hiddenframe.medication = ’’;
    parent.hiddenframe.writeform();
">

</FORM>
<HR>
</BODY>
</HTML>
```

The gateway **medication** is:

```
#!/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;
```

Open a new window displaying an existing file:
Flanagan pp. 176−178, 197−200, 568−570; Wagner pp. 245−248

Don’t set the **defaultStatus** of the new window until it has been completely loaded.

There are two ways to tell the window to do something when it’s completely loaded: (1) write the **onLoad =** attribute in the **<BODY>** or **<FRAMESET>** tag of the file to be displayed in the window; (2) store the name of a function in the window object’s **onload** property as in line 15.

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>Open another window</TITLE>
 4 </HEAD>
 5 <BODY>
 6 <H1>Open another window</H1>
```

```
This is window 1.
<SCRIPT TYPE = "text/javascript">

var w2 = window.open("open2.html", "window2",
    "innerWidth=300,innerHeight=200," +
    "screenX=0,screenY=0," +
    "status");

w2.onload = greet;

function greet()
{
    w2.defaultStatus = "Hello, window 2!";
}
</SCRIPT>
</BODY>
</HTML>
```

This file is **`open2.html`**:

```
<HTML>
<HEAD>
<TITLE>Window 2</TITLE>
</HEAD>
<BODY>
<H1>Window 2</H1>
This is window 2.

<SCRIPT TYPE = "text/javascript">
window.opener.defaultStatus = "Hello, window 1!";
</SCRIPT>

</BODY>
</HTML>
```

Open a new window and write a text/html document into it

A **`window`** and a **`document`** both have a method named **`open`**.

```
<HTML>
<HEAD>
<TITLE>Open a new window and write a text/html document into it</TITLE>
</HEAD>
<BODY>
<H1>Open a new window and write a text/html document into it</H1>
<SCRIPT TYPE = "text/javascript">

var w2 = window.open("", "window2",
    "innerWidth=300,innerHeight=200," +
    "screenX=0,screenY=0," +
    "status");

w2.document.open("text/html");

w2.document.write(
    "<HTML>",
    "<HEAD>",
    "<TITLE>A text/html document</TITLE>",
    "</HEAD>",
    "<BODY>",
    "<H1>A text/html document</H1>",
    Date(),
    "</BODY>",
    "</HTML>"
);

w2.document.close();

</SCRIPT>
</BODY>
</HTML>
```

Open a new window and write a text/plain document into it:
Flanagan pp. 223−224; Wagner pp. 313, 835

If the document has lots of **<**’s, **>**’s, and **&**’s, it’s simpler to let it be MIME type **`text/plain`** instead of **`text/html`**:

```
<HTML>
<HEAD>
<TITLE>Open a new window and write a text/html document into it</TITLE>
</HEAD>
<BODY>
<H1>Open a new window and write a text/plain document into it</H1>
<SCRIPT TYPE = "text/javascript">

var w2 = window.open("", "window2",
    "innerWidth=300,innerHeight=200," +
    "screenX=0,screenY=0," +
    "status");

w2.document.open("text/plain");

w2.document.write(
    "<HTML>",
    "<HEAD>",
    "<TITLE>A text/plain document</TITLE>",
    "</HEAD>",
    "<BODY>",
    "<H1>A text/plain document</H1>",
    Date(),
    "</BODY>",
    "</HTML>"
);

w2.document.close();

</SCRIPT>
</BODY>
</HTML>
```

Open a new window and write a image/x-xbitmap document into it:
Flanagan pp. 225−226

```
<HTML>
<HEAD>
<TITLE>Open a new window and write a image/x-xbitmap document into it</TITLE>
</HEAD>
<BODY>
<H1>Open a new window and write a image/x-xbitmap document into it</H1>
<SCRIPT TYPE = "text/javascript">

var w2 = window.open("", "window2",
    "innerWidth=300,innerHeight=200," +
    "screenX=0,screenY=0," +
    "status");

w2.document.open("image/x-xbitmap");

w2.document.write(
    "#define square_width 24\n",
    "#define square_height 24\n",
    "static unsigned char square_bits[] = {\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0x00, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xAA, 0xFF,\n",
    "   0xFF, 0x55, 0xFF,\n",
    "   0xFF, 0xAA, 0xFF,\n",
    "   0xFF, 0x55, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "   0xFF, 0xFF, 0xFF,\n",
    "};\n"
);

w2.document.close();

</SCRIPT>
</BODY>
</HTML>
```

To see the rows and columns of hexadecimal numbers for debugging, change the **"image/x-xbitmap"** to **"text/plain"** in line 14.

```
#define square_width 24
#define square_height 24
static unsigned char square_bits[] = {
    0xFF, 0xFF, 0xFF,
    0xFF, 0x00, 0xFF,
    0xFF, 0xFF, 0xFF,
etc.
```

The 16 hexadecimal digits

A *hexadecimal digit* is an abbreviation for four bits. There are 16 hexadecimal digits because there are 16 possible combinations of four bits. You can write them in either upper or lowercase, but make sure you have a **0x** (zero lowercase x) in front of each pair.

0	□□□□
1	■□□□
2	□■□□
3	■■□□
4	□□■□
5	■□■□
6	□■■□
7	■■■□
8	□□□■
9	■□□■
A	□■□■
B	■■□■
C	□□■■
D	■□■■
E	□■■■
F	■■■■

For example, **0x0F** is ■■■■□□□□.

Where to find .xbm files

The directory **/usr/local/apache-1.3.14/icons** on **i5.nyu.edu** has many **.xbm** files:

```
1$ cd /usr/local/apache-1.3.14/icons
2$ pwd
3$ ls -l *.xbm | more
```

To copy **ball.xbm** to your **public_html** subdirectory,

```
4$ cd
5$ cd public_html

6$ cp /usr/local/apache-1.3.14/icons/ball.xbm .
7$ ls -l ball.xbm

8$ chmod 644 ball.xbm
9$ ls -l ball.xbm
```

Draw an image/x-xbitmap flag

```
<HTML>
<HEAD>
<TITLE>An image/x-xbitmap flag</TITLE>
</HEAD>
<BODY>
<H1>An image/x-xbitmap flag</H1>
<SCRIPT TYPE = "text/javascript">

var bits_per_byte = 8;
var bits_per_stripe = 8;            //height of each horizontal stripe, in bits
var height = 13 * bits_per_stripe;  //thirteen colonies
var width = 2 * height;             //must be multiple of bits_per_byte

var w2 = window.open("", "window2",
    "innerWidth=" + (width + 16) + ",innerHeight=" + (height + 16));
w2.document.open("image/x-xbitmap");

w2.document.write(
    "#define flag_width ", width, "\n",
    "#define flag_height ", height, "\n",
    "static unsigned char flag_bits[] = {\n"
);

for (var row = 0; row < height; row = row + 1) {
    for (var col = 0; col < width / bits_per_byte; col = col + 1) {
        if (row < height / 2 && col < width / (2 * bits_per_byte)) {
            //Black union jack in upper left corner.
            w2.document.write("0xFF, ");
        } else if (row % (2 * bits_per_stripe) < bits_per_stripe) {
            //black stripe
            w2.document.write("0xFF, ");
        } else {
            //white stripe
            w2.document.write("0x00, ");
        }
    }

    //at end of each row
    w2.document.write("\n");
}

w2.document.write("};\n");
w2.document.close();

</SCRIPT>
</BODY>
</HTML>
```

List the MIME types:

NEXT TIME: retrofgit fetaures from **`sortmime.html`** in next handout.
Flanagan pp. 212, 471−474, 478

```
<HTML>
```

```
<HEAD>
<TITLE>List the MIME types</TITLE>
</HEAD>
<BODY>
<H1>List the MIME types</H1>
<TABLE BORDER>
<TH></TH>
<TH>type</TH>
<TH>description</TH>
<TH>suffixes</TH>
<TH>enabledPlugin</TH>

<SCRIPT TYPE = "text/javascript">
for (var i = 0; i < navigator.mimeTypes.length; i = i + 1) {
    document.write(
        "<TR>",
        "<TD ALIGN = RIGHT>", i + 1, "</TD>",
        "<TD>", navigator.mimeTypes[i].type, "</TD>",
        "<TD>", navigator.mimeTypes[i].description, "</TD>",
        "<TD>", navigator.mimeTypes[i].suffixes, "</TD>",
        "<TD>", navigator.mimeTypes[i].enabledPlugin == null ? "&nbsp;" :
            navigator.mimeTypes[i].enabledPlugin.name,
        "</TD>",
        "</TR>"
    );
}
</SCRIPT>

</TABLE>
</BODY>
</HTML>
```

▼ **Homework 10.1: make a table of plug-ins**

Make a table of the elements of the `navigator.plugins` array, like the above table of the `navigator.mimeTypes`.

▲

Do you have the plug-in for a given MIME type?

A MIME type may be built into the browser (e.g., `text/html`) or may be enabled by a plug-in or helper application. If it's enabled by a plug-in, you can `<EMBED>` it:

```
<SCRIPT TYPE = "text/javascript">
if (navigator.mimeTypes["video/quicktime"] == null) {
    document.write('Click <A HREF = "http://www.quicktime.com">here</A>',
        ' to learn how to get QuickTime.');
} else if (navigator.mimeTypes["video/quicktime"].enabledPlugin == null) {
    document.write("Built-in or enabled by a helper application.");
} else {
    document.write(
        "Enabled by plug-in.",
        "<BR>",
        "<EMBED SRC = "file.mov" WIDTH = 100 HEIGHT = 100>"
    );
}
```

```
</SCRIPT>
```

Eliminate repetition with a variable:

```
<SCRIPT TYPE = "text/javascript">
var mt = navigator.mimeTypes["video/quicktime"];

if (mt == null) {
    document.write('Click <A HREF = "http://www.quicktime.com">here</A>',
        ' to learn how to get QuickTime.');
} else if (mt.enabledPlugin == null) {
    document.write("Enabled by helper application.");
} else {
    document.write(
        "Enabled by plug-in.",
        "<BR>",
        "<EMBED SRC = "file.mov" WIDTH = 100 HEIGHT = 100>"
    );
}
</SCRIPT>
```

Put a Java applet in a page of HTML

Create a file named `MyApplet.java` in your `public_htm` subdirectory, containing

```
import java.applet.*;    //Applet Package
import java.awt.*;   //Abstract Windowing Toolkit Package

public class MyApplet extends Applet {
    public void paint (Graphics g)
    {
        g.setColor (Color.blue);
        g.drawRect (0, 0, 20, 10);
        g.drawString ("lower left", 0, 99);

        g.setColor (Color.red);
        g.drawLine (149, 0, 149, 99);
    }
}
```

```
1$ setenv SYSNAME alpha_osf32c
2$ echo $SYSNAME

3$ javac MyApplet.java
4$ ls -l MyApplet.class

5$ chmod 644 MyApplet.class
6$ ls -l MyApplet.class
```

In a page of HTML you can now write

```
<APPLET CODE = "MyApplet.class" WIDTH = 150 HEIGHT = 100 ALT =
"This browser understands the APPLET tag but does not have Java enabled now.">
This browser does not understand the APPLET tag.
</APPLET>
```

□