**Lab 08**

**Introduction to Event Handling in JavaScript**

**Background.**  
JavaScript is a client-side scripting language for the web. That means that the code is executed by a JavaScript engine integrated into the client browser. All current browsers support JavaScript, although not all are compliant to the same standard. For this lab and for the exam, you need only test your code under Mozilla Firefox.

JavaScript is not "baby-Java." The only relationship between Java and JavaScript is that both are object-oriented languages and both share C-style control structures and operators. That is about where the similarity ends. The choice of name is unfortunate. JavaScript was created by Netscape and the name is a trademark of Sun.

Client-side scripts are generally triggered by events that occur when the user interacts with the client browser, often through FORM elements. Since JavaScript is client-side, there are far more ways to interact with the user in JavaScript than there are with PHP. You also have more control over writing to the current page in JavaScript. Click these buttons for a demonstration:

Server-side script languages, like PHP, provide control over things that happen behind the scenes on the server and things that occur over the long term (not just when the user is viewing the page). PHP is good for logging information, controlling user access, and anything involving processes on the server (a mailer for example). JavaScript is good for things that occur only during the time the user is on the page.

The focus of this lab is the event-handling aspects of JavaScript, where user activity is tied to elements in the HTML. Objects in the tree are linked to [events and event handlers](http://docs.sun.com/source/816-6408-10/handlers.htm). Although a JavaScript can be in a standalone file, in this lab you will embed your scripts into an HTML file (lab08.html). Since JavaScript is client-side, you can write lab08.html on your local PC and load it onto a web server when you are finished (if you want). Otherwise, modify lab08.html directly from within /public\_html in your account on odin.

The debugging capabilities of the JavaScript interpreter are browser dependent. If you use Netscape, Mozilla or Firefox, you will either type 'javascript:' in the URL bar or open the Tools->Error Console from the menu (depending on the version). Any errors in your script will be displayed in the console window. All versions of Mozilla Firefox after 3 use the Tools->Error Console from the menu bar. You can also install a Firefox extension called [firebug](http://getfirebug.com/).

You will need to know a little HTML for this lab, as well as the basics of how embedding Javascript into HTML works. Create a 'lab08' directory under your $HOME/public\_html. (Keep the default permissions since all directories you want accessible on the Internet must have 755 permission and files must have 744.) Copy events.html into $HOME/public\_html/lab08/:

cp /home/fac/clei/public\_html/teaching/files/S19\_3500/labs/lab08/JavaScript/events.html .

(To make the existing code in events.html work with minimal modification, you may want to create an Images subdirectory to hold your images for this lab.)

View the source of events.html while looking at the behavior in your browser: [events.html](http://cs.csub.edu/~clei/files/S17_3500/labs/lab08/events.html). Everything you need to complete this lab is in the documentation in events.html.

**WHAT TO DO FIRST**

1. Make a backup of events.html for reference in case you need it. For the first few items, you will insert code into the opening script in events.html. The remaining items involve putting JavaScript into event handlers.
2. Download your own images for each of the image references in events.html. Grab an image directly from the command line on server:
3. $ wget http://www.anywhere.com/some\_image.jpg // will grab some\_image.jpg

Move the images from $HOME to your /public\_html/Images subdirectory.

1. Open events.html in vim.

**PROBLEM #1**

1. Following the syntax in the code below, create an array to hold four background images in the opening script:
2. var myArray = []; // this array will grow as needed
3. myArray[0] = new Image();
4. myArray[1] = new Image();
5. and so on...
6. Assign the src attribute of each image object to one of your image filenames following this code:
7. myArray[0].src = "some\_image.jpg";
8. and so on....
9. Modify the event handler that changes the background image of the body of the document to toggle through your array of background images.

**PROBLEM #2**

1. Create an array to hold 4 or five different hex color codes This code should also be in the opening script. Assign each element in the array a different hex color code; e.g.: "#AABBCC".
2. Modify the event handler that changes the background color of the upper DIV area, toggle through your array of hex color codes.

**PROBLEM #3**

1. In the lower DIV area, add an onClick event to one of the images to prompt the user for a hex color code:
3. prompt('enter hex code: ','#FFFFFF'); // #FFFFFF is the default value

Change the DIV area background to that color.

1. Modify the text on the page so I know which image to click.

**PROBLEM #4**

1. Create a display() function to write your name in the DIV area labeled myArea - an empty function named display() is already there.
2. In the lower DIV area, add an onClick event in one of the images that doesn't have one to call display().

That's it.

What to submit for this lab

The URL of your completed file.