JavaScript Tutorial: Image swap on mouse over and mouse out

Overview

This tutorial covers how to write JavaScript and HTML so that when visitors to your site move the mouse over an image, the image will change to a new image. When the visitor moves the mouse out from (off) the image, the original image will reappear. This application, called image swapping, is a common use of JavaScript and often the first and only use many people make of JavaScript. It certainly has intrinsic appeal. However, you can build on the understanding gained by this examples to design and build new and different applications, including games.

This application will make use of the following JavaScript and HTML features:

- script tag
- function
- onMouseOver, onMouseOut and onClick events
- img tag with a name attribute

For an example of mouse over/image swapping, see
newmedia.purchase.edu/~Jeanine/jsexamples.html

Key Design Issues

Task: Identify a place in the HTML document that will hold, in turn, both images.
Logic: The application requires one of two images to appear in a particular place. We need a way to specify the place and also independently specify the image.
Solution: The HTML image tag, <img…> sets up an image. The src attribute specifies the file for the image. To implement image swapping, you will use JavaScript to change the value of the src attribute. One (or two) more things are necessary here. To use the JavaScript, you need to specify the img tag that you want to change (whose src attribute you want to change). This is done by giving the img tag a name. The other thing that may be necessary is to fix the size of the image. This is done using the img attributes for width, height or both.

Task: Indicate to the browser, the computer program that displays the HTML document, that you want something to happen when the visitor moves the mouse over and out of certain areas on the screen.
Logic: The programming terms for this type of task is event handling: we want the system to look out for certain events happening and, when they do, respond or handle the events in ways we specify.
Solution: JavaScript provides for just the event handling we need! We can specify JavaScript to handle onMouseOver and onMouseOut for anchor tags (a tags). So since we want the action to be done when the visitor moves the mouse over an image, we make

the image be between the <a …> and tags. Within the a tag, you specify the handling of an event, say the mouse over event, by writing:
<a …. OnMouseover="*the JavaScript code*" >. In order to make this line compact, we suggest making the JavaScript code a function call. This is explained below.

Implementation

Before proceeding, you need to create or download two images into the same directory in which you store the HTML file. Following the example cited above, let us say that the two image files are Liz-book.jpg and Darcy3.jpg. These names are whatever you make them. Please note that the files need not have the same file extension. The img tag is the following:

```
<img src="Liz-book.jpg" height=300 name="picture1">
```

(You should try this without the height attribute. If your two image files are different in size, you will see two different size images and this may or may not be what you want.)

The critical attribute here is the **`name="picture1"`**. This gives the image tag a name that can be referenced by JavaScript.

JavaScript code is located within certain tags, such as the a tag or within a pair of tags called script and /script located in the head section of the HTML file. Both places are used for this application. In this example, we choose to define a function to perform the image swap. A function is a set of statements that you, the programmer, encapsulate with its own name. You then use the function as if it was a built-in part of the language. Defining a function is optional but it is generally a good tactic to do in programming. The function is defined in the head section of the HTML file. The call to the function is within the body of the HTML document. Specifically, the call will be between an a and /a tags.

Since anchor tags can have attributes specifying the handling of events, as we indicated above, we put the img tag between <a …> and the tags. The HTML within the <body> …. </body> tags is

```
<a href="" onMouseOver="movein('Darcy3.jpg');"
onClick="return false;"
onMouseOut="movein('Liz-book.jpg');">
<img src="Liz-book.jpg" name="picture1" height="300" > </a>
```

The **`href=""`** indicates that there is no hyperlink. It is also necessary to specify that the action of clicking the mouse on the anchor (the image) results in no action. This is indicated by the JavaScript **`"return false;"`** for the value of onClick. (You can change this if you like. Remove the whole **`onClick`** part and make the hyperlink a file name or a full URL.) Note that JavaScript statements end in semicolons. The handling

for the three events, **onClick**, **onMouseOver** and **onMouseOut**, are each specified the same way: you use an equal symbol (=) and a quoted string. The **onClick** action of **"return false;"** is to not do the usual action. This is what returning false does. (Try removing this clause and see what happens when you click on the image.) The actions for the other two events are similar: a function called **movein**, to be described next, is to be called with two different argument values. The HTML in the a tag indicate indicates the function **movein** is to be invoked when each of the events happen. When the event is MouseOver, then the call is done using the string **'Darcy3.jpg'** and when the event is MouseOut, the call is done using the string **'Liz-book.jpg'**. The use of single quotation marks within the string delimited by double quotation marks is necessary. If we used

```
onMouseOver="movein("Darcy3.jpg");"
```

The JavaScript interpreter would set the event handler to **"movein("** and this would cause errors when the event happened.

We now return to the subject of functions. In JavaScript and other programming languages, a mechanism exists for programmers to extend the language to improve the clarity of the code. In this example, we know we need to change the image file in an image tag in two different situations. Rather than writing similar code in two different places, we define a function we call *movein*. That name is totally arbitrary. Function names should be evocative of what they do and this one is. The function is defined in the head section of the HTML within script and /script tags. Here is that part of the code:

```
<head>
<title> Rollover test </title>
<script language="JavaScript">
<!--
function movein(image)
{
window.document.picture1.src=image;
}
// End -->
</script>
</head>
```

The script tag indicates the language to be used. (The other common choice is VBScript.)

The <!-- and the // End --> are tags indicating comments. What they do is prevent the JavaScript from being displayed by a browser that does not understand JavaScript. Of course, if the browser does not understand JavaScript, the mouse over/image swap will not work, but at least you will still see the original picture with no extra, meaningless code.

The function definition sets the name of the function as **movein**. The **(image)** indicates that the function will be called with what is termed an argument or a parameter. Within the body of the function definition, that is, what is between the curly brackets, the name image will refer to whatever the argument was on the function call. This function has one line of code. The one line indicated the following action: make **image** the **src** (source) of the image tag named **picture1** of the **document** located in the current **window.** Note that **picture1** is the value of the name we gave (put in) the **img** tag. Note also that this is arbitrary (it needs to be all one word, no spaces, starting with an alphabetic character) but once you name something, you need to be consistent (not think it is pic or picture or pic1).

Put it all together! Do also make the changes indicated above and observe the effects. Try the following:

1. Print out the source of your HTML/JavaScript file. Draw lines to match up each quotation mark with the ending quotation mark, each tag with the ending /tag. Practice doing this will help you catch a most common type of error.
2. Make the action on moving the mouse out from the picture be to swap in a third picture. You can restore the original image by reloading the page.
3. Take the img tag out from between the a and /a tags. Instead, just put some text there, such as Move Mouse here to see a new image.
4. Include a second image, with a different name (picture2 is a good choice). Make the action on moving in and out over one image do something to the second image by adding a second line to the movein function and a second parameter.

```
function movein (image1, image2) {
  window.document.picture1.src = image1;
  window.document.picture2.src = image2;
}
```