ast month we looked at a very simple application of JavaScript: displaying a line of text in the web browser's status bar when the mouse is over a hyperlink. The status bar will remain something which we use in the future, but this month we'll look at how to expand the link-trigger mechanism to create an active site menu.

tep 1: Design Guidelines

The first step in designing an active menu is to decide how each individual menu item will be represented. For our example, we'll use a simple graphic with the name of the page in question:

or each page listed in your site menu you would create a graphic. Since we will be pre-loading all of the graphics for the active menu, you should try to keep the individual images as compact as possible to reduce the time it takes for the page to load.

The guidelines for a good-looking menu are up to you, the programmer (or your graphics designer, if we're talking more-than-one-dude site management). For the example graphic above, the name of each page is to be done in anti-aliased text. Customize your own graphics to be as complex (or simple) as you wish.

Next, you must consider what effect you wish to, errr, effect on the original graphic when the mouse passes over it. In our example, a glowing box will surround the text:

s the mouse is moved into the graphic, the glowing box appears surrounding it, and as the mouse moves off of the graphic the border effect must disappear. This is quite typical of the buttons you've seen elsewhere.

tep 2: Plan the Page

The next step in your active menu is to plan how the menu will be placed on the page. For a simple page such as the one we're creating this month, we'll just place the graphics on the page using IMG tags. In order to make our JavaScripting a bit easier, we'll also assign each image a name:

```
<IMG SRC="news_mu.gif" BORDER="0" NAME="News_Menu">
```

Assigning the images a name will make our JavaScript more readable and much easier to program. You should adopt some kind of generalized naming scheme, as well — I've used the general subject of the page represented by the button (News) and tacked on "_Menu" as an indication that this image is used in the menu. Our example this month has two menu items, one for a corporate news page and the other for a page on the company's management structure:

```
<BODY BGCOLOR="white">
    <IMG SRC="news_mu.gif" BORDER="0" NAME="News_Menu"><BR>
    <IMG SRC="management_mu.gif" BORDER="0" NAME="Manage_Menu">
</BODY>
```

The example I'm using here is very simple — you can probably imagine a more complex menu which you lay out using HTML tables.

tep 3: Integrate the JavaScript Code

Now that you have the graphics made — both normal and glowing types for each menu item — it's time to delve into the JavaScript. Since JavaScript is so much like C and C++, readers familiar with those languages will immediately recognize what's going on. For those without such a background, I'll explain.

Typically you will declare a section of the web page to contain any variables or functions you create. This declaration is made using the SCRIPT tag:

```
    <!--   Hide the scripting from old browsers.
    <SCRIPT Language="JavaScript">
```

You may be wondering why the script is enclosed in comment brackets (`<!-- -->`). An older browser would display the actual text for the script in the browser window, since it does not understand how to interpret the scripting commands. Therefore, the comment tags hide the script from being printed. Browsers which support JavaScript will handle the scripting properly.

Immediately following the `SCRIPT` tag begins our program code. The first order of business in this instance is reading in the menu images:

```
    //
Initialization code:
    news_norm = new Image();
    news_norm.src = "news_mu.gif";
    news_glow = new Image();
    news_glow.src = "news_mg.gif";

    manage_norm = new Image();
    manage_norm.src = "management_mu.gif";
    manage_glow = new Image();
    manage_glow.src = "management_mg.gif";
```

I won't go into the specifics of what we're looking at here, except to say that two new image objects are created for each menu. Each menu item needs two — after all, we have a normal and a glowing version to work with. The source (`src`) fields of the images are then set appropriately to the URL for the graphic in question. In this case, the graphics are contained in the same folder as the web page containing this HTML.

Following the "initialization code" shown above are the function definitions. We need to define two functions for this page: one to make a menu item glow, another to make the menu item change back to its normal form. We'll call the first function glow:

```
    function glow(num)
    // Written:          Jeff Frey, 5/15/98
    // Purpose:          Make an item on the menu glow.
    // Last Mod:   n/a

      switch (num)

        case 'News_Menu':
          document.images[num].src = news_glow.src;
        break;

        case 'Manage_Menu':
          document.images[num].src = manage_glow.src;
        break;
```

When called, the "glow" function receives a parameter called "num." This parameter specifies which menu item you are referencing. The data you pass in "num" is the name of the image in which you're interested — for the news menu item we're interested in the 'News_Menu' image. The switch statement which follows is used to decide which image is being passed. Think of it as a really long if... then statement: if num is 'News_Menu' then do this, otherwise if num is 'Manage_Menu' then do this, otherwise... etc. Once the switch finds the correct menu item, our function tells the browser to set the image referenced by the name in "num" to a glowing image. That's all there is to it!

The second function, aptly called "unglow" sets the image referenced by "num" to the original, unglowing graphic:

```
function unglow(num)
// Written:        Jeff Frey, 5/15/98
// Purpose:        Make an item on the menu not glow.
// Last Mod:   n/a

  switch (num)

    case 'News_Menu':
      document.images[num].src = news_norm.src;
    break;

    case 'Manage_Menu':
      document.images[num].src = manage_norm.src;
    break;
```

The differences between "glow" and "unglow" follow naturally from the purpose of each function. Where the "glow" function sets a glowing image, "unglow" must set a non-glowing image.

This completes the initialization code and the functions for the page. To indicate that you've finished scripting, use the /SCRIPT tag:

```
    // End hiding JavaScript -->
</SCRIPT>
```

First, we close the comment bracket for older browsers. Then, we end the scripting section with the /SCRIPT tag.

tep 4: Creating the Links Which Make the Graphics Move

Now that all of our functions are in place and ready to go, we need to set up the hyperlinks so that they will display the correct graphic when the mouse is over the image, for instance:

```
<A HREF="news.html" onMouseOver="glow('News_Menu'); return true;"
onMouseOut="unglow('News_Menu'); return true;">
```

As we saw last month, the onMouseOver event is raised when the mouse enters the link. In that case, the mouse has now rolled over the graphic, so we should display the glowing menu item. This is accomplished by calling our "glow" function and specifying that we're looking at the news menu (`glow('News_Menu');`). The reverse effect, making the menu item not glow, is accomplished by calling the "unglow" function in the same manner.

tep 5: Test and Relax

That's it! You've now added some more interesting active content to your web page. You can find the complete source code for this project at http://indigo.lvc.edu/~frey/dec/test.html .

As always, please take a look at the full source for a better understanding. Also, be aware that there are many ways to expand on the idea of the active menu. For instance, if you create a two-frame page and use the first frame for your active menu, you have the ability to add a third graphic to the previous two. This third graphic specifies that you're already on the page referenced by the menu item. The site at http://indigo.lvc.edu/~sg/ is an excellent example of just this — when one of the pages in the menu is active, that page's menu item will no longer "glow" when the mouse moves over it.

Keep the column ideas flowing, too! My email address is right below and I do take suggestions seriously!

Jeff Frey
jeff@applewizards.net