by Don Lancaster

Bouncy Bricks

A major upgrade to my *Guru's Lair* website has been long overdue, so I wanted to add some of the latest available features and technology. Newer stuff that goes way beyond the essential webmastering fundamentals we looked at back in *BWHISTLE.PDF*.

The dilemma is that new "gee whiz" features come at a price. They may lengthen files and download times. Or act differently on different browsers. Or favor webmaster, ISP, *advertiser*, or user at the expense of the others. Or end up annoying or garish. Or place an unreasonable load split between server and user.

Or have an ugly learning curve.

Let's take a look into several of the newer web tools and techniques that I chose to use. I used ***keep it subtle*** as my foremost guideline when adding anything glitzy. I found Adobe's *GoLive* to be a useful site editor.

Although I am very much a *Netscape* fan, it appears that Netscape 6 is an unmitigated disaster. Our *stats* show us a 70% - 20% overwhelming dominance of IE over Netscape 4, with Netscape 6 an utterly negliglible 0.3% usage. Netscape also has apparently stopped further browser development. Thus, a few of our enhancements ended up IE only but are still statically viewable on Netscape 4.

I did try to retain much of the "retro" look and "library shelf" model of the original site. I purposely chose *not* to use cascacding style sheets, grids, floating boxes, frames, full Java, interstitals, sound, grabs, or lockouts. These seem to me to often cause more problems than they solve.

And *SVG* or ***scalable vector graphics*** still seems rather premature, despite its obvious potential. And I do not yet have a secure shopping cart order system.

But here's some of the stuff I have found useful...

Meta Tags

We'll start with a few trivial and obvious suggestions. Every HTML document header is supposed to have its very own **<NAME>** tag. ***Always be sure each and every page on your website does in fact have a different name!*** It is easy to forget this crucial detail, leading to severe search engine and access problems.

Headers can optionally include **<META>** tags. These will provide information of one sort or another but should not directly affect what goes down on the screen. Proper use of **<META>** tags can very much improve your site's exposure and access. Meta tags are easily added with *GoLive* or most any text editor...

```
<META name="description" content="Welcome to...">
<META name="keywords" content="guru, lair, ...">
```

The first of these gives you a different page name for use by search engines or whatever. Which may or may not be similar to the actual **<NAME>** tag. But usually friendlier and with a somewhat stronger sell. Which makes your site a tad more attractive during a search.

The second meta tag lists the keywords that the search engines should associate with your site. Keywords should accurately reflect what you have to offer. Second guessing or spamming for better positioning is usually futile.

Another useful meta tag is the **refresh**. One use is to be sure your viewer is seeing the latest page updates. But the biggie is to **redirect** you from one page to another. Such as from an older .HTML page to a newer .ASP one...

```
<META HTTP-EQUIV="refresh" CONTENT="4"
 URL=http://www.tinaja.com/blat01.asp">
```

That "**4**" above is your time delay in seconds before your new site gets accessed. It is quite important to ***never use a zero time delay***, since the user's back arrows won't work!

When and where possible, a server side redirect (ask your ISP for details) should be used instead, as this is invisible and eliminates the delay and the extra web traffic.

Safe Colors

While zillions of colors are possible on a website, certain browsers and some platforms may show them differently. Your colors are best limited to a set of 216. Assigning only hex values of $00, $33, $66, $99, $CC, and $FF to each of the RGB red, blue, and green primaries.

See the demo at *www.htmlhelp.com/icon/hexchart.gif*.

An On-Site Search Engine

It is super important to provide a site search engine that handles both HTML and PDF format files. And making sure it is ***prominently*** accessible from every site page.

The first step is to get an IIS search up and working. This can be extremely difficult as their code seems exceptionally hostile and lacks useful error messages.

It is usually best the "steal the plans" from some working site. Then you slowly and carefully make only *incremental* changes in the code untill it does what you want it to.

Several useful IIS books are stashed in our *Webmastering* Library. *microsoft.public.inet.server.iis* is one newsgroup. The others are quickly found through *Deja News*.

In general, there are three elements that concern you as webmaster. First, you'll create an **.IDQ** textfile which is a rather strictly specified list of parameters the search engine needs to work with. Next, you'll create an **.HTX** container which is a HTML-like page that is to accept the engine's findings and report them. Finally, you will activate your on-page searches where you want by embedding a **<POST>** inside a **<FORM>** action that references your .IDQ.

Attention to detail is a must.

The usual debug problems are unfindable includes or a mixup of virtual and absolute filenames. Try your ISP and *Deja News* (now Google) for more help.

After your IIS is up and working, you go to the *Adobe* website and pick up a *PDF Search Plugin*. Installing this plugin is an absolute joy compared to getting IIS to work properly in the first place.

.ASP Files

The best of web sites will be a mix of things done by the server and browser. IE has a variation on an HTML file that is called an **ASP** file.

Short for *Active Server Pages* An .ASP file can be a set of instructions from which the server will build and then send out a web page. This allows for dynamic delivery of different content to different users. Usually, the .ASP file tells the server what to build up. An ordinary but custom HTML file then goes out to the viewer.

.ASP files give you lots of flexibility, and you definitely should consider using them. Because they greatly simplify site creation and maintenence. They are somewhat harder to test, since they must work "live" with your ISP.

A list of recommended .ASP programming books appears on our *Book Access* library shelf.

One obvious .ASP use is for counters and related record keeping, where you can keep score of who is accessing any page from where. While there are many, many advantages to .ASP files, the two biggies for me have been **includes** and **banner rotations**. Let's start by...

Including Includes

By far your simplest and most useful feature of any .ASP script is the include. This is just a plain old subroutine. For instance, you might have one standard footer that you use on all of your web pages. For changes, you'll modify only your include file and everything gets automagically picked up on each and every page.

Other obvious includes: a search engine module, a group of nav buttons, advertiser data, a list of recommended files, products or books. Or most anything else that you want to have magically appear on several pages.

The usual format for an include is...

```
<!--#include virtual=/subdirectory/filename"-->
```

Note that include files are just that. They might provide HTML code or scripting commands, but they must *not* have any <HTML>, <HEAD> or <BODY> start or end tags. Includes are dropped in place just as if you were entering them by hand. *GoLive* lets you edit these without any internal head or body commands. I usually include an "**X**" or a "**Z**" at the end of any include main filename to remind me that it is a fragment that cannot stand alone.

Another important .ASP file use is for my...

Banner Rotation

There's a banner rotator built into the IE .ASP processor. It is well documented, fairly easy to set up and it works. Sort of. But I wanted something totally different. I wanted to have eight banner slots per page. Into which eight (or more) banner advertisers would appear in a random yet exclusive order. Each advertiser thus appears only once in each page, alternating positions and ad neighbors. ***With no repeats***. The reader gets a changing and more interesting page that (hopefully) ups their click throughs. And only a single and easily changed master banner data file is needed for the whole site.

You can view a demo on our *Home Page* or most of our other web pages. Simply hit your *reload* key repeatedly to get new groups of banners.

I chose to use a nonstandard data file I call *banndata.txt* It basically contains three arrays of VBScript values. **blnk(n)** holds the url for advertiser **n** as plaintext. **bimg(n)** holds the image GIF filename for advertiser **n**. **balt(n)** holds the alternate text message for advertiser **n**.

Remember that an .ASP file is a set of instructions from which the server will build and then send out a web page. Early in this page, the banndata file is read as an include. Your net result is to enter three VBScript data arrays onto the specific page script from the single master data file.

Next, an array of the first eight letters **a-h** gets created. Each letter represents a possible page position for a banner. This array then gets shufffled by using a rule of ***"exchange any element with itself or a lower one"***. Like so...

```
Randomize
For i = 7 to 1 Step -1
j = Int(Rnd*(i+1))
x = a(j)
a(j) = a(i)
a(i) = x
Next
Pattern = a(0) & a(1) & a(2) & a(3) & a(4)
          & a(5) & a(6) & a(7)
```

The result creates a unique new eight character shuffled string such as **(fbdegach)**. This string determines who goes where this time on the page to be delivered. The ***position*** of each letter in the string determines location.

The .ASP file then finds ordinary fixed banner images and sends them to the client. Such as...

```
<%kk = (inStr(pattern, "a")%>
<A HREF="<%=blnk(kk)%>"
<IMG src=
"http://www.tinaja.com/banners/<%=bimg(kk)%>"
src="http://www.tinaja.com/banners/<%=bimg(kk)%>"
alt="http://www.tinaja.com/banners/<%=balt(kk)%>"
lowsrc="newsum.jpg"
height=65 width=280 border=0></A>
```

We first find out where "a" lies in our string and assign a number **kk** from 0 to 7 to it. We then generate the correct link for this advertiser, followed by their correct image, followed by their correct text alternate. An already loaded small file is used for lowsrc. This guarantees that all your **banners load last**, after the rest of the page is complete.

The next banner position in rotation would start off

with a **<%kk = (inStr(pattern, "b")%>** , and so on.

Note that your delivered file simply has the preshuffled banners in fixed positions.

Lots more on banners is in my ***BANNYEAR.PDF***. More source code details are available on request, and special designs are offered by way of our ***InfoPack*** services.

JavaScript

JavaScript is an interpreted computer language which (usually) runs on the client side, typically after a page gets fully downloaded. JavaScript can do all sorts of very good and very bad things for you and your user. It is especially useful for validating forms data, performing date and time stamping, or animating demos. But an incredible variety of new uses are emerging.

JavaScript is often activated early during a download. Various functions are defined as or before they are needed. They later are called in response to user actions.

The JavaScript Bible is one useful book, and I have gathered lots more JavaScript books together for you up at ***http://www.tinaja.com/amlink01.asp***. There is JavaScript newsgroup support at ***comp.lang.javascript***

I've got some disgustingly elegant JavaScript calculator examples up on my ***Magic Sinewave*** library pages. These do a complete classic direct ***Fourier transformation***. While rapidly Newton Method "shake the box" solving as many as 24 complex multiple angle ***Chebycheff*** trig polynomonial equations in 24 unknowns that (so far) sometimes go up as high as the 47th power. Which turns out a great heaping bunch uglier than it sounds. But has led to a stunning and utterly unexpected ***energy efficiency*** breakthrough.

For some strange reason, JavaScript only is active on JavaScript enabled browsers. Thus, older browsers or a user who purposely shuts JavaScript off will be unable to use any JavaScript features you may provide.

My main and much simpler use for JavaScript so far is for my really sneaky improvement on...

Bouncy Buttons

A ***dynamic*** website is simply one that changes itself in response to user actions. The most common activity is to slide your mouse on over a button or hot area and get it to change color, position, or form. Let's look at an example that is again found at ***http://www.tinaja.com*** and my other Guru's Lair website pages.

I first created two button images, a "normal" button I called ***barro.jpg*** and a "pressed" button called ***barrox.jpg***

You place the following JavaScript code near your body beginning...

```
<SCRIPT language=JavaScript1.1>
function sI(imgName, type){if type == "x")
{glotz = "barrox.jpg"}
else {glotz = "barro.jpg}
document.images[imgName].src = glotz
return=false}
</SCRIPT>
```

Or, restated in English: Turn JavaScript on and define a function **sI** that accepts two variables. The first variable is the name of button the mouse is near. The second is either the lower case letter "x" for a pressed button, or any other letter (including "n") for a normal button. If "x" is selected, image the pressed button. If not, image normal.

Later, when you get to a button to be bounced, you add this code...

```
<A HREF="http://www.tinaja.com/info01.asp"
onMouseOver "return sI ('b18', 'x')
onMouseOut "return sI ('b18', 'n')>

<IMG src=barro.jpg name=b18
height=34 width=37 border=0 > </A>
```

We see two HTML statements. The first is your usual link statement with two new JavaScript mouse commands. The first command says "pass the button number and "x" to the previously defined JavaScript function **sI**, then come back. The second statement is your usual IMG size and default name info to which this particular **b18** button has now been uniquely defined.

Each button has to be given its own unique name so JavaScript can find it.

But even neater are my JavaScript...

Better Bouncy Bricks

The bouncy buttons are a standard way to liven up a JavaScript web page. Which works on any browser this side of Netscape 4. But there are two serious problems: Each button on a page seems to need a unique id. Making for edit, copy, and rework problems. Worse, each button that has a label or lettering on it needs two, three, or even four unique images. If you want bunches of unique labeled buttons, file sizes get huge and slow, edit gets painful, the pages fragment, and web traffic skyrockets.

Both IE4 and NetScape 6 offer lots of new features. One of which is letting your mouse control your ***background color*** of any ***table cell***. Yup. On mouseover, it takes any message, remembers it, recolors the background, and then rewrites the message.

I call these my ***bouncy bricks***.

All done using zero images and a only few dozen new bytes of code per button. This can be nearly ideal when you want lots and lots of live selections. The biggest downside is that you no longer can have button texture or shadow. A second is that really ancient browsers can go invisible on you.

The Javascript IE per-brick code is amazingly simple...

```
<td bgcolor=#33CCCC
onMouseOver="this.bg.color='#009999'"
onMouseOut="this.bg.color='#33CCCC'" >
<A href=:www.link_url_goes_here.com>
your link title goes here</A>
</td>
```

In this example, the button darkens on mouseover. To brighten, try **#66FFFF**. Note that a single quote gets used for any quote inside a quote. You could purposely emphasize your borders to group your buttons, or you can use zero width borders and cell spacing to create color highlights instead. You can color code all selections, separating, say, navigation, onsite, and offsite. Ferinstance, search engines seem purple somehow.

You can also add ***onMouseDown*** for a third color. Or use ***onMouseUP*** for a "been there, done that". But either of these violates our subtlety rule.

Detailed bordered and borderless examples appear at my *http://www.tinaja.com* and elsewhere on my website.

Bouncy bricks only work in IE4 and higher. Netscape 6 can do the same thing using a somewhat different coding. Netscape 4 gives you a static but readable button. Earlier browsers may totally trash the message. Again, JavaScript has to be active for the bouncy bricks to work.

Let me know if you find a way to further shorten this code. Or a way to add texture or shadow.

Fast Nav

Once any website gets above a certain size, users can easily get lost. Or not find your more important pages. At least not as often as you'd like them to. So, considerable thought should be given to site navigation.

A lost viewer is a gone viewer.

Your headers and footers should have links that take the user to other favored pages. Use of internal *anchors* is a must. An anchor differs from a "normal" url in that it starts with a page "#" and does a relative jump on the *same* page. Such as */#links*.

On any larger page, it should always be easy to get to the top or bottom from *any* screen view. Any long lists or sequences should be broken up and reachable by different navigation buttons or bricks.

Another sneaky trick are ads for yourself. Place small "crossref" text lines under each feature...

Click here for Instant Resource Solutions

It is also a good idea to use "two step" photos where you provide a fast loading thumbnail on the page and then let interested users take the time to expand upon it.

While there are JavaScript solutions that give you exact placement control and other benefits, use of the plain old HTML **TARGET="_BLANK** command works good enogh most of the time...

```
<A HREF="http://big_pix_image.jpg"
TARGET="_blank">

<IMG="http://thumbnail_image.jpg"
WIDTH=125 HEIGHT=125
LOWSRC="http://load_me_first.jpg"
ALT="Click to Enlarge"> </A>
```

Clicking anywhere on the small image will cause the large image to open in a new window. You then close this window to return to your original page.

The general rule is that anything between the **<A>** and the **</A>** is active for clicking.

Color coding and having your nav features consistently appear in expected ways and in expected places is super important. Watch this detail.

An easily reached *full site map* is essential. A well done *main library* page is also often useful. As will be a list of your *most popular files*.

As always, ***test, test, test***. Then test again. *Doctor HTML* is an especially useful aide here.

Favicon Icons

Let's wrap up with a simple enhancement for IE users. One that might significantly raise your repeat site traffic. Favicon is short for "favorite icon". If you have a properly formatted file named *favicon.ico* stashed in your server's home page directory, then any IE user can get your icon and place it on their favorites folder, on their location bar, on their start button, on their desktop, or on their taskbar.

The format is basically a RGB pixel map of size 16 x 16. There is a free paint-like favicon editor available to you at *www.favicon.com* You get your icon looking the way you want it to. They then email it back to you and you then FTP it to your website. Quick and simple.

Check it out.

For More Help

As mentioned, extensive IIS, ASP, JavaScript, and Acrobat book listings are found on our *Book Access* page.

Much more on these techniques might be found on my *Webmastering*, *Blatant Opportunist*, and *Acrobat* library pages of my *Guru's Lair* website.

Sourcecode, development software, analysis tools, and custom design help is also available through the *InfoPack* service from *Synergetics* or by emailing *don@tinaja.com*

Let's hear from you. ✦

Microcomputer pioneer and guru Don Lancaster is the author of 35 books and countless articles. Don maintains a US technical helpline you will find at (928) 428-4073, besides offering all his own ***books***, reprints and ***consulting services***.

Don also offers surplus bargains found on ***eBay*** and on his ***Bargain Pages***

Don is also the webmaster of ***www.tinaja.com*** ***You can also reach Don at Synergetics, Box*** *809, Thatcher, AZ 85552. Or you can use email via* ***don@tinaja.com***