HTMLib Tips 'n' Tricks

Here are some tips and tricks that may help HTML authors when developing sites. They are by no means comprehensive rules, more suggestions of how to accomplish some commonly desired effects.

NOTE : All of these tips were tested in **Internet Explorer** (3.0 beta 2) and **Netscape** (3.0 beta 5a).

Using fonts with <MARQUEE> elements
Using fonts within <TABLE>s
Browser independent background sound
How to load more than one document within a <FRAME> document
Creating an apparent BACK button
Browser independent <MARQUEE>s
How to determine the users browser type
Noisy Links

HTMLib Tips 'n' Tricks

Here are some tips and tricks that may help HTML authors when developing sites. They are by no means comprehensive rules, more suggestions of how to accomplish some commonly desired effects.

NOTE : All of these tips were tested in **Internet Explorer** (3.0 beta 2) and **Netscape** (3.0 beta 5a).

Using fonts with <MARQUEE> elements
Using fonts within <TABLE>s
Browser independent background sound
How to load more than one document within a <FRAME> document
Creating an apparent BACK button
Browser independent <MARQUEE>s
How to determine the users browser type
Noisy Links

Using fonts with <MARQUEE> elements

See Also

If you wish to specify any <FONT> settings for an **Internet Explorer** specific <MARQUEE> element, then those settings must be declared *outside* the <MARQUEE> element.

That is :

```
<FONT FACE="Comic Sans MS">
<MARQUEE BGCOLOR="Yellow">This is Comic Sans MS</MARQUEE>
</FONT>

<MARQUEE BGCOLOR="Yellow">
<FONT FACE="Comic Sans MS">This is not</FONT>
</MARQUEE>
```

<MARQUEE> elements can also take standard Style Sheets presentationtechniques, such as those using the <STYLE> element, or attribute.

Using fonts within <TABLE>s

See Also

Within <TABLE> elements, any previous <BASEFONT> or <FONT> settings are ignored and the text will be displayed in the normal text size (as set in the browser preferences) *unless* the <FONT> settings are specified *inside every* <TD> data cell.

For example :

```
<BASEFONT SIZE="5">
<TABLE BORDER>
<TR>
<TD><FONT SIZE="7">Size 7</FONT></TD><TD>Default text size</TD>
</TR>
<TR>
<TD><FONT SIZE="+1">Default size +1</TD><TD>Default text size</TD>
</TR>
</TABLE>
This text is size 5
```

The <BASEFONT> is set to 5, but none of the <TABLE> data cells use this setting. Those that have no explicit <FONT> setting are displayed in the browsers default text size (as set in the browsers preferences).

So, for a table for which every cell needs to be a different size from the default text, requires an explicit <FONT> setting inside *every* <TD> data cell.

This situation is somewhat eased in **Internet Explorer** for which a Style Sheets declaration can be set to globally specify any <TABLE> styles.

Browser independent background sound

See Also

If your page uses background sound, it is now possible to get both **Netscape** and **Internet Explorer** to play that background sound.

Using the following in the HTML for example :

```
<EMBED SRC="clouds.mid" HIDDEN="True">
<BGSOUND SRC="clouds.mid">
```

will play the sound file `"clouds.mid"` in both browsers. The `HIDDEN="True"` attribute in the `<EMBED>` element forces **Netscape** to not display the control console for playing of the sound file, so it behaves exactly as **Internet Explorer** does for the `<BGSOUND>` element and autostarts the sound file, playing it as background accompaniment for the page.

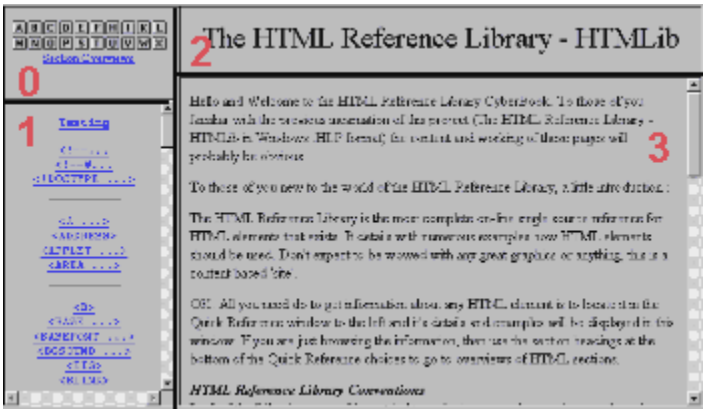
Other sound files can be placed in the document using the `<EMBED>` element and these will play over the top of the background sound when activated.

Multi document loading within a <FRAME> document

See Also

This effect can only be achieved by using either a JavaScript, or VisualBasic Script script function. First, we'll consider the JavaScript script.

Consider the framed document shown below.



(The frame borders have been highlighted with black lines and the red numbers represent the numbers that Scripts uses to locate the frames (***NOTE :*** Frame numbers begin at zero, rather than one)). The Script function uses the frames object from the JavaScript object model to reference two URLS for two specific frames in which to load the specified documents. For instance, a link in frame number 1, that wishes to load two documents with the same link into frames numbered 2 and 3 would call the following script function :

```
<SCRIPT LANGUAGE="JavaScript">
  <!--
  function frame_update(){
    parent.frames[2].location="intro.htm";
    parent.frames[3].location="title.htm";
  }
  // -->
</SCRIPT>
```

The text which is the link that calls the script then uses the following syntax :

```
<A HREF="#nowhere" OnClick="frame_update()">Testing</A>
```

which updates frames numbered 2 and 3 (see screenshot) with the files `"intro.htm"` and `"title.htm"` respectively.

In this example, a HTML fragment is used as the value of the `HREF` attribute. This is to prevent the browser re-loading the document which contains the link, as it would if `<A HREF="">` was used. The link uses the `OnClick` event to call the Script function

NOTE : The above JavaScript solution will work in **Internet Explorer** and **Netscape**. For browsers that don't support JavaScript, the text will be coloured according to any link colourings, but will not be a useable link. This solution should therefore only really be used in situations where the author can guarantee that the user will be using either **Internet Explorer** or **Netscape**. For information on determining the users browser type, see the tip How to determine the users browser type.

The VisualBasic Script solution is very similar to the JavaScript solution, except for syntax differences. The Script function is :

```
<SCRIPT LANGUAGE="VBScript">
  <!--
  function frame_update()
    parent.frames(2).location="intro.htm"
    parent.frames(3).location="title.htm"
  end function
  // -->
</SCRIPT>
```

NOTE : VBScript uses normal brackets '(,)' instead of square brackets '[,]' to delimit the frames number and does not require the '{' characters to delimit the function and semi-colons at the end of each line. The syntax for the link text is :

```
<A HREF="#nowhere" OnClick="frame_update()">Testing</A>
```

This VisualBasic Script example will only work in **Internet Explorer**.

NOTE : Both of the above scripts can also reference the particular frames, by using their frame name, as defined in the `<FRAME NAME="...">` code of the main frame document.

Both of the above examples specify the URLs of the documents to be loaded within the function. Obviously, if there are many links on a page that will load more than one document, the above solutions will be seriously inefficient, as a separate function would be required for each link. A better way is to specify the function as receiving arguments, which are passed to the function in the link. For example, using the following JavaScript function :

```
<SCRIPT LANGUAGE="JavaScript">
  <!--
  function frame_update(URL1,URL2)
  {
    parent.frames[2].location=URL1;
    parent.frames[3].location=URL2;
  }
  // -->
</SCRIPT>
```

will allow any link to call the function, passing it two URLs as arguments. The two URLs passed to the function will be loaded into the specified frames. For example :

```
<A HREF="#nowhere" OnClick="frame_update('intro.htm', 'title.htm')">Testing</A>
```

The similar function in VisualBasic Script would be :

```
<SCRIPT LANGUAGE="VBScript">
  <!--
  function frame_update(URL1,URL2)
    parent.frames(2).location=URL1
    parent.frames(3).location=URL2
  end function
  // -->
</SCRIPT>
```

with the link text being, for example :

```
<A HREF="#nowhere" OnClick="call frame_update('intro.htm','title.htm')">Testing</A>
```

Note the use of the `'call'` keyword to call the function.

Both of the above examples can be extended to use as many URLs as necessary, to update as many frames as necessary.

Creating an apparent BACK button

See Also

Essentially this tip involves accessing the browsers History object. In JavaScript, using the following function :

```
<SCRIPT LANGUAGE="JavaScript">
<!--
  function retrace()
  {
    history.back()
  }
// -->
</SCRIPT>
```

accesses the browsers history object when called from a link such as :

```
<A HREF="#back" OnClick="retrace()">Go back</A>
```

or a button such as :

```
<FORM>
<INPUT TYPE="BUTTON" NAME="GOBACK" VALUE="Go back 1" OnCLick="retrace()">
</FORM>
```

which forces re-loading of the document last viewed in the browser. ***NOTE :*** At the time of writing, this method worked in **Netscape** (even within framed documents), but caused **Internet Explorer** to crash (as did a VisualBasic Script variation of the script).

The history object also takes `'forward'` and `'go(int)'` methods, allowing forward navigation and navigation across any integer (positive or negative) number of pages in the history list.

Browser independent <MARQUEE>s

See Also

At present, **Netscape** does not support the <MARQUEE> element for producing scrolling text effects in HTML. However, using the GIF construction kit, a multi-block 'Banner' style GIF file can be created, which will produce the exact same effect. If the <MARQUEE> that you wish to use, uses a font that may not be present on the users system, using this method can be advantageous, as no font files are required.

For more information on the GIF Construction Kit, visit http://www.mindworkshop.com

How to determine the users browser type

See Also

This tip is possibly misnamed slightly, as using the example script below, it is only possible to distinguish between **Internet Explorer** (3.0 and above) and **Netscape** (2.0 and above) as it uses the `navigator.UserAgent` object to determine the browser type, which can only be interrogated (using non CGI scripts) for these two browsers.

This tip is useful if, for example, you have a set of pages that relies heavily on ActiveX content, that will be unuseable in **Netscape**. A button can be used that runs the browser checking script and re-directs the user to the correct destination, based on their browser type. Any wider browser checking routines require CGI scripting, to trap and interrogate the `HTTP_USER_AGENT` field that is sent by every browser to every server when requesting HTML documents. (Any CGI reference should contain information on this topic).

The following script can be used to determine whether the user is using **Internet Explorer** or not.

```
<SCRIPT LANGUAGE="JavaScript">
function browser_check()
 {
 var BrowserType=navigator.userAgent;
 if (BrowserType.indexOf("MSIE") == 25)
   {location.href="index2.html"}
 else {
   location.href="browser_check.html"}
 }
</SCRIPT>
```

The script is a JavaScript, because it will then check **Netscape** *and* **Internet Explorer**. A VisualBasic Script version would only work on **Internet Explorer**, thus not fulfilling its purpose.

Basically, the script takes the `navigator.userAgent` object and checks to see whether the string `"MSIE"` is at the 25th position in the `navigator.userAgent` string. If it is, then the browser loads the page `index2.html` else it loads the page `browser_check.html`. The two `browser_check.html` strings presented by **Netscape** and **Internet Explorer** are :

```
Mozilla/2.0 (compatible;MSIE 3.0A;Windows 95)
Mozilla/3.0b5 (Win95;I)
```

for **Internet Explorer** (3.0 beta 2, Windows 95) and **Netscape** (3.0 beta 5, Windows 95) respectively.

The above script could easily be extended to trap any of the contents of the two `navigator.userAgent` strings and act accordingly.

A simple button can be included within the HTML document to run the script like :

```
<FORM>
<P><INPUT TYPE="button" VALUE="Check the Browser" NAME="BCheck"
  onClick="browser_check()">
</FORM>
```

Noisy Links

See Also

It is possible to have the browser play sound files whenever the mouse is positioned over a link within a document.

The first method for this tip takes advantage of **Netscape** LiveConnect technology and so only works when using **Netscape**. It also requires the browser to be equipped with the LiveAudio plug-in, which is packaged as standard with **Netscape** from v3.0.

Within the HTML document, the sound file must be included via the plug-in mechanism using the <EMBED> statement :

```
<EMBED SRC="whoosh1.wav" NAME="snd" WIDTH="1" HEIGHT="2" MASTERSOUND>
```

The NAME and MASTERSOUND attributes are specific to the LiveAudio plug-in when using LiveConnect functions. MASTERSOUND specifies that that particular instance of the sound file is to be controlled by any script functions. The NAME attribute specifies a unique identifier for the plug-in, which can be used in script functions. (This example uses the particular WIDTH/HEIGHT settings, because using the HIDDEN attribute, to hide the console only allows the sounds file to be played once and then re-loads the page. Using values anything less than those given, causes the console to be displayed. Both of these ruin the effect.)

Then, the link text which is to control the playing of the sound file specified in the above <EMBED> statement is :

```
Here is a <A HREF="another_page.htm" OnMouseOver="Play_()">link</A>
```

Here, the OnMouseOver event has been used to call the script Play_() (detailed below). This causes the sound file to be played whenever the mouse passes over the link. The OnCLick event can also be used, which causes the sound file to be played when the link is actually activated.

Finally, the script function used to actually play the sound file is :

```
<SCRIPT LANGUAGE="JavaScript">
<!--
function Play_(){
  document.snd.play(false);
  }
// -->
</SCRIPT>
```

Basically, this finds the object in the current document named as snd (which is the plug-in) and uses the play controlling function on it, playing the sound file specified in the plug-in.

NOTE : If the user is using **Internet Explorer** the ActiveMovie control panel will be displayed, as **Internet Explorer** will use this module (if installed) to play files embedded into documents using the <EMBED> element.

However, exactly the same effect can be achieved in **Internet Explorer**, if the sound file is included into the HTML document using the ActiveMovie ActiveX control. To include the sound file, via the ActiveMovie control, use the following syntax :

```
<OBJECT
  ID="snd"
  WIDTH=4
  HEIGHT=4
```

```
  CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A">
    <PARAM NAME="_ExtentX" VALUE="106">
    <PARAM NAME="_ExtentY" VALUE="106">
    <PARAM NAME="ShowDisplay" VALUE="0">
    <PARAM NAME="ShowControls" VALUE="0">
    <PARAM NAME="FileName" VALUE="whoosh1.wav">
    <PARAM NAME="Volume" VALUE="-593">
    <PARAM NAME="Balance" VALUE="0">
</OBJECT>
```

The `ShowDisplay` and `ShowControls` settings serve to hide the default ActiveMovie control panel and display from the user (the `HEIGHT` and `WIDTH` settings are auto-set by the ActiveX Control Pad when the display and controls are disabled). The VisualBasic Script routine is as follows :

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub PlaySound()
  snd.Run
end Sub
-->
</SCRIPT>
```

The function basically uses the `Run` function on the object in the document who's identifier is specified as `snd`. In this case, the ActiveMovie control (via the setting of the `ID` attribute : `ID="snd")`

The link text that forces playing of the sound file is :

```
Here is a <A HREF="another_page.htm" OnMouseOver="PlaySound()">link</A>
```

When the mouse is passed over the link, the sound file is played. As with the first method (for **Netscape**) this ActiveX/Visual Basic Script method can also employ the `OnClick` event in the link to play the sound file when the link is clicked, rather than passed over

NOTE : These methods require the loading of an external sound file, which while fine for local/intranet Web sites, may seriously hamper the performance of an 'open' web site.

LiveConnect is a **Netscape** technology that allows Java applets, JavaScript and plug-ins to communicate with each other. Plug-in modules have to be LiveConnect enabled in order to be able to communicate with other applets/Scripts and so particular details of the use of these methods is plug-in dependent. For more information, visit the **Netscape** web site (at http://www.netscape.com/) or the developer of the particular plug-in you wish to use.

HTMLib Tips 'n' Tricks