Professional Frames in HTML

by Carl Dodson


Comment on this article

This tutorial is both an introduction to frames in HTML, and a guide to proper frame design.

Prerequisites include an understanding of HTML and access to a browser that is frames-capable, like Netscape 2.2 or better, or Internet Explorer 3.0 or better.

Controversial Frames

Frames have caused a lot of discussion since their introduction. On the far right are the Neo-Phobes, who do not respond well to anything radically new on the Internet. These are the people who do not update their browsers, or use third party browsing software that came with their system or Internet service. On the far left are the Techies, who jump at the first mention of new technology. Both of these groups of people are necessary to the stability of any technology. Techies push the boundaries, forcing technology to surge ahead, while Neo-Phobes resist change, forcing technology to remain reverse compatible, and therefore accessible, for as long as possible. Somewhere in the middle exist the Rest Of Us.

The Rest Of Us are people with varied interests and wide ranging tastes. We demand full compatibility. We demand stability. We demand more, and complain when more is delivered at a cost to compatibility or stability. We are people who love the pretty and striking imagery we see on the Internet, but hate the download times it causes. Most of us either love or hate frames, and all of The Rest Of Us have good reasons why we feel the way we do.

Frames are an excellent tool for some applications. They are a miserable choice for others. And like any tool, any programming language, and any utility, when implemented properly they deliver all they promise and more. When used improperly, they drive The Rest Of Us completely nuts.

It is my hope, and the purpose of this tutorial, to provide HTML authors with usable knowledge, not just on how to design and implement frames, but on how to do so *professionally*.

Fact or Myth ?

With the proliferation of websites using poorly designed and implemented frames, it is no real wonder that so many bad feelings have been generated over the use of frames. With the "functionality vs. implementation" question in mind, I have compiled a few of the opinions and replies about using frames.

OPINION : FRAMES TAKE LONGER TO DRAW. THEY ADD SIGNIFICANTLY TO THE DOWNLOAD TIME.

Well, yes and no. A frames-capable browser will see the FRAMESET tags, and draw the frames according to the instructions it finds. This does take about one second longer to do for the browser. However, it is not true that frames add to the download time. The two largest bandwidth hogs in frames are multiple server requests and graphics.

When the user enters a website with frames, the browser sends out a separate request for each HTML document that is being called. This is very important, as the purpose of frames is to display more than one document at a time in the same browser window. When large documents are called by each frame, there can be a noticeable slowdown in the download time.

Another large download increase comes with large or multiple graphics files. There are no rules for determining the proper size for graphics, but generally anything over 25 kilobytes is considered large large. There are several things that the designer can do to speed up download times while still incorporating graphics into the website.

- Consider whether to use GIFs or JEPGs for the images. Under certain circumstances one will produce a smaller image than the other.
- Consider using one image map instead of several smaller images. A single large image will be much smaller than several small images stacked, especially when using GIFs.
- Try to keep the per-page content (including graphics) low.
- Be careful with animations; they can really slow down a website.

Up to a point, large graphics may not present a large problem. Graphics that are designed for a user interface (like a selection table, image map or control panel in one frame) draw only once in a frame, and then they stay static. The user clicks on buttons or hypertext links from the "Selection" frame, and the other frames are targeted with the updating pages. However, many designers place huge image maps or lots of graphics into most of the frames, forgetting that these pages must redraw each time they are updated. Having the ability to put lots of things on different pages is NOT a license to over-burden users of your website with graphics. When I view a framed website I often opt for text versions when available, simply because I don't like to wait for the pages to redraw. Imagine what it's like for a user with a slower modem.

Keep the graphics small, elegant, and simple. Put larger complex graphics into frames that do not update often, if at all. This keeps the long waits to a minimum. Ultimately, measure how long it takes for a user on a modem to download ALL of the text and graphics from each frame that is visible on your website. The following table gives examples of a few simple ratios to use when deciding how much content to put on a page:

Speed Ratios	
Modem	**Kilobytes Per Second**
14.4	2
28.8	4
36.6	6
56K	8

When totaling the kilobytes for your pages, be sure to add in the file sizes of any JPG or GIF images, images maps, unique icons, animations, and of course the text content. It is a good habit to total these up for *each page* of the website.

OPINION : PEOPLE WHO USE FRAMES ON THEIR WEBSITE DON'T TAKE INTO CONSIDERATION THOSE OF US WHO DON'T HAVE FRAMES-CAPABLE BROWSERS.

Sadly, this is true. And as a designer myself, I find it difficult to justify spending the time it takes to build a framed website, then provide backward compatibility for older browsers that cannot use frames. This is probably the best reason why to use tables instead of frames.

Yet, there are times when you can, and possibly should use frames. Tables are wonderful devices but they do have practical limits. For instance, it is a little cumbersome to include a large selection table on every page you design, where a frame displays the menu once and forever. Tables limit what you can do with graphic interfaces (buttons), for the same reason; unless you are using server-side #include tags, you must include the table in each document where you want it displayed. Maintaining such a website would be frustrating at best. And where a frame can be designed to stay in place, always displaying an array of options, a table always scrolls with the page. A small point, perhaps, but if you are designing for effect, the frame provides an obvious advantage, especially for websites with lots of pages and content.

The best answer to this problem is in the planning of your website. Exactly WHO do you want to see your website? At the time of this writing, it has been suggested that fewer than 5% of Internet users are still clinging to browsers that are not frames-capable. If you depend on lots of users, though, you may need to read that statistic as *fully 5% of the people on the internet cannot see your website*. One good side effect of designing websites with frames and noframes options: Since you probably want to use ONE set of pages for both, instead of writing two separate websites, you are forced to be conservative, plan carefully, target your pages correctly, and test thoroughly. That usually translates to a much better website for all users.

OPINION : FRAMES DO NOT WORK. FRAME DESIGNERS DON'T TEST THE SITE THOROUGHLY, AND THE USERS ARE STUCK WITH THE ERRORS.

This, unfortunately, is sometimes true. Designing a website with frames is much like writing an application in a programming language. The links you include on pages displayed in a frame must now be told where they are to be drawn. This can get very confusing for the designer, and often results in poorly tested websites. Anyone who has ever clicked on a link and watched a third or forth set of DUPLICATE frames ("sub-frames" complete with duplicate information) slowly draw in the browser knows "frames frustration" to the fullest.

Once again, this and other errors are entirely preventable with patience and testing. A good understanding of how and where to target pages is required; if a designer does not know or is not willing to learn proper page targeting concepts, they have no business designing websites with frames.

OPINION : FRAMES ARE CONFUSING. THEY ARE NOT INTUITIVE, AND THEY MAKE IT HARDER TO NAVIGATE WEBSITES.

This is a fact if the designer has not taken the time to consider what users will do once they are looking at the website. Frames can be very confusing, especially if they are not laid out in an intuitive form.

The majority of users have application interfaces that display controls in a certain format: scroll bars at the bottom and right sides of the window, menus and buttons at the top, etc. Take that "uniformity" into consideration when designing the layout for your website.

- Try to avoid putting vertical frames down the right side of the window. They are not that intuitive, and can be very irritating to users who occasionally miss the scroll bar.
- When you utilize small frames in "remote areas" of the window, remember that updating information in them may not be readily visible to users. If the smaller frame updates with buttons or hypertext links, place a title somewhere in the frame that updates as well. This gives the user more visual feedback.
- Be very critical when implementing "floating" frames. The functionality of floating frames is perhaps questionable to some designers, and it is beyond the scope of these articles to focus on specific uses for them. However, more than any other frame, their placement on the website is critical. Since they are not limited to boundaries determined by existing frames, floating frames are much more visible than regular frames. As such, they can become potentially confusing OR irritating very quickly.

As we have explored here, nearly every major complaint about frames can be overcome with a little extra effort in the design stage.

Understanding FRAMESET tags

<FRAMESET> ... </FRAMESET>

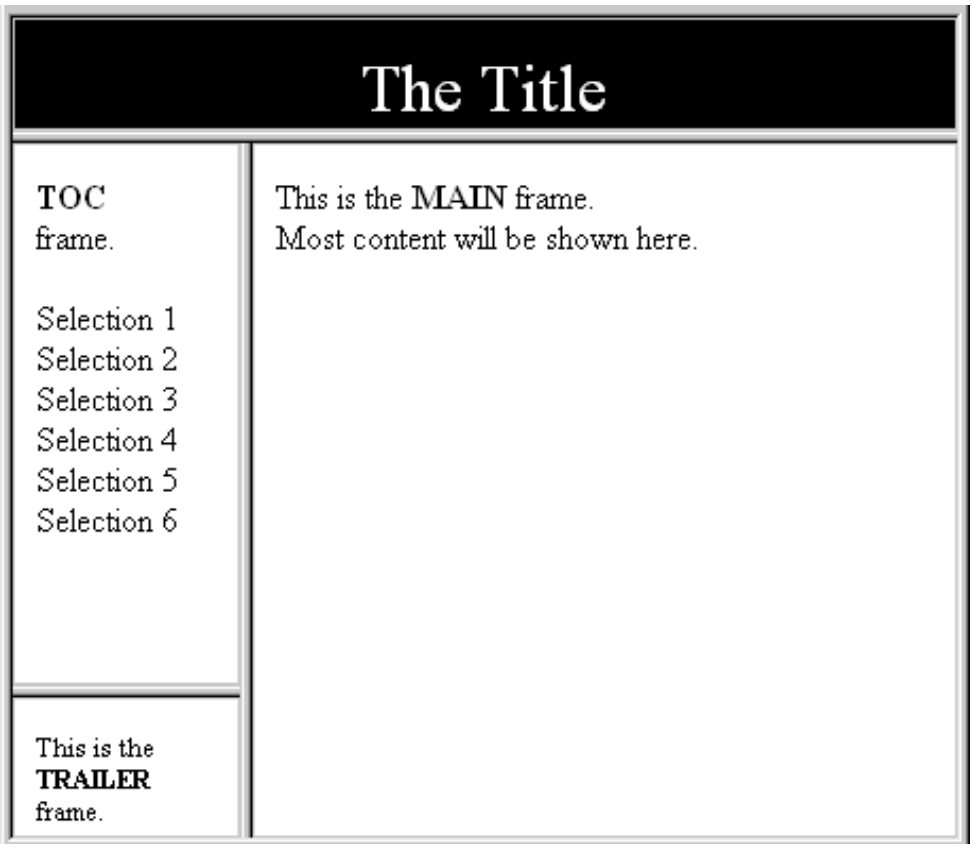
The frameset tag is a container tag, meaning it has a beginning tag **< >** and a ending tag **< / >**. Tags and elements that appear between the frameset tags are interpreted by the browser as instructions for drawing, formatting and controlling the frames. Following is an example for a common frames layout using FRAMESET tags:

Code Example #1

```
1)    <FRAMESET COLS="100%" ROWS="10%,*">
2)        <FRAME NAME="TITLE" SRC="title.html" SCROLLING="no" NORESIZE>
3)        <FRAMESET COLS="20%,*">
4)            <FRAMESET ROWS="85%,*">
5)                <FRAME NAME="TOC" src="toc.html" SCROLLING="auto">
6)                <FRAME NAME="TRAIL" src="trailer.html" SCROLLING="no">
7)            </FRAMESET>
8)            <FRAME NAME="MAIN" SRC="welcome.html" SCROLLING="auto">
9)            </FRAMESET>
10)   </FRAMESET>
```

This probably look a bit confusing, so we will take it one line at a time. Note that each line is numbered.

This particular example creates a view with four frames in a browser window:



In the spirit of "simple is usually best," I have chosen to design a layout that is simple for the user to read and respond to, AND simple for me to update and maintain over time. The individual frames in the example above are labeled according to their purpose :

- The TITLE area at the top will hold a simple text name for this website. Most likely, this frame will never change while a person is viewing the website.
- The TOC (Table of Contents) frame will hold links to sections or webpages. This frame is also going to be static; it will always display the same information.
- The MAIN frame is the main view. This view will react to the links listed in the TOC frame, redrawing a new HTML page when requested.

The titles I have used are not important, but the layout is. Think in terms of "frustration factor" when designing the frames layout: The more frustrated a user is when navigating a website, the less likely the user is to return. One of the keys to successful frames layout is to give the user an easily learned interface to maneuver through.

The first thing that can be gathered from the code example above is the FRAMESET declaration. FRAMESET tags are container tags; they have a beginning tag (<FRAMESET>) and a terminating tag (</FRAMESET>). FRAMESET tags can also be nested within each other; in Code Example #1, I have nested the FRAMESET tags three levels deep.

Line by Line

Line #1

Line #1 does a lot to set up the initial view:

1. Declares a Frameset
2. Sets the initial **COL**umn**S** to "100% of available browser width"
3. Sets the initial **ROWS** to "10% of available browser height for the first row, and * for the second row"

That asterisk (*) may throw you at first. It means "any viewing area left over". So here, I am saying, "Allocate any leftover viewing area to the second ROW".
You can use the asterisk in both COLS and ROWs. Additionally, you can use more than one asterisk; the browser will divide the remaining space equally.

Line #2

Line #2 declares a FRAME. FRAME tags are NOT containers; note that there is no terminating tag. The FRAME tag gives a NAME to the frame being created, and then tells the browser what file to put into the new frame. FRAME tags are acted on by the browser in the order that they are read. Obviously, it is important to maintain some control over what file gets displayed in which frame. This is one very good reason to come up with easy to remember names for your frames during the layout stage.

Line #2 goes on to tell us some important information :

- FRAME NAME="TITLE" - The name of this frame is TITLE.
- SRC="title.html" - The source file for this frame is called title.html.
- SCROLLING="no" - Scrolling is to be disabled.
- NORESIZE - Re-sizing the frame from the default setting of 10% is not allowed.

There is one additional piece of information that Line #2 tells us. Look back at the ROWS element in Line #1. There are TWO rows being described here. Each row is to be broken into a frame in the browser window. Line #2 applies only to the FIRST of those two frames; we can read that just by its placement. That's a lot of information for one line, but all of it is essential for the proper behavior of our website.

Line #3

Line #3 is another FRAMESET tag. Just like Line #1, we are now creating a new set of frames. The difference is the placement; we are taking an already existing frame (declared in the ROWS statement in Line #1) and breaking it into TWO more frames. The COLS="20%,*" element creates these frames by splitting the space from top to bottom, making a left side (20% of available space) and a right side (the rest of the available space).

Line #4

Line #4 is the third of our nested FRAMESET tags. The ROWS="85%,*" element splits the available space into two parts again, this time from left to right, creating a top portion and a bottom portion.

Can you see a pattern emerging here? COLS elements tell the browser to split the space into *columns*, ROWS split the space into *rows*. The placement of the FRAMESET tag tells the browser WHICH portion to apply the ROWS or COLS to.

Line #5 and #6

Lines #5 and #6 are both FRAME tags. Just like Line #2, they tell the browser what these to new frames are called, and which files should be displayed in them. Note that I have set the TOC frame to SCROLLING="auto". If the text contained in the file toc.html cannot fit into the TOC frame as it is drawn, the browser will automatically put Windows scroll bars at the right and bottom sides of the TOC frame, allowing the user to scroll around the page if necessary. It is important to properly gauge the amount of space you will need to allocate for the size of some frames. Some careful observation here will increase the "readability" of the whole website.

Line #7

Line #7 is a terminating tag. We are finished describing the TOC and TRAIL frames to the browser.

Line #8

Look at Line #8. It may look a little out of place, since there isn't a FRAMESET tag right before it. Well, technically there is, and this convention is why developers indent their code. Look back at Line #3. Remember the COLS="20%,*" element? Well, frames TOC and TRAIL were described in the same block of code (as shown by the indention). They are drawn in that first 20% of the browser that Line #3 describes. The remaining space, which we described using the *, is still open until Line #8 names it. Line #8 gives this frame the name MAIN, and points the browser to the file welcome.html.

Line #9 and #10

Lines #9 and #10 should make sense to you. We are terminating the container tags from Lines #3 and #1. I have again indented here to make the code easier to understand.

Finishing up

Following are some general facts about FRAMESET tags:

- The FRAMESET group should usually be the first page in the website. If you want to set up a website so that the homepage is displayed in a frame, then the default.html file should be the FRAMESET group.
- COLS and ROWS can also be given the size of the frame in PIXELS. For example, you can change Line #3 to read

<FRAMESET COLS="170,*">

This would tell the browser to make the first column 170 pixels wide, and give the rest of the space to the second column.

- The SCROLLING element can be set to "yes" (always draw scroll bars), "no" (never draw scroll bars) and "auto" (draw scroll bars when needed). These settings can have a slightly different effect depending on the user's browser. Normally this doesn't make much difference, but it is a good idea to allow for small differences in browsers (like Internet Explorer and Netscape) if the difference makes your website un-viewable or difficult.
- Frame borders are handled in slightly different ways by Netscape and Internet Explorer. Specifically, turning them on or off, and frame colors use different element tags, or are not supported. Consult the Netscape or Microsoft documentation for details.

Here are some simple ideas for exercises for practicing frames layouts.

- Change the COLS and ROWS percentages, and see what effect that has on the browser.
- Change the COLS and ROWS to whole numbers (instead of percentages) and explore how setting the sizes in pixels alters the way you think about the layout.
- Practice laying out the frames by moving the placement of the FRAME elements. Remember that each declaration in a ROW or COL element must have an accompanying FRAME element.

The TARGET tags

- <BASE="">
- <TARGET="_self">
- <TARGET="_blank">
- <TARGET="_top">
- <TARGET="_parent">

All of the above tags are preceded with an underscore character (_). Netscape refers to these as "magic tags". Magic tags are used to force a new page to draw in a specific named window.

The BASE tag

Technically, the BASE tag is not restricted to frames. The BASE tag is placed into the <header> ... </header> container tag. It helps the browser resolve relative paths and partial URLs. The implementation used in frames allows for setting a "default" target frame. Here is an example of what the header text would look like with a BASE tag:

```
<HEAD>
    <TITLE>Targeting Frames</TITLE>
    <BASE TARGET="MAIN">
</HEAD>
```

To test the BASE tag, create a new HTML document and name it whatever you like. Open the toc.html file and change the header tags to look like the code above. Pick one of the "Selection" texts, and anchor it to point to your new page. For this example, I anchored Selection 4 to point to a page called SELECTION4.HTML. Save the files, and load the file DEFAULT.HTML in your browser.

See what happens? The TOC.HTML file has a BASE tag in the header. The BASE tag is telling the browser, "If you come across any un-targeted URL links on this page, then just assume that they are supposed to go to the MAIN frame."

The BASE tag has other uses; you may want to read through the HTML 3.2 specifications to fully exploit them.

Making Magic with the TARGET Element

The TARGET element is pretty powerful, and somewhat dangerous. Incorrectly targeted pages will quickly make a website un-readable and confusing. Using the BASE tag on certain pages will help this situation to a point, but the wise designer doesn't rely on BASE tags to "clean up" messes. TARGET tags are provided to properly direct pages to their intended viewing destination. All of them are equally important, and we will cover them here.

TARGET elements are placed into an anchor tag. They are used to tell the browser WHERE to draw the document that this anchor points to.

TARGET="_self"

This attribute tells the browser that the page this anchor points to should be drawn into the current frame. The effect is that the chosen document will display into the same frame area containing the selected link.

TARGET="_blank"

This attribute opens the selected URL into a new browser window. This is especially useful when you have directed the user to another website or document not within your own. You could target the off-site destination into an internal frame, of course, but it is somewhat strange looking to have a completely different website (with different color themes, controls and content) drawing into a frame on your site. Also, if the targeted website uses frames, you will end up with a confusing jumble of frame bars inside the browser window.

A better option is to target the website to a new window, leaving your website completely intact. Of course, you now have put the user completely out of your control until they choose to return to your website. Use some discretion when linking to off-site destinations if you are commercially dependent on getting your users to navigate your website.

TARGET="_top"

This attribute is handy. The _top attribute instructs the browser to draw the targeted document into the current browser window. This is useful for "erasing" the frames to make new ones, replacing one layout with another one, or simply making more viewing space. I use it most for returning to the home page of the website. By specifying the "DEFAULT.HTML" page as the target, I can quickly return the user to the original welcome page.

TARGET="_parent"

Explaining this one in simpler terms isn't easy. The _parent attribute tells the browser to draw the targeted document into the *parent* window of the current document. The most likely effect of this is that the page will be directed to fill the whole browser window, not unlike the _top attribute does. To test this attribute, we will make some changes.

```
<A HREF="default.html" TARGET="_self">default.html</A>
<A HREF="welcome.html" TARGET="_parent">welcome.html</A>
```

Copy this code into the WELCOME.HTML page and save it. Now point your browser to the DEFAULT.HTML page. You should get everything laid out normally, with the new anchor links showing in the MAIN frame.

Now, click on the default.html link. When the new frames are finished drawing, click on the default.html link again. Do the preceding one more time. You should now have three full sets of frames nested inside each other, and each frame should contain the document specified to it.

Now, click on the WELCOME.HTML link. Do it twice more.

See what is happening? Each time you select the WELCOME.HTML link, the page redraws itself into its **parent**. The browser "knows" which frame contains the parent document, and reacts accordingly.

TARGET="FRAME_NAME"

Perhaps the most useful (or at least most used) is the custom TARGET tag, the one that you as a designer use to direct selected pages to a specific frame. By placing a TARGET element in an anchor, we can direct the page to draw in the indicated frame. For Example: In the previous tutorial, we saw how to assign a name to a frame:

```
<FRAME NAME="MAIN" SRC="welcome.html" SCROLLING="auto">
```

In the above code snippet we have given this frame the name "MAIN". In your editor, create a new page, and name it whatever you like. Open the toc.html file, and make an anchor out of one of the "Selection" texts. Point it to the new page. Change the anchor so it looks like this:

```
<A HREF="mypage.html" TARGET="MAIN">Selection</A>
```

Save all files, and open default.html in your browser. Click on your new Selection. The new page you made should draw in the MAIN frame.

Here are some simple ideas to try:

- Target the different lesson files into different frames
- Make up new HTML pages, and anchor links to them on the TOC.HTML page. Practice targeting them with the different TARGET attributes.
- Make links on some pages to outside websites you know of. Observe how these interact when targeted to different frames and windows.
- Create new pages, and anchor them on the toc.html page to selections. Practice targeting the anchors to different frames.

The Compatibility Issue

This was a very controversial subject that has since become much less important. As recently as 1996, compatibility of HTML features and browsers was a major issue, with Internet Explorer and Netscape each supporting certain features that the other did not, and each competing to support the newer and better features. There were many other browsers that did not support many of these features, like frames. Some of these other browsers have been around longer than Netscape and Internet Explorer, and therefore had a measurable share of the market.

There are still other browsers in the market, but the general acceptance of Netscape and Internet Explorer as the industry leaders have lead to the increase of users obtaining and using these products. This translates directly to fewer users using non-frames-capable browsers. Whether this is a positive or negative trend is well beyond the scope of this tutorial. However, to be complete we are going to talk for a moment about providing backwards compatibility with older less-capable browsers.

There is no science to making a framed website more compatible with older browsers. Largely, it is a matter of how much work the designer is willing to do. Since designing a website can be very time intensive, deciding on compatibility is an issue best decided right up front. It isn't much fun to finish a clean design, then go back and re-plan from the beginning.

<NOFRAMES> ... </NOFRAMES>

The NOFRAMES tag is placed on the same page as the FRAMESET tags are. Whatever is put into the NOFRAMES container gets displayed in browsers that are not capable of using frames.

The observant reader is already asking the question, "How is it that an older browser knows what the NOFRAMES tags are and doesn't know what the FRAMESET tags are?"

The answer is: they don't. Browsers work by parsing the HTML document. Whatever the browser understands, it acts on; whatever it doesn't understand, it ignores. Most browsers would display the FRAMESET page as blank. The information is there, but the browser doesn't know what to do with it. Since the browser cannot display it as text and cannot act on the commands, it simply rejects them.

This includes the NOFRAMES tags; the non-frames browser rejects them as well. Any tags that are placed afterward (presumably the designer has used more standardized HTML code here) are likely to be understood, and the browser will act on them accordingly.

Why doesn't a frames-capable browser display all of the material in the NOFRAMES tags? Because frames-capable browsers understand that whatever is in the NOFRAMES container is to be ignored entirely. The browser doesn't care where the NOFRAMES container is placed in the document, as long as it is outside of the BODY container tags. Whatever is in the NOFRAMES container tag is skipped, and the rest of the document is processed.

Unfortunately, that is not all that must be done to make the website compatible. The design and layout of the website has probably been optimized for frames. The designer usually creates a "path" for the user's eyes to follow. Even if the designer has allowed for compatibility from the beginning, providing the same visual flow to the user as they move through the website is tricky. Here are a few ideas to get around this:

- Duplicate the entire website, one using frames, one using another method (like tables). Provide a front page that allows the user to pick whether or not they wish to use frames.
- Duplicate the entire website, one using frames, one using another method. From the FRAMESET page, use the NOFRAMES container to provide a set of selections to other pages. Put a copy of this selection on EVERY page in the no-frames version of the website.
- Create one frames website. From the FRAMESET page, use the NOFRAMES container to provide a set of selections to other pages. Create your pages with a selection that links back to the default page.
- Create one frames website. On all pages include a table with links to all other pages. A better way would be to use server side #includes, so updating the tables would be easier.

These ideas assume the website is designed with a Table of Contents or selection list. Other ideas may be more applicable when designing framed websites that use other layouts.

To NOFRAMES or not to NOFRAMES ...

Many designers ask "Why bother? Why put so much time into compatibility for only a small percentage of the Internet community?"

This is a valid question. If you don't care about reaching the users who cannot see frames, then don't waste the time. If you are building a commercial website, weigh the cost of development versus the potential customers.

Conclusion

Although frames are not a "new" technology, they are still a controversial one. There are many reasons not to use frames for website development; lack of information should not be one of them. I hope that you find this information useful.
