CHAPTER 22

Creating Frames

In This Chapter

An introduction to frames

Problems with framed sites

Developing the master frame document

Targets

Creating FRAMEs

Enhancing navigability

NOFRAMES

Adding inline frames (IFRAMEs)

Frames are an exciting, though controversial, way to enhance the navigational possibilities of your page. Frames give you an interesting kind of control over the layout of your site because parts of your page can remain static, while other parts change. This chapter explains how frames work, why they are controversial, and how best to use them. Finally, at the end of this chapter is a section about a newcomer to frames, the inline frame, which adds frames to your site in a way that doesn't carry all the liabilities of traditional frames.

Introducing Frames

Frames are a way of dividing your browser window so it can hold more than one logical page. When you create a framed site, the page has a URL, and then each frame within the page has its own URL. Because each frame within a framed site has its own URL, you can load other pages into only a part of your browser, keeping the rest of your frames static.

Figure 22-1 shows one way of dividing your site using frames. This is a common way. At the top, in the banner, you put the site name, logo, ads, and so on. In the lower-left frame, the navigational tools (buttons, a table, or text) are placed. Finally, the content goes in the main frame. The content can change, but banner and navigational tools frames stay the same.

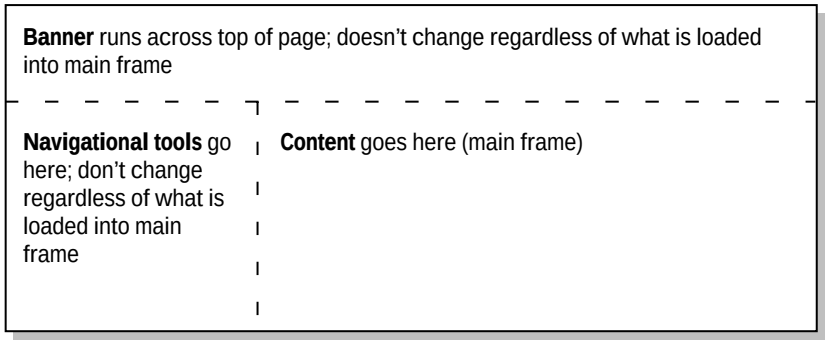

Figure 22-1: Sample layout for a framed site

Developing the Master Frame Document

So how does the browser know it needs to load more than one URL and where to put each one? You create a master frame document. Instead of a `BODY` element, you create a `FRAMESET` element or, as in the previous example, nested `FRAMESET` elements.

The master frame document is a standard HTML document, but there is no `BODY` element. Instead, there is a `HEAD` (and you should include all the usual `HEAD` information, including `META` elements and a `TITLE` element), and a `FRAMESET` element. The `FRAMESET` element is defined as follows:

Frameset `<FRAMESET>`

Start Tag: Required

Content: `FRAMESET`, `FRAME`, and `NOFRAMES` elements

End Tag: Required

Attributes: `id`, `class`, `title`, `style`: defined previously

`rows`: used to indicate the widths of the rows in order from top to bottom; the default is 100%, meaning one row; you can specify the widths in pixels, a percentage of the browser window, or relative width

`cols`: used to indicate the widths of the columns in order from left to right; the default is 100%, meaning one column; you can specify the widths in pixels, a percentage of the canvas, or relative width

`events`: defined in Chapter 48

Rows only

If you define only rows, you will have a framed document that looks like Figure 22-2.

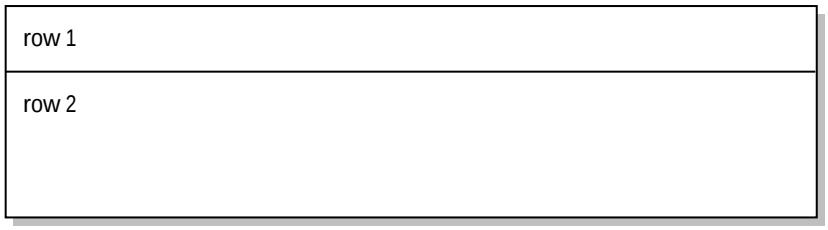

Figure 22-2: A framed page with two rows

The widths of the rows in Figure 22-2 would be determined by the values you specify in the `rows` attribute. The following `FRAMESET` shows using a percentage to indicate the size (pixels and relative sizes are discussed in the "Columns Only" section).

```
<FRAMESET rows="15%, 85%">
</FRAMESET>
```

Columns only

Or, you might want to define a framed document with only columns, as in Figure 22-3.

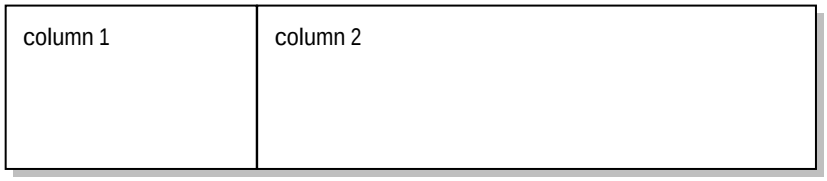

Figure 22-3: A framed page with two columns

The widths of the columns in Figure 22-3 would be determined by the values you specify in the `cols` attribute. The following FRAMESET might have created Figure 22-3. The number 200 indicates the number of pixels (which might be dictated by the width of a graphic you are including in that column). The asterisk (*) tells the browser to use the rest of the space for the second column.

```
<FRAMESET cols="200, *">
</FRAMESET>
```

Both rows and columns

If you define both rows and columns, you end up with a grid-like framed document. Figure 22-4 is one example of a framed document with both rows and columns.

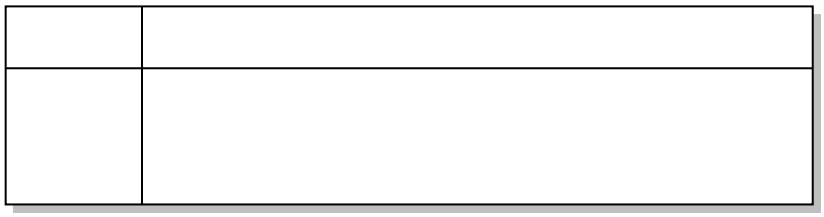

Figure 22-4: A framed document with both rows and columns

Figure 22-4 might have been created with the following HTML:

```
<FRAMESET cols="200, *" rows="1*, 5*">
</FRAMESET>
```

In the previous definition, the first column will be 200 pixels wide, regardless of how wide the page is. The second column will be the rest of the page. The first row will be one part of the height of the page; the second column will be five parts of the height of the page. The browser will calculate that the page must be divided into six parts and assigned to the rows in that proportion. The 1*, and 5* mean 1 part and 5 parts, respectively.

Definition

Relative versus Absolute widths and heights. You can define cols and rows using both relative and absolute measurements. If you use absolute measurements, you need to know exactly how many pixels you want for the width or height. If you use relative measurement, you specify the percentage or the proportion of the browser window you want the frame to fill.

Tip

A warning is in order about using absolute values for frame widths. If you specify an absolute column width and visitors to your site don't have their browser in full-screen mode, they might only see the left-hand column and not even realize a right-hand column (or more than one) exists. If possible, use relative values for column widths and row heights.

Nested FRAMESETs

What if you want to create a framed document, like the example in Figure 22-1? After all, that is probably the most common model on the Web today. Use nested FRAMESETs. The outer FRAMESET will define the rows. The inner FRAMESET divides the second row into two columns.

```
<FRAMESET rows="20%, 80%">
         ... FRAME element for first row ...
         <FRAMESET cols="25%, 75%">
                ... FRAME element for first column ...
                ... FRAME element for second column ...
         </FRAMESET>
</FRAMESET>
```

Targets

The browser needs to know three things. First, how much space you want it to allocate for each of your frames. You have already told it this with the FRAMESET element. Second, how you want to refer to each of those frames: as target names. And third, how to populate each of those frames initially.

With the FRAME element, one of the attributes is name. With the name, you specify how you want to refer to the frame. This is called the *target name* of the frame. You can direct other links into this target. For example, if you have navigational tools in one frame, you can have the destination of those links load into your main frame by specifying the target attribute on the A element.

Consider the following master frame document:

```
<HTML>
<HEAD>
  <TITLE>This is the title of my site</TITLE>
  ... lots of META elements here ...
</HEAD>
<FRAMESET rows="200, *">
  <!-- creates first row as banner -->
  <FRAME name="banner" src="banner1.html">
  <FRAMESET cols="25%, 75%">
     <!-- creates column on left for navigational tools -->
     <FRAME name="navigate" src="navigational-tools.html">
     <!-- creates main frame for content -->
     <FRAME name="main" src="home.html">
  </FRAMESET>
</FRAMESET>
</HTML>
```

Now, look at a possible version of navigational-tools.html, the file containing the navigation buttons (in this case text) and the links. Notice how the A element for each of the links includes a target attribute directing the results of the link to load into the main frame.

```
<HTML>
<HEAD>
  ... TITLE and META elements here ...
</HEAD>
```

```
<BODY>
  <A href="products.html" target="main">Products and
Solutions</A>
  <A href="custsvc.html" target="main">Customer Service</A>
  <A href="white.html" target="main">White Papers</A>
  <A href="jobs.html" target="main">Job Postings</A>
</BODY>
</HTML>
```

You must start your `target` names with letters. After that, you can use any of the regular, acceptable characters. Some `target` names begin with underscore (_). These are the reserved `target` names HTML makes available to help you with your frames. Using reserved names gives you a way to reference your frames relatively. Even if you change the `target` names of your frames, the reserved names still work.

- **blank.** This causes the results of the link to be loaded into a new browser window. Unless the document referred to by the src attribute is a master frame document, the new page won't use frames.
- **self.** This causes the results of the link to be loaded into the same frame as the A element, which created the link. In the previous example, you would use the _self target for links you have within the main frame.
- **parent.** This causes the results of the link to be loaded into the FRAMESET parent of the current frame. If there is no FRAMESET parent, the results of the link are loaded into the same frame.
- **top.** This causes the results of the link to be loaded into the full browser window, canceling frames. When would you want to use this? If you have a "back to home" link on the bottom of your main screen, you'd want it to point to the master frame document with a target of _top. This way, you wouldn't end up with the problem of unintentionally nested frames.

Creating FRAMEs

Finally, having told the browser how much space to allocate for each frame and having given each frame a name, you need to tell the browser what to put into each frame. You do this with the `FRAME` element. You've already seen the `FRAME` element at work.

Frame `<FRAME>`

Start Tag: Required

Content: Empty

End Tag: Forbidden

Attributes: id, class, title, style

name: used to indicate the name to be associated with this frame (the target name)

src: used to specify what should initially be loaded into this frame

longdesc: specifies a link to a longer frame description using the title attribute, and may be particularly useful for nonvisual user agents.

frameborder: tells browser to draw a border around this frame (1, the default value), or not to draw a border (0); if the adjoining frame has a frameborder, it will appear between the unbordered frame and the bordered frame

marginwidth: width of margin in pixels (margin between frame contents and borders on sides)

marginheight: height of margin in pixels (margin between frame contents and borders on top and bottom)

noresize: Boolean attribute (meaning if you want to indicate noresize, you just include the attribute without a value) that tells the browser this frame is not resizable by the visitor

scrolling: There are three values for this attribute: yes, no, and auto. Choose no if you know the contents of the frame will always fit into the frame (say you have column width and row height set in pixels to fit the image in the frame). Choose yes if you want there always to be a scroll bar on the side of this frame. Choose auto if you want the browser only to display a scroll bar if the content doesn't fit into the frame.

You must define a FRAME element for each frame you want to create. The FRAMESET element tells the browser how many frames and how much space to give them. The FRAME element names each frame and gives it an initial value.

When defining FRAMEs, you can specify URLs using either relative or absolute file names. Just as with any other reference element (A or IMG), you can specify both local and remote file names. It is probably better to refer to local file names with relative file names. Remote file names require fully qualified URLs.

Enhancing Navigability

Frames are a great way to ensure that visitors always have navigational tools right on the screen where they can see them. Right? Not necessarily. One thing you should be cautious about is creating a page you plan to use only in the main frame, when a search engine might index it as a stand-alone page. Why is this a problem?

Because when visitors come to that page from a search engine, they have no way to get to the rest of your site. Does this mean you need to put navigational tools on every page? Not necessarily. What you should have on every page is a link back to the front page of your site. That link must have a target of _top to work properly.

If you don't include a target of _top, you will have a problem with your site self-nesting when anyone clicks that link if the main page is actually part of the framed site.

If your site looks like Figure 22-5, and someone clicks the `[Go to Home Page]` link you rightly have at the bottom of every page, but you forget the `target="_top"` attribute-value pair in your link, visitors get the results shown in Figure 22-6.

Figure 22-5: Your framed site

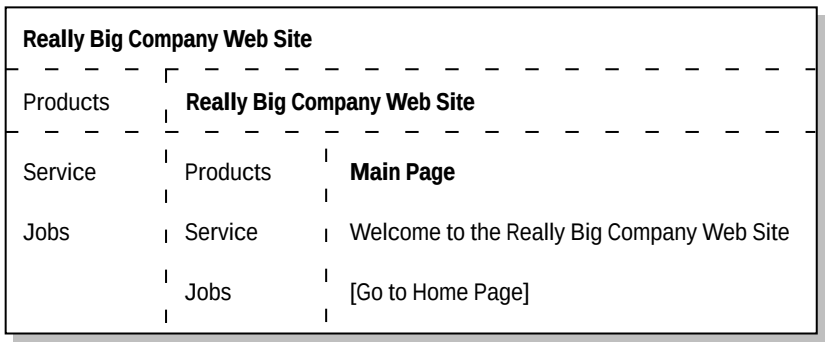

Figure 22-6: Your framed site with unintentional nesting

NOFRAMES

You can place one more element in your FRAMESET element. This is the NOFRAMES element. The NOFRAMES element is one you should include in the outermost FRAMESET, after you have defined all your other FRAMESETs and FRAMEs. NOFRAMES contains the page you want your visitors to get if their browsers don't support frames, or if their browsers support frames but are configured not to display frames. It could be as simple as a message telling them to upgrade their browsers or enable frames to see your site properly.

If you do use the NOFRAMES element, and you should, be sure the text you include is something you wouldn't mind seeing on a Search Engine Results page. Search engines are notorious for using this text as your synopsis when indexing a site. This won't help attract visitors to your site.

```
<NOFRAMES>
Over the Web provides robust, turn-key modules for your
professional Web site.  The Over the Web site is only
accessible with a frames-capable browser. Please upgrade your
browser at one of the following locations:<BR>
<UL>
<LI><A href="http://www.netscape.com/">Netscape
Communications</A>
<LI><A href="http://www.microsoft.com/ie">Microsoft</A>
<LI><A href="http://www.aol.com">America Online</A>
</UL>
</NOFRAMES>
```

More likely, it will be the front page for the scaled-down version of your site that doesn't require frames. It's up to you to decide if the expense is justified to maintain a non-frames version of your site.

No frames <NOFRAMES>

Start Tag: Required

Content: HTML for non-frames page

End Tag: Required

Attributes: id, class, title, style, lang, dir

Adding Inline Frames (IFRAMEs)

Inline frames give you the best of both worlds. They provide many of the benefits of frames without the disadvantage of having navigation in more than one frame. Inline frames do not look like frames at all. No border separates an inline frame from the rest of the page. Inline frames also do not have the capability of independent scrolling.

An inline frame is like a server-side include. You refer to the HTML file you want to appear in the inline frame, and when the page is rendered, it appears as if the inline frame were part of the main HTML document.

Definition

Server-side include. A *server-side include* is a clever way to reuse HTML. If all your pages take advantage of the same header and footer, you can put the header HTML and the footer HTML into separate HTML files and *call* them from all your documents. The syntax of using a server-side include varies from Web server to Web server, so check with your systems administrator if you want to use these. When the page is rendered in the browser, the server-side include HTML (the HTML in the separate file) appears in the main HTML document as if it were typed there. The advantage of using a server-side include is that you can change only the HTML in the server-side include and have it reflected on every page that calls the server-side include.

The principal disadvantage of using inline frames is they are not well supported by the browsers in use. Inline frames are a recent addition to the HTML elements. Currently, only Internet Explorer 4 and newer browsers support inline frames. Inline frames and `<IFRAME>` are identical. Browsers that don't support `IFRAME`, or have `IFRAME` support turned off, will display everything contained between the `<IFRAME>` tag and the ending `</IFRAME>` tag in the current window. Those browsers with IFRAME support will link to the SRC attribute for the URL of the page to display in the inline frame, and ignore text contained in the `IFRAME` tags.

Inline `<INLINE>`

Start Tag: Required

Content: Empty

End Tag: Required

Attributes: `id`, `class`, `title`, `style`, `longdesc`

`name`: name of this element; useful for scripting

`src`: required; the URL of the page you want inserted into the inline frame

`frameborder`: defaults to "1"; set to "0" if you don't want a border

`marginwidth`: width of margin in pixels for both sides

`marginheight`: height of margin in pixels for top and bottom

`scrolling`: yes, no, or auto; you only need to worry about this if you set the height and width to be smaller than the known contents of the URL

`height`: height of the inline frame in pixels

`width`: width of the inline frame in pixels

`align`: deprecated; used to specify alignment on the page

From Here

Jump to Chapter 33 to learn about CSS positioning options.

Proceed to Chapter 23 to learn more about grouping elements with `DIV` and `SPAN`.

Summary

In this chapter you learned about frames. You learned how framed documents work—with a master frame document, targets, and frames. You learned how to define horizontal frames, vertical frames, or some combination of both. This chapter also discussed how your screen gets divided: using absolute measurements, per-centages, or fractions of the screen. You learned about naming your frames, directing the results of links into specific frames (targets), and reserved target names. This chapter also covered how to define the initial values of your frames.

One of the controversial aspects of frames is that all browsers don't support them. Using the `NOFRAMES` element, you learned how you can make this problem transparent to your site visitors. Finally, you learned about inline frames and how they help you get around some of the drawbacks of using traditional frames.

✦ ✦ ✦