12

CHAPTER

Understanding HTML 4 Document Architecture

In This Chapter

What's new in HTML 4: new and obsoleted elements

Components: the building blocks of HTML

Block and inline elements: what they mean and when to use one or the other

Nesting: how it works

HTML: the element

HEAD: the element

What else you can put in the HEAD

BODY: the element

Making your HTML legible

This chapter is about building the basic infrastructure of your page. Later, you learn about making everything look nice, but here you'll lay the foundation to build upon.

What's New in HTML 4?

HTML 4 introduces a number of new elements. They fall into the categories of frame elements, forms elements, annotation elements, table elements, and CSS-facilitating elements. One stray new element doesn't conveniently fall into any category — the BDO element — which gives you the ability to change text direction for an inline element. If the standard text direction of your document is left-to-right and you want to override that to display one sentence of a language requiring right-to-left directionality, you would use the BDO element.

Frames

Even though you may have been using frames for two years or engaged in heated debates about whether frames are a good way to structure a site, frames have never been an official part of the HTML specification until now. The World Wide Web Consortium (W3C) has finally embraced frames or at least recognized frames by adding the FRAME, FRAMESET, NOFRAMES, and IFRAME elements to the HTML 4.0 specification.

Framesets are what you use to define the presence of frames on your page. The IFRAME element is truly new to HTML and is a fabulous way to introduce an element to your page that stays consistent across pages. Unfortunately, the IFRAME element is not yet supported in both the major browsers, but the IFRAME element may well replace the FRAME element, once it is well supported.

Cross-Reference

You can read more about the FRAMESET element, the FRAME element, and the IFRAME element in Chapter 22.

Note

The IFRAME element is supported in Internet Explorer 3 and higher, but not in Netscape 4.7.

Forms

A number of new elements have been introduced into the HTML specification to make forms more intuitive to the site visitor. The OPTGROUP element enables you to group your form fields in such a way that the visitor to your site should be able to navigate them more easily. Using style sheets, you can use one background color for the personal information form fields and another background color for the order form fields. The LABEL element gives you the opportunity to label each group of fields to process back on your server or to increase clarity to the visitor. The BUTTON element increases your flexibility when creating submit-type buttons at the bottom of your form.

Annotation

There are literary conventions that aren't currently supported in browsers, even though they may be useful. These include the capability to indicate what part of a text has been deleted and what part of a text has been inserted (for example, in drafting a law) using the DEL and INS elements. Other annotation-type elements are the ABBR (abbreviation) element, the ACRONYM element, and the Q (quote) element. None of these last three elements are currently displayed in any special way by browsers.

Cross-Reference

If you read more about these three elements, you can clearly see how they could come in handy. Check out Chapter 17 where they are discussed in detail.

Tables

You might have thought there wasn't anything you could imagine that would make tables any more attractive to you as a Web developer, but the new HTML 4 elements for tables have something for everyone. Even though the rest of HTML 4 moves formatting into style sheets, CSS1 doesn't support table formatting well, so the new elements are all about improving and simplifying table presentation. They include the COLGROUP, THEAD, TBODY, TFOOT, and LEGEND elements.

Facilitating CSS

The last group of new elements helps make CSS possible. These elements are the `OBJECT` element, which is how you incorporate images, sounds, movies, animations, and other types of multimedia into your page, and the `SPAN` element, which helps you define an area to which you want to apply a style from your style sheet.

What's gone from HTML 4.0?

Very little. As with any language, a few things fall by the wayside but many new terms become accepted. The only three obsoleted elements were associated with changing the way the text looked on the page to make it use fixed-width font. The way to do this in HTML 4 is probably the way you've been doing it all along — with the `PRE` element. The ways not to do it are with the `LISTING` element, the `PLAINTEXT` element, or the `XMP` element. Also, there are deprecated elements and attributes.

Definition

Obsoleted. An obsoleted element is one that won't necessarily work in future versions of browsers. You should find ways to remove it from your pages now.

Deprecated. A deprecated element is one that will be obsoleted in the future. You can use it today without any negative consequences, but you might have to go through all your pages in a year or two to pull it when it gets obsoleted.

Components of HTML

This is probably a good time to review the basic building blocks of HTML. You learned them in Chapter 1, but that was a long time ago. To review, they are elements, attributes, and entities.

Elements

Elements are how you tell the browser what part of the page a certain segment of text is in your page. The browser isn't smart enough to know that if you leave a blank line in front of a few sentences of text and leave another blank line after the text, that is probably a paragraph. You tell it that is a paragraph by using the `P` element. If, instead, this text was a blockquote, you'd tell it using the `BLOCKQUOTE` element — and you'd be pleased the browser hadn't guessed wrong.

Cross-Reference

All the elements are listed in Appendix B.

Attributes

Attributes are how you dress up and customize your elements. Just about every element has at least two attributes. These are the `id` attribute and the `class` attribute. There are hundreds of other attributes, but each element only takes a few. Many of these they share in common. You'll soon find you remember which attributes go with which elements. HTML editors often provide a list of attributes for an element as part of tag hints.

Cross-Reference

You'll learn everything you want to know about the `id` attribute and the `class` attribute in Chapter 26. All the attributes are listed in Appendix C.

Entities

Entities enable you to display special characters on your page. Without entities, you can only display the characters you can type, minus the characters that have special meaning to HTML, such as less than (<), greater than (>), and ampersand (&). With entities, you can display the copyright symbol (©), the trademark symbol (™), the cent sign (¢), the symbols for other currencies (¥ yen, £ pound sterling), and hundreds of other characters, including letters and symbols from other alphabets.

Cross-Reference

All the entities are listed in Appendix D.

Block Versus Inline Elements

Elements fall into two categories: block and inline. *Block elements* can contain other block elements and can also contain inline elements. *Inline elements* can only contain other inline elements. Block elements have more to do with the structure of your page. Inline elements are more about emphasis. Consider Table 12-1.

Table 12-1
Comparison of Block and Inline Elements

Block elements	***Inline elements***
structure your document	add emphasis
define parts of your document	set a part of a block apart from the rest of the block
permit formatting	
take IDs and classes for purposes of style sheet formatting	
can contain other block elements	
can contain inline elements	

Examples of block elements are: P, HTML, BODY, and BLOCKQUOTE. Examples of inline elements are: B (bold), I (italics), Q (quote), and A (anchor).

Understanding Nesting

To make HTML work, you must learn to put block elements within other block elements and to put inline elements into other inline elements and into block elements. This process is called *nesting*. Consider the following nested list:

- This is the first bulleted item
- This is the second bulleted item
  - This is a numbered item
  - This is another numbered item
  - This is a third numbered item
- This is the third and final bulleted item

The following HTML will create the same thing:

```
<UL>
  <LI> This is the first bulleted item
  <LI> This is the second bulleted item
  <OL>
       <LI> This is a numbered item
       <LI> This is another numbered item
       <LI> This is a third numbered item
  </OL>
  <LI> This is the third and final bulleted item
</UL>
```

First, you should know — and you may already know — that the rules about nesting are not enforced by all browsers. What this means is if you break them, there might not be any visible penalty *today*. In the future, as more elements are introduced, nesting rules will have to be more strictly enforced. Now the rules:

1. **Close your elements in the reverse of the order you opened them.** If you have a paragraph in which you identify a book title, which you both bold and italicize, you must close the inner element before you close the outer element.

   In the following example, notice the B element is the last one opened and, therefore, is the first one closed. The P element, which is the only block-level element, is the first one opened; therefore, it is the last one closed.

   ```
   <P>One book I can highly recommend is Dostoyevsky's
   <I><B>Crime and Punishment</B></I></P>.
   ```

2. **Always close inline elements before you close the block element that contains them.** This is a subset of the previous rule. But the consequences of breaking it can be visible on your page. If you fail to close an inline element when you intend to, you could end up with bolded or italicized and bolded text on the rest of your page.
3. **Sometimes opening a new block automatically closes the previous open block.** You need to play this by ear, but there are times when beginning a new block element automatically ends the previous block element. One time this absolutely works is with paragraphs. You simply can't put a `P` element within another `P` element. Go ahead and try it. When your browser sees the second `P` element, it automatically closes the first one. However, you certainly can put a list within another list. Starting a second `UL` element doesn't close the previous `UL` element. You can also put a `P` element into a `UL` element, which doesn't close it either.

The HTML Element

The `HTML` element is the mother of all block-level elements. It contains the `HEAD` and the `BODY`. That said, it is optional, according to the HTML 4 specification, but older browsers will expect to see it. Your browser is smart enough to know—when it hits a `HEAD` element—that it is squarely in the middle of an `HTML` element. Putting it in is considered good form but no calamities will befall you if you leave it out (at least not any calamities related to the `HTML` element). The end tag is also optional.

HTML `<HTML>`

Start Tag:	Optional
Content:	`HEAD`
	`BODY`
End Tag:	Optional
Attributes:	`Lang`: language
	`Dir`: text direction
	`Version`: version (deprecated: should go on its own line)

```
<HTML>
<HEAD>
  <TITLE>This is the title</TITLE>
</HEAD>
<BODY>
           The body goes here.
</BODY>
</HTML>
```

The HEAD Element

Generally speaking, you'll want to include a HEAD element at the top of your document.

Head <HEAD>

Start Tag:	Optional
Content:	TITLE: title
	BASE: base directory for relative references
	META: meta elements
	SCRIPT: script
	STYLE: style element for internal style sheets
	LINK: link to external style sheets
	OBJECT: object
End Tag:	Optional
Attributes:	Lang: language
	Dir: text direction
	Profile: URL of metadata

```
<HEAD>
     <TITLE>A really spiffy page</TITLE>
     <META name="creator" content="John Q. Public">
</HEAD>
```

Additional HEAD Elements

What else can you put into the HEAD? A TITLE element, a BASE element, a SCRIPT element, a STYLE element, a LINK element, and an OBJECT element. You remember the TITLE element from Chapter 3. You should have one, and only one, of those. You use the TITLE element to title your document; it also shows up in search engines. The TITLE element shows up in the top of the browser, above the menus.

Title <TITLE>

Start Tag:	Required
Content:	Document title
End Tag:	Required

Attributes: `lang`: language code (see Appendix F for a complete list), if different from the language code specified in the

`BODY` element

`dir`: text direction, if different from that specified in the

`BODY` element

The `BASE` element is a clever device you can make use of whether or not you are using a frameset. If you are using a frameset, you can use the `BASE` element to tell all references contained in your page to open in the same frame in which the current page is open. (This will make more sense in Chapter 22.) If you are not using frames, the `BASE` element is a way to say: "start looking for all relative references from this point." This can be convenient if all your relative references (in images and in links) start from two directories below the current page. You can avoid having all your URLs begin with "../../". It also makes things easier if you have to move a page later. You can make the `BASE` element the directory where the page used to reside so you needn't recode all the links.

Base `<BASE>`

Start Tag: Required

Content: Forbidden

End Tag: Forbidden

Attributes: `target`: frame name where you want all links in this page to open

`href`: URL to use as base point for all relative references

If you want to have a script that is event-driven in your page, then you should use the `SCRIPT` element in your `HEAD` to include it. An *event-driven script* is one that only gets called when a certain event occurs. Events can be created by visitor actions, by the clock, or by other things you set up. The results of actions taken by the visitor — clicking a check box, for example — are the most common type of events.

Cross-Reference

You'll learn about JavaScript and writing scripts in Chapter 48.

Script `<SCRIPT>`

Start Tag: Required

Content: The script

End Tag: Required

Attributes: `type`: Scripting language; content type

If you want to include a style sheet within your page, which is *not* the recommended way, you can use a STYLE element to do this. The recommended way is to define the style sheet externally, in a text document with a .css extension, and link to it using the LINK element. The style sheet definition will look exactly the same, whether you define it in a STYLE element or in an external file linked to your document by the LINK element.

Style <STYLE>

Start Tag:	Required
Content:	Style sheet definition
End Tag:	Required
Attributes:	type: content type, just as with the SCRIPT element
	media: screen, print, and so forth (not actually implemented yet, but a good idea)

Link <LINK>

Start Tag:	Required
Content:	Forbidden
End Tag:	Forbidden
Attributes:	id, class, lang, dir, title, style: as defined elsewhere in this chapter
	events: see Chapter 48
	href: URL for linked resource, such as external style sheet
	hreflang: language code for linked resource
	type: content type of href
	rel: forward link types
	rev: reverse link types
	target: target frame information
	media: for rendering on media
	charset: character encoding of linked media

```
<LINK href="styles/css1.css" type="text/css">
```

The BODY Element

The BODY element is the block-level element that holds nearly everything that shows up in your browser window. It can contain all the other block elements, except for HEAD and HTML.

Body <BODY>

Start Tag:	Optional
Content:	Document title
End Tag:	Optional
Attributes:	onload, onunload: event, see Chapter 48
	background: URL for background image (deprecated)
	text: foreground color for text (deprecated)
	link: foreground color for link (deprecated)
	vlink: foreground color for visited link (deprecated)
	alink: foreground color for active link (deprecated)
	id, class: for style sheets
	lang, dir: language and direction, as previously explained
	title: title
	style: style information
	bgcolor: background color (deprecated)

Structure Versus Presentation, Again

You will notice this book often lists all those nasty legacy attributes from HTML 3.2 that affect presentation right in your HTML code. Again, you should try to avoid using them as much as possible. Even though you may now be convinced you can live without style sheets and that CSS is something too complicated for your simple site, when the time comes to maintain your pages and you have to wrestle with all those nasty presentation attributes, you will regret it.

Cross-Reference

The HTML 4 Way is to separate presentation from structure. If you are dying to get to the interior decorating and you are willing to wait to learn how to build the structure, jump to Chapter 25 and get it out of your system now. If you are willing to deny yourself the pleasure of designing for the nonce and to stick with learning and applying the fundamentals, you will be rewarded later.

Making Your HTML as Readable as Possible

When you create your HTML, you can make sense of it most easily if you use a few typographical conventions. Consider implementing the following guidelines in your files:

- ✦ Leave a blank line between block-level elements.
- ✦ Indent subordinate block-level elements.
- ✦ Indent any nested elements.
- ✦ Leave inline elements on the same line as the block in which they are contained.
- ✦ Format paragraphs as paragraphs (except the indent on the first line, which won't show up anyway).
- ✦ Format lists as lists.
- ✦ Keep each cell on a table in its own line.

This book scrupulously follows these guidelines because they make writing easy and maintaining easier. Most HTML editors also implement some sort of formatting of the HTML to make it easier to read.

Cross-Reference

CSS has its own rules about readability. Read all about them in Chapter 26.

From Here

Cross-Reference

Jump to Chapter 25 and get acquainted with style sheets.

Jump to Chapter 15 and learn about structuring text with block-level elements and lists.

Summary

This chapter gave you a thorough explanation about the nitty-gritty of page structure. If you've made it this far, you won't have any problems with HTML. Much of what this chapter covered you needn't know for your first page, but if you made it through Chapter 12 of a book this size, you're obviously serious about your HTML.

You know what's new in HTML 4 and what's gone altogether. You learned the difference between a block element and an inline element. You know about nesting.

You learned about the HTML element, the HEAD element — with the somewhat confusing META element and all its variations — and the other elements that can go in the head. You also learned the basics of the BODY element. Finally, you learned how to write your HTML so it will be readable and instantly make sense to you when you come back to it in six months.

✦ ✦ ✦