

Chapter 6

Creating a Simple Page (HTML Overview)

In [Part I: Getting Started](#), I provided a general overview of the web design environment. Now that we've covered the big concepts, it's time to roll up our sleeves and get started on the specifics of creating a real web page. It will be a simple page, but even the most complicated pages are based on the principles described in the following example.

In this chapter, I create a simple web page step by step. The important lessons here are:

- How HTML tagging works
- How an HTML document is structured
- How browsers display tagged documents

Don't worry about learning specific text-formatting tags at this point. All the tags will be discussed in detail in the following chapters. For now, just pay attention to the process and the overall structure of the document. Once you understand the basics, adding tags to your bag of tricks is simple.

IN THIS CHAPTER

An introduction to HTML tags and attributes

A step-by-step demonstration of putting together a simple web page

The basic structural tags

An overview of formatting text and adding images and links

Saving pages and viewing them in a browser

Troubleshooting broken web pages

HTML the Hard Way

With all the wonderful web-authoring tools out there today, chances are you will be using one to create your pages. In fact, I recommend it; the time and sanity savings are too good to pass up.

You may be asking, “If the tools are so great, do I need to learn HTML at all?” The answer is, you do. You may not need to have every tag memorized, but some familiarity is crucial for everyone who wants to make web pages. If you go looking for a job as a “web designer,” it will probably be assumed that you know your way around an HTML document.

I stand by my method of teaching HTML the old-fashioned way—*by hand!* There’s no way to truly understand how HTML works other than typing it out, one tag at a time, then opening your page in a browser. It doesn’t take long to develop a feel for tagging documents properly.

Understanding HTML will make using your authoring tools easier and more efficient. In addition, you will be glad to be able to look at an HTML source file and understand what you’re seeing. Say you see a really cool web page trick. You can always view the source to see how it’s done, but if the source looks like a bunch of gibberish to you, it won’t do much good.

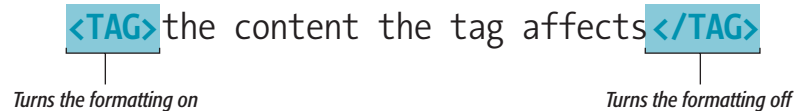
Once you know the basics, you can continue your learning by reading *Web Design in a Nutshell* (O’Reilly, 1999) or *HTML & XHTML: The Definitive Guide, Fourth Edition* by Chuck Musciano and Bill Kennedy (O’Reilly, 2000).

Introducing... the HTML Tag

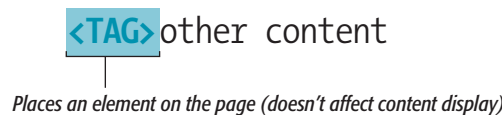
If you’ve read through Part I of this book, you know web pages are formatted using HTML tags. The characters within the tag are usually an abbreviation of a formatting instruction or an element to add to the page (Figure 6-1).

Figure 6-1

Container Tag Structure:



Standalone Tag Structure:



Most HTML tags are **container** tags. They consist of two tags (a beginning and an end tag) that are wrapped around a range of text. The tag instruction applies to all the content contained within the tags. Think of them as an “on” switch and “off” switch. The end tag looks the same as the start tag, only it begins with a slash (`/`).

A few tags are **standalone** tags: you just drop them into place where you want an element to appear. They do not have a closing tag.

We’ll be using both types of tags in the following web page demo.

Assembling a Web Page

We are ready to make a web page. This demonstration has four steps:

Step 1: Setting up the HTML document. You'll learn about the tags used to give an HTML document its structure.

Step 2: Formatting text. We'll use container tags to format the text on the page.

Step 3: Adding graphical elements. We'll use standalone tags to add pictures and rules to the page. We'll also look at how tag attributes work.

Step 4: Adding a hypertext link. Since the Web is about linking, a web page demo would be incomplete without an introduction to linking.

I'll be typing the HTML by hand using an HTML editor called BBEdit (on a Mac). You could also use Allaire HomeSite if you're on a PC. Word processing programs such as Microsoft Word are not appropriate because they add hidden information to the code, and what we're after is pure text characters (ASCII).

I'll be checking my work in a browser frequently throughout this demonstration—probably more than you would in real life—but since this is a first introduction to HTML, I find it helpful to show the cause and effect of each change.

The end result will be the home page for a site called “Jen’s Kitchen” (Figure 6-2) that links to a number of my favorite recipes.

Figure 6-2



Free Software Samples

You can try out these HTML editors for free!

For BBEdit, go to the Bare Bones Software site at www.barebones.com/free/free.html and download a free demo.

For Allaire HomeSite, start on the product page at www.allaire.com/products/homesite/index.cfm and choose “Download HomeSite 4.5.” You’ll need to fill out customer information, but when that’s through, there is a free evaluation version of HomeSite available on the list of downloads.

In this chapter, we'll assemble this web page step-by-step. It's not very fancy, but we have to start somewhere.

To Capitalize or Not to Capitalize

Throughout this book, I have written my tags in all capital letters, but they could also be lowercase. In other words, tags are not “case-sensitive.” The choice is yours, but here are some things you might take into consideration:

On the one hand, using all capital letters makes the tags stand out against a sea of code. This is helpful when you are writing your HTML code from scratch.

On the other hand, as HTML evolves, related tagging systems (such as XML and XHTML) require lowercase tags only. If you are learning HTML for the first time and you anticipate becoming a web professional, you might want to get in the habit of writing all lowercase tags from the start.

Step 1: Setting Up the HTML Document

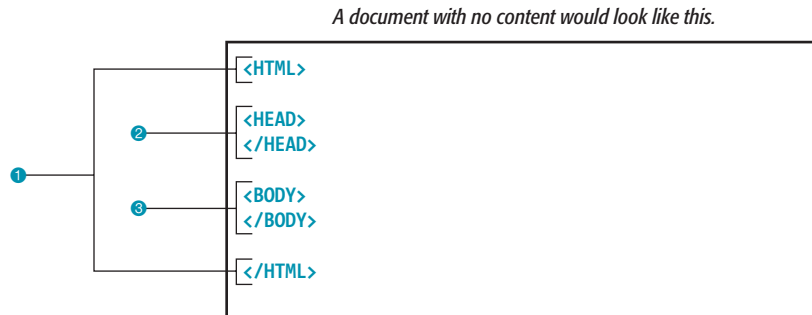
There are two things that make an ordinary text file a browser-readable web document. The document must have a name that ends in `.htm` or `.html` in order to be recognized by the browser, and it must contain the basic HTML tags that define the structure of the web document.

Basic structure

Begin a new web document by giving it a skeleton.

There are really only two parts to an HTML document: the `head` (also called the `header`) and the `body`. The head contains information about the document (its title, for instance); the body contains the actual content of the document. The structure of the document is identified by using the `<HTML>`, `<HEAD>`, and `<BODY>` container tags (Figure 6-3).

Figure 6-3

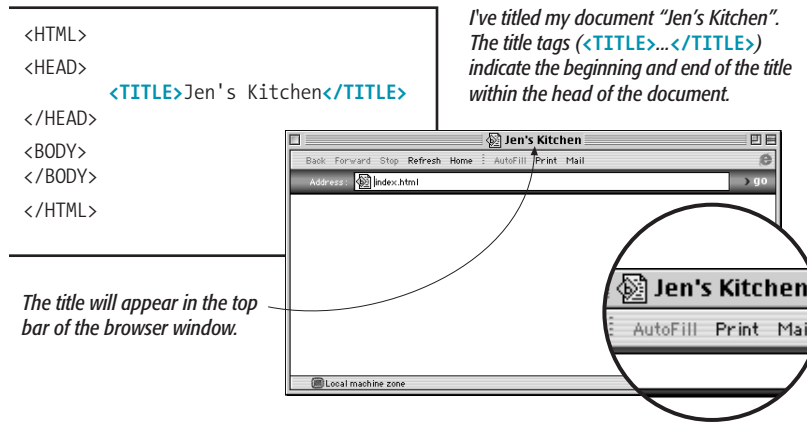


- ❶ First, tell the browser that the text is in HTML format by labeling the entire document as “HTML.” Place the “start HTML” tag (`<HTML>`) at the very beginning of the text and the “end HTML” tag (`</HTML>`) at the end.
- ❷ The `<HEAD>...</HEAD>` tags define the beginning and end of the head section of the document. Right now, that section is empty.
- ❸ The `<BODY>...</BODY>` tags define the body of the document. This is where we’ll put the contents of the page; that is, everything we want to display in the browser window.

Giving the page a title

Another essential part of the document is its title. This is the name you give to the page; it is shown in the top bar of the browser window. If you don't give the document a title, the filename will be used instead. The title, indicated by the `<TITLE>` container tag, is placed within the head of the document (Figure 6-4).

Figure 6-4



The Importance of the Title

The title is one of the most important pieces of information you provide about your web page. In addition to appearing at the top of the browser when the page is open, it will be listed in the Bookmarks (or Favorites) menu when someone bookmarks your page. The title is also the first thing search engines look at when indexing your page. Make sure it is descriptive and useful. How many bookmarks would you want named "Welcome!"?

Adding content

So far, so good, but if we want something to appear in the browser window, we need to put some content in the body of the document. I've typed up a simple introduction and list (Figure 6-5).

Figure 6-5



TOOL TIP

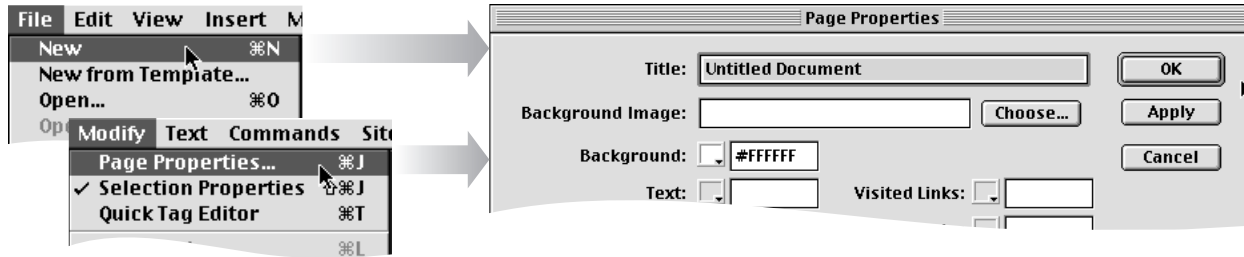
Document Creation

When you use a web-authoring tool such as Macromedia Dreamweaver or Adobe GoLive, the structural tags are added automatically when you create a new document. The tools usually add some extra document information to the

header as well (such as `<META>` tags that say what software was used to create the file). In most tools, the title is specified on the page that has settings for the whole document.

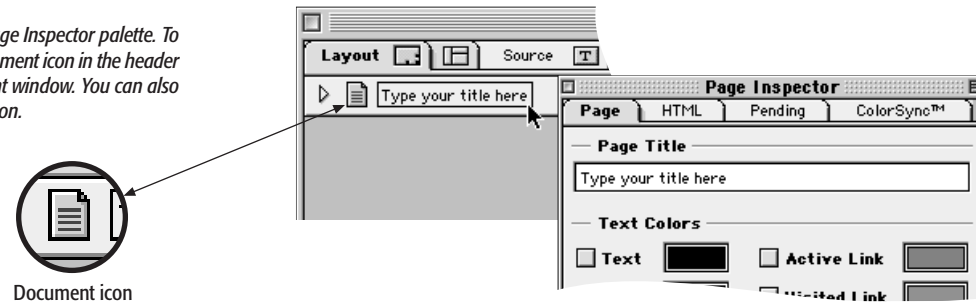
DREAMWEAVER 3

The document title is entered in the Page Properties dialog box.



GOLIVE 4

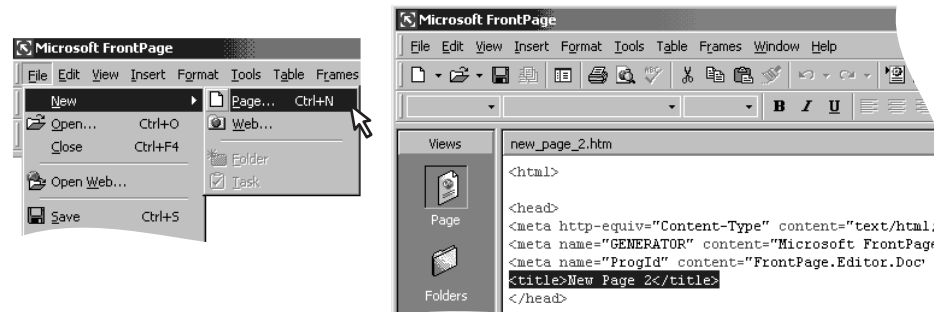
Enter the document title in the Page Inspector palette. To open the palette, click on the document icon in the header section at the top of the document window. You can also just type in the title next to the icon.



Document icon

FRONTPAGE 2000

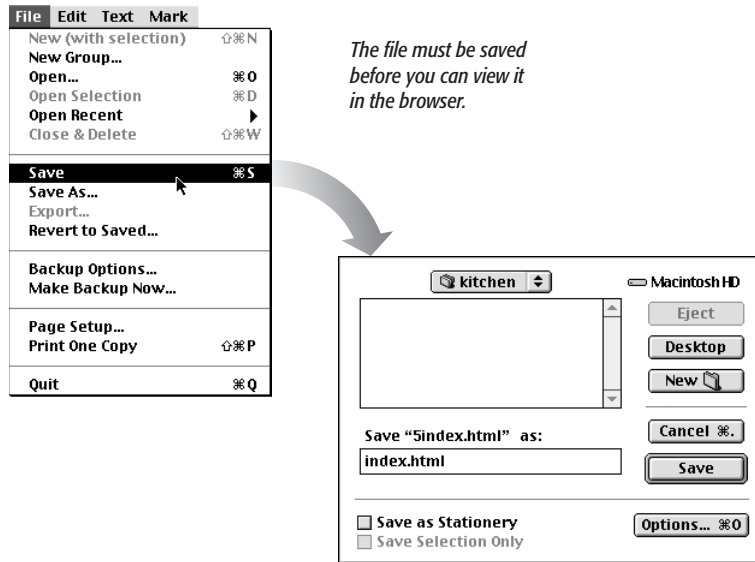
Open a new page and default title will be highlighted in code. Type your title between `<TITLE>` and `</TITLE>` to replace.



Saving and viewing the page

We've got a document with the proper HTML structure and some content, but in order to view it in the browser, we need to save the file and give it a name (Figure 6-6). The filename needs to end in `.htm` or `.html` in order to be recognized by the browser as a web document. See the sidebar [Naming Conventions](#) for more tips on naming files. I've named my file `index.html`.

Figure 6-6



Now we can view `index.html` in a browser. Opening a file from your computer's hard drive is called opening a file "locally." You don't need an Internet connection to check your work on a browser. Just launch your browser and choose "Open Page" or "Open Local" (or similar wording) from the File menu and locate your file in the dialog boxes (Figure 6-7, following page).

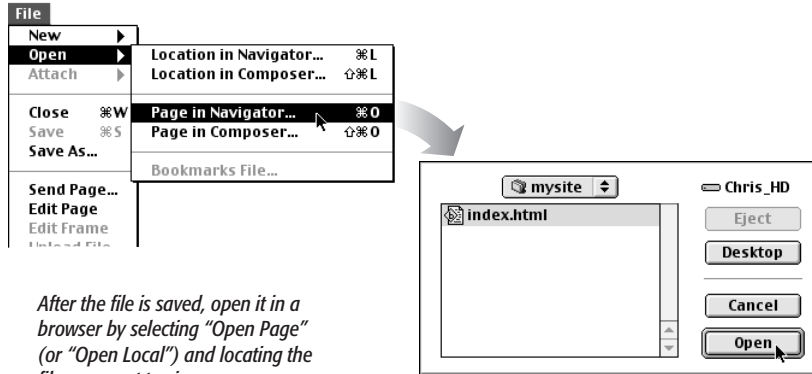
Naming Conventions

Follow these rules and conventions when naming your files:

- Use proper suffixes for your files. HTML files must end with `.html` or `.htm`. Web graphics must be labeled according to their file format: `.gif` or `.jpg` (`.jpeg` is also acceptable).
- Never use character spaces within filenames. It is common to use an underline character to visually separate words within filenames, such as `lynch_bio.html`.
- Avoid special characters such as `?`, `%`, `#`, `/`, `:`, `;`, `'`, etc. Limit filenames to letters, numbers, underscores, hyphens, and periods.
- Filenames are case-sensitive in HTML. Consistently using all lowercase letters in filenames, while not necessary, makes your filenames easier to manage.
- Keep filenames short.
- If you really must give the file a long, multiword name, you can separate words with capital letters, such as `ALongDocumentTitle.htm`, or with underscores, such as `a_long_document_title.htm`, to improve readability.

Assembling a Web Page

Figure 6-7



After the file is saved, open it in a browser by selecting “Open Page” (or “Open Local”) and locating the file you want to view.

Without text-formatting HTML tags, the content of my page gets run together when viewed in a browser.

index.html

```
<HTML>
<HEAD>
  <TITLE>Jen's Kitchen</TITLE>
</HEAD>
<BODY>
  Jen's Kitchen
  People who know me know that I love to cook. I've created
  this site to share some of my favorite recipes and online food
  resources. Bon Appetit!
  From Jen's Cookbook:
    tapenade (olive spread)
    garlic salmon
    wild mushroom risotto
    asian dishes
</BODY>
```



The only way a browser will make a new paragraph or add a space is if it sees a tag in the file that specifically tells it to do so. Otherwise, it will just ignore returns, tabs, and consecutive spaces.

Whoa! The text is all run together—not what I had in mind! My page title is up in the top bar of the browser, so that’s a start. But let’s look at what else happened. Notice how the browser ignored all of my line breaks. It also ignored the extra spaces I had added to indent the recipe names.

A browser will make a new paragraph or add a space only if it sees a tag in the file that specifically tells it to do so. Otherwise, it just ignores returns, tabs, and consecutive spaces in the text file.

This feature comes in handy, in a way, since you can enter as many returns and indents in your HTML document as you like. This makes it more readable while it’s in the editor and won’t affect your final product. The sidebar [What Browsers Ignore](#) provides some useful insights on how browsers interpret HTML code.

What Browsers Ignore

Some information in an HTML document will be ignored when it is viewed in a browser, including:

Line breaks (carriage returns). Line breaks are ignored. Text and elements will wrap continuously until a paragraph (`<P>`) or line break (`
`) tag is encountered in the flow of the document text.

Tabs and multiple spaces. When a browser encounters a tab or more than one consecutive blank character space, it will display a single space. So if the document contains:

```
long,           long           ago
```

the browser will display:

```
long, long ago
```

Extra spaces can be added by using the “nonbreaking space” character string (` `) for each desired character space. (See the section [Some Special Characters](#) at the end of [Chapter 7, Formatting Text](#).)

Multiple `<P>` tags. When a browser sees a `<P>` (paragraph) tag, it will add a line space; however, a series of `<P>` tags (or paragraph containers, `<P>...</P>`) with no intervening text is interpreted as redundant and will display as though it were only a single `<P>` tag. Most browsers will display multiple `
` tags as multiple line breaks.

Unrecognized tags. A browser simply ignores any tag it doesn’t understand or that was incorrectly specified. Depending on the tag and the browser, this can have varied results. Either the browser displays nothing at all, or it may display the contents of the tag as though it were normal text.

Text in comments. Browsers will not display text between the special `<!--` and `-->` elements used to denote a comment. Here is a sample comment:

```
<!-- This is a comment -->
<!-- This is a
multiple-line comment
that ends here. -->
```

There must be a space after the initial `<!--` and preceding the final `-->`, but you can put nearly anything inside the comment otherwise.

Step 2: Formatting Text

Let’s quickly put some text-formatting tags in there to whip that text into shape ([Figure 6-8](#), following page). At this point, don’t worry too much about the specific tags—I just want you to get acquainted with the tagging process. We’ll be discussing text-formatting tags in detail in [Chapter 7](#).

At this point, don’t worry too much about the specific tags—I just want you to get acquainted with the tagging process.

A Brief History of HTML

Before HTML there was SGML (Standard Generalized Markup Language), which established the system of describing documents in terms of their *structure*, independent of appearance. SGML tags work the same way as the HTML tags we've seen, but there can be far more of them, enabling a more sophisticated description of document elements. Publishers began storing SGML versions of their documents so that they could be translated into a variety of end uses. For example, text that is tagged as a heading may be formatted one way if the end product is a printed book, but another way for a CD-ROM. The advantage is that a single source file can be used to create a variety of end products. The way it is interpreted and displayed (i.e., the way it *looks*) depends on the end use.

Because HTML is one application of an SGML tagging system, this principle of keeping style information (instructions for how elements look) separate from the structure of the document remains inherent to the HTML purpose. Over the past few years, this ideal has been muddled somewhat by the creation of HTML tags that contain explicit style instructions, such as the `` tag, which gives designers control over the font, size, and color of text it contains, and the use of table tags for page layout.

A new system called Cascading Style Sheets has been introduced that promises to keep style information out of the content by storing all style instructions in a separate document (or separate section of the source document).

(continued on next page...)

Figure 6-8

I've added text-formatting tags to the HTML file to create headings (`<H1>`, `<H2>`), paragraphs (`<P>`), line breaks (`
`), and italic text (`<I>`).

index.html

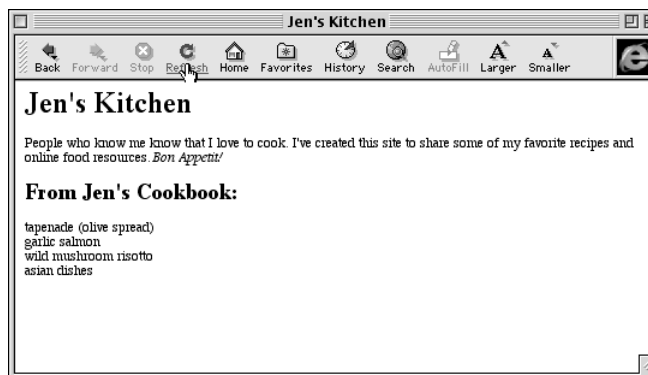
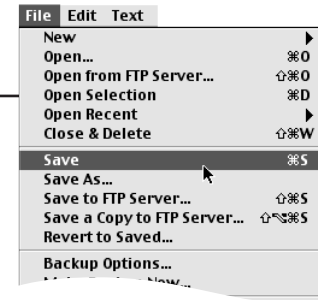
```

<HTML>
<HEAD>
  <TITLE>Jen's Kitchen</TITLE>
</HEAD>
<BODY>
  ① <H1>Jen's Kitchen</H1>
  ② <P>People who know me know that I love to cook. I've created
    this site to share some of my favorite recipes and online food
    resources. <I>Bon Appetit!</I></P>
  ③ <H2>From Jen's Cookbook:</H2>
  ④ <P>
    tapenade (olive spread)<BR>
    garlic salmon<BR>
    wild mushroom risotto<BR>
    asian dishes
  </P>
</BODY>

```

Remember that you need to save your document in order for the changes to show up in the browser.

Because my page is already open in the browser, this time I can just hit "Refresh" (or "Reload") to see my new page.



1 `<H1>...</H1>`

I've placed start and end first-level heading tags around the headline of my page. The browser renders the text between `<H1>` tags in the largest bold text available. After the introduction, I've created a second-level heading (`<H2>`). You can see it is slightly smaller than the `<H1>` text. Notice also that these tags cause line breaks and extra space to be added above and below the headings.

2 `<P>...</P>`

Paragraphs are indicated by wrapping the text in paragraph container tags (`<P>...</P>`). When you define a paragraph, the line automatically breaks and some space is added above and below.

It is possible to separate paragraphs by placing a single `<P>` tag between them. Browsers will render it the same as `<P>...</P>`. However, it is proper and preferable to use container tags, and furthermore, containers are required if you want to add formatting with style sheets. So you might as well start off on the right foot.

3 `<I>...</I>`

For emphasis, I've tagged the words "Bon Appetit!" with tags that start italic formatting (`<I>`), and then turn it off (`</I>`).

4 `
`

To cause a line break without extra space (for instance, between recipe names), I added break tags (`
`) at the points I want breaks to occur. The line break tag is an example of a standalone tag. It doesn't work on a range of text: it just inserts the break.

Again, I've saved my file and checked my progress in the browser window. This time, I was able to just hit the "Reload" or "Refresh" button since I already had the last version of the page in my browser window. Remember, your changes won't be visible on reload unless you save your file first.

Step 3: Adding Graphical Elements

By now, I'm sure you're getting the hang of container tags. Let's drop in some graphical elements to give the page more pizzazz. Again, don't worry too much about the specific tags at this point; they'll be covered in [Chapter 8, Adding Graphic Elements](#).

I'll add a title graphic to the top of the page and a horizontal rule (a line) to break up the page ([Figure 6-9](#), following page). These elements will give us a good opportunity to look at how standalone tags and attributes work.

A Brief History of HTML

(continued from previous page)

Style sheets are an advanced technique and therefore are not covered in depth in this book (see [Chapter 20, How'd They Do That?](#) for an introduction). Let's consider an example, though. Within the document, a heading will be labeled with a standard `<H1>` to indicate the type of information; elsewhere, in the style sheet, the designer specifies, "I'd like H1s to be 36 point, blue Helvetica type centered on the page." This pleases both the designers and the HTML purists. As Martha Stewart says, it's a "good thing."

Remember, your changes won't be visible on reload unless you save your file first.

Figure 6-9

index.html

```

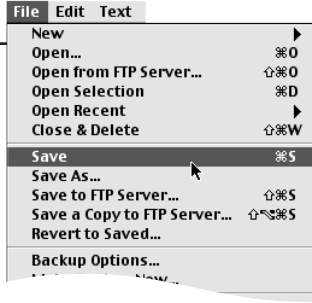
<HTML>
<HEAD>
  <TITLE>Jen's Kitchen</TITLE>
</HEAD>
<BODY>
  <IMG SRC="kitchen.gif">
  <P>People who know me know that I love to cook. I've created
  this site to share some of my favorite recipes and online food
  resources. <I>Bon Appetit!</I></P>
  <HR WIDTH="50%" SIZE="6">
  <H2>From Jen's Cookbook:</H2>
  tapenade (olive spread)<BR>
  garlic salmon<BR>
  wild mushroom risotto<BR>
  asian dishes
</BODY>
  
```

1
Inserts an image.

2
Inserts a horizontal rule (line). I've added attributes to change the length and thickness of the rule.

3
Hit "Refresh" to view the changes. The page is almost done!

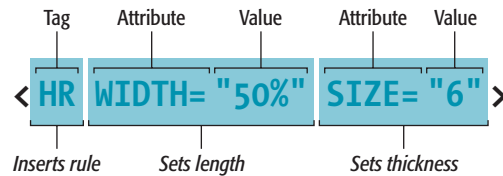
Gotta save it!




- 1 I've replaced my text heading with a much spiffier graphic heading. The graphic is added to the page by placing an `` tag where I want the graphic to appear. The image tag is a good example of a **stand-alone tag**—there is no closing or end tag, you just plop it into place.
- 2 I've also added a horizontal rule (line) using the `<HR>` standalone tag. I've added some attributes within the tag to change the length (`WIDTH=`) and thickness (`SIZE=`) of the rule.

An **attribute** is a bit of information that is added to a tag to modify its action or behavior. Attributes are separated from their value settings by an equals sign (Figure 6-10). The sidebar **About Attributes** has more useful facts about this important feature of HTML.

Figure 6-10



The `SRC=` part of the `` tag is an example of a required attribute—without it, the browser wouldn't know which graphic to grab. The attributes in the `<HR>` tag are optional. Without the attributes, the default horizontal rule would be one pixel thick and the width of the browser window.

- 3 I've saved and reloaded the page to check the progress so far. It's almost done! Just one more thing...

Step 4: Adding a Hypertext Link

What's a web page without links? Fairly pointless, if you ask me, so let's add one. Don't worry about learning everything about linking from this example. Linking is an important part of web design and I've dedicated a whole chapter to it ([Chapter 9, Adding Links](#)). For now, I just want to give you a flavor of how it's done.

In the end, I'd like each of my recipe names to link to their respective recipe pages, so I'll start with the first one in this example (Figure 6-11, following page).

About Attributes

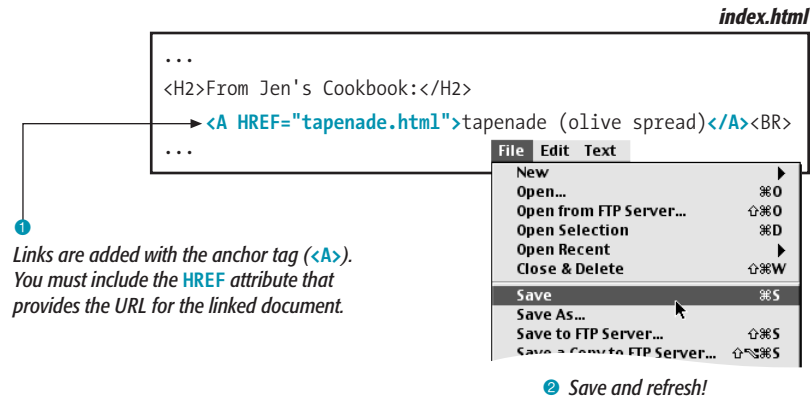
The real power and flexibility of HTML lies in the attributes—small instructions added within a tag to modify its behavior or appearance. The formula for using attributes is as follows:

```
<TAG ATTRIBUTE="value">
affected text</TAG>
```

Here are some important things to know about attributes:

- Attributes go only in the opening container tag. The closing tag includes just the tag name, even if the opening tag is loaded with attributes.
- Most (but not all) attributes take **values**, which follow an equals sign (=) after the attribute's name. The value might be a number, a word, a string of text, a URL, or a measurement.
- You can add several attributes within a single tag.
- It is good practice to put quotation marks around values; however, they may be omitted if the value is a single word or number.
- Some attributes are required; for example, the `SRC` attribute within the `` tag.

Figure 6-11



And there we have it: a simple web page, complete with a graphic and a link. Not very fancy, but a web page nonetheless!



- 1 Links are added with a container tag called an **anchor** (<A>...). Like other container tags, anchor tags are placed around the text that you want to link. But you have to specify what page you want to link to, right? That's where the **HREF=** attribute comes in. It is a required attribute that gives the browser the URL of the target page. In my example, I've used a relative URL (a URL that points to a document on the same server) to create a link to the tapenade recipe page (*tapenade.html*).
- 2 When I save my file and reload it in the browser, the anchor text will appear as blue, underlined, clickable text.

The page is done! At this point, I could upload it to the server. See [Chapter 3, Getting Your Pages on the Web](#), for step-by-step instructions on uploading.

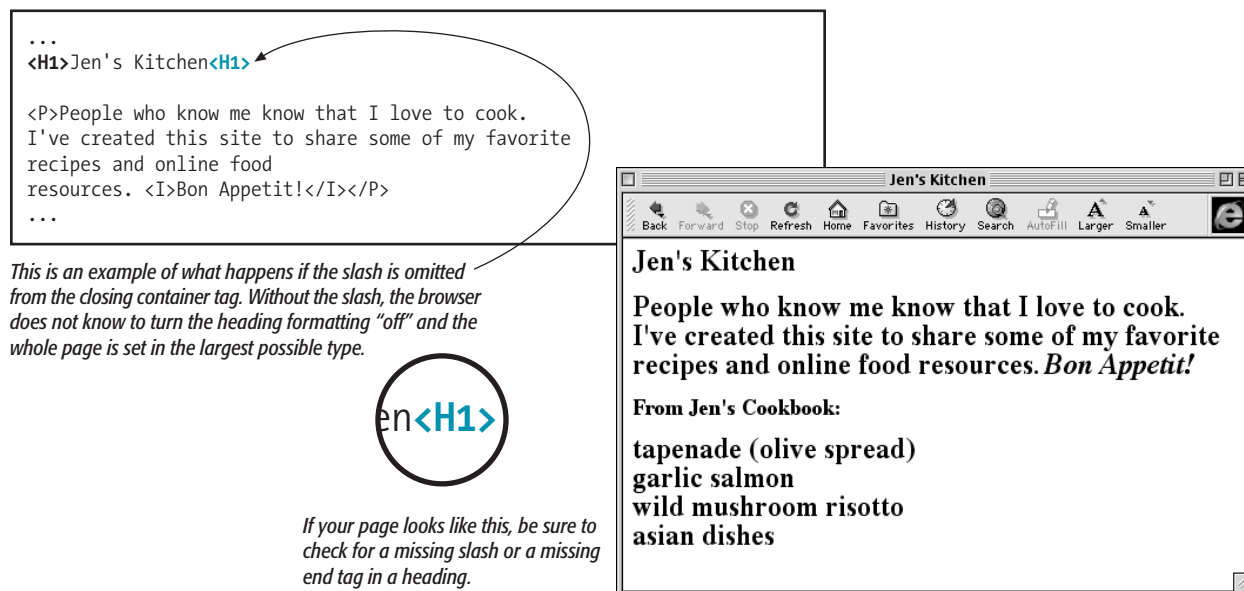
I know, I know... you're thinking, "That page is really boring." That's okay, I'm thinking the same thing. But it is a real web page nonetheless and we picked up some key concepts on the way. In future chapters, we'll add to your bag of tricks so you can make pages that are a bit more exciting.

When Good Pages Go Bad

The previous demonstration went very smoothly, but it's easy for small things to go wrong when typing out HTML code. Unfortunately, one missed character can break a whole page. I'm going to break my page on purpose so we can see what happens.

What if I had forgotten to type the slash (/) in the closing header tag (</H1>)? Just one character out of place (Figure 6-12). As you can see, the entire document displays in big bold heading text! That's because without that slash, there's nothing telling the browser to turn "off" the heading formatting, so it just keeps going.

Figure 6-12



When Good Pages Go Bad

Having Problems?

The following are some typical problems that crop up when creating web pages and viewing them in a browser:

Q: I've changed my document, but when I reload the page in my browser, it looks exactly the same.

A: It could be you didn't save your HTML document before reloading. It's an important step.

Q: All the text on my page is HUGE!

A: Did you start a heading tag and forget to close it? Make sure each tag you've used has its end tag. Also, make sure that end tag has a slash (/)!

Q: Half my page disappeared!

A: This could happen if you are missing a closing bracket (>) or a quotation mark within a tag. This is a common error when writing HTML code by hand.

Q: I put in a graphic using the tag, but all that shows up is a broken-graphic icon.

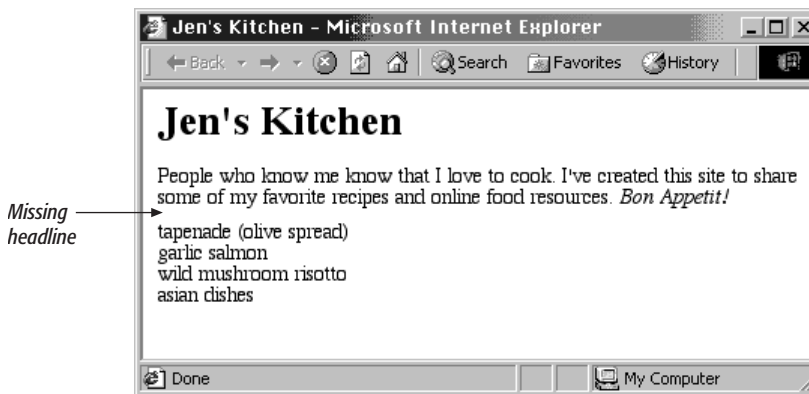
A: The broken graphic could mean a couple of things. First, it might mean that the browser is not finding the graphic. Make sure that the URL to the graphic is correct. (We'll discuss URLs further in Chapter 8, Adding Links.) Make sure that the graphic is actually in the directory you've specified. If the graphic is there, make sure it is in one of the formats that web browsers can display (GIF or JPEG) and that it is named with the proper suffix (.gif and .jpeg or .jpg, respectively).

I've fixed the slash, but this time, let's see what would have happened if I had accidentally omitted a bracket from the end of the first <H2> tag (Figure 6-13).

Figure 6-13

```
...
<H2From Jen's Cookbook:</H2>
<P>
  tapenade (olive spread)<BR>
  garlic salmon<BR>
  wild mushroom risotto<BR>
  asian dishes
</P>
...
```

In this example, I've accidentally left out a bracket. Notice how "From Jen's Cookbook" disappears in the browser. Without that bracket, it's reading all the following characters as some elaborate tag it's never seen before.



If you find text vanishing when you view your page in the browser, check for a missing bracket or missing quotation mark. They are usually to blame.

See how a whole chunk of text is missing? That's because without the closing tag bracket, the browser assumes that all the following text—all the way up to the next closing bracket (>) it finds—is part of that <H2> tag. Browsers don't display any text within a tag, so my heading disappeared. The browser just ignored the foreign-looking tag and moved on to the next element.

Making mistakes in your first HTML pages and fixing them is a great way to learn. If you write your first pages perfectly, I'd recommend fiddling with the code as I have here to see how the browser reacts to various changes. This can be extremely useful in troubleshooting pages later. I've listed some common problems in the sidebar, [Having Problems?](#) Note that these problems are not specific to beginners! Little stuff like this goes wrong all the time, even for the pros.

HTML Review—Structural Tags

In this chapter, we covered some important tags for establishing the structure of the document. The remaining tags introduced in the demonstration will be treated in more depth in the following chapters.

<i>Tag</i>	<i>Function</i>
<HTML>	Identifies the whole document as HTML
<HEAD>	Identifies the head of the document
<BODY>	Identifies the body of the document
<TITLE>	Gives the page a title

