4TH EDITION

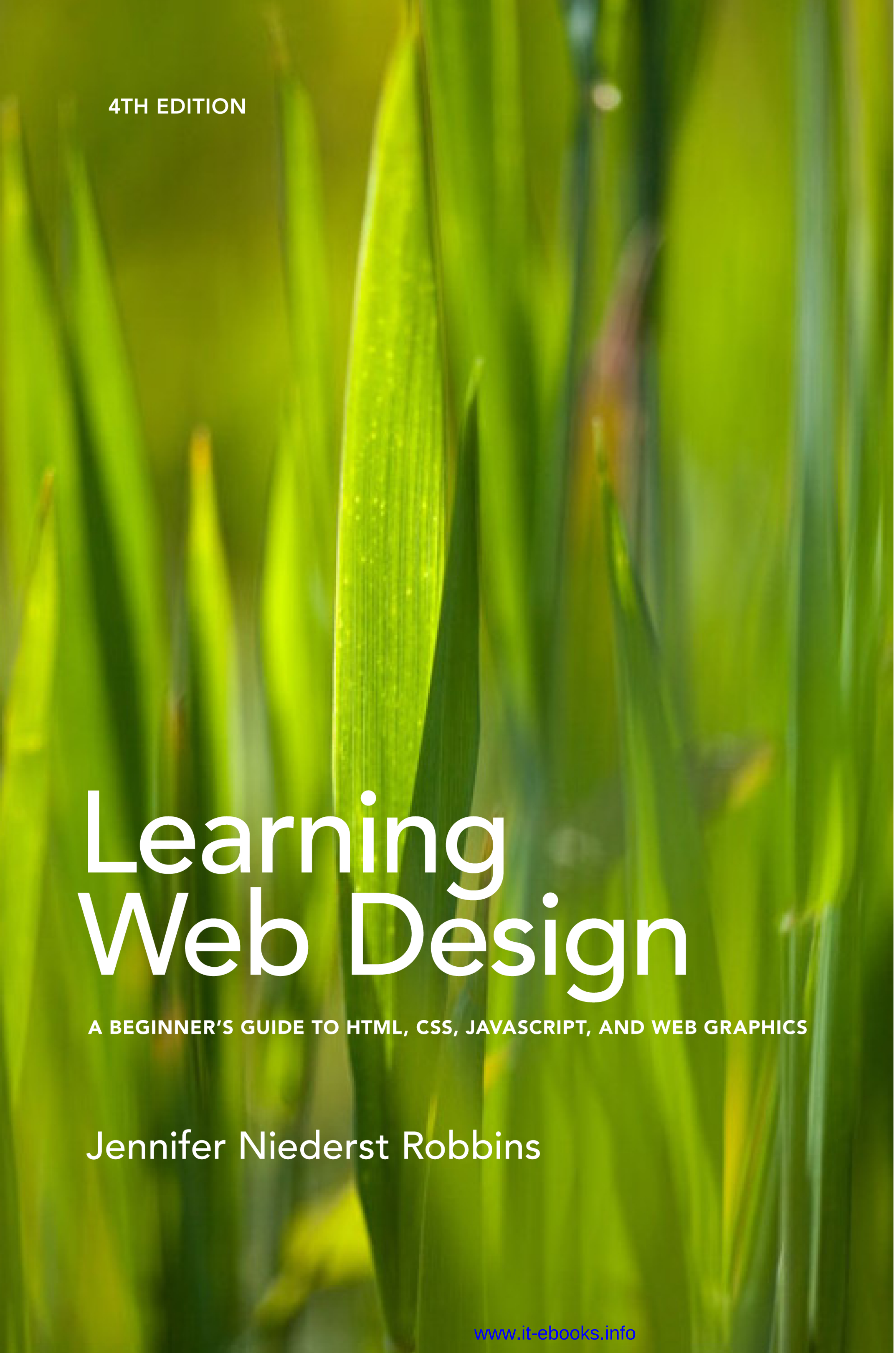
Learning Web Design

A BEGINNER'S GUIDE TO HTML, CSS, JAVASCRIPT, AND WEB GRAPHICS

Jennifer Niederst Robbins

Whether you're a beginner or bringing your skills up to date, this book gives you a solid footing in modern web production. I teach each topic visually at a pleasant pace, with frequent exercises to let you try out new skills. Reading it feels like sitting in my classroom! —Jennifer Robbins

O'REILLY®



CREATING A SIMPLE PAGE

(HTML Overview)

Part 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 start creating a real web page. It will be an extremely simple page, but even the most complicated pages are based on the principles described here.

In this chapter, we'll create a web page step by step so you can get a feel for what it's like to mark up a document with HTML tags. The exercises allow you to work along.

This is what I want you to get out of this chapter:

- Get a feel for how markup works, including an understanding of elements and attributes.
- See how browsers interpret HTML documents.
- Learn the basic structure of an HTML document.
- Get a first glimpse of a style sheet in action.

Don't worry about learning the specific text elements or style sheet rules at this point; we'll get to those in the following chapters. For now, just pay attention to the process, the overall structure of the document, and the new terminology.

IN THIS CHAPTER

An introduction to elements and attributes

A step-by-step demo of marking up a simple web page

The elements that provide document structure

A simple stylesheet

Troubleshooting broken web pages

A Web Page, Step by Step

You got a look at an HTML document in Chapter 2, How the Web Works, but now you'll get to create one yourself and play around with it in the browser. The demonstration in this chapter has five steps that cover the basics of page production.

Step 1: Start with content. As a starting point, we'll write up raw text content and see what browsers do with it.

Step 2: Give the document structure. You'll learn about HTML element syntax and the elements that give a document its structure.

Step 3: Identify text elements. You'll describe the content using the appropriate text elements and learn about the proper way to use HTML.

Step 4: Add an image. By adding an image to the page, you'll learn about attributes and empty elements.

Step 5: Change the page appearance with a style sheet. This exercise gives you a taste of formatting content with Cascading Style Sheets.

By the time we're finished, you will have written the source document for the page shown in Figure 4-1. It's not very fancy, but you have to start somewhere.

We'll be checking our work in a browser frequently throughout this demonstration—probably more than you would in real life. But because this is an introduction to HTML, it is helpful to see the cause and effect of each small change to the source file along the way.

HTML the Hard Way

I stand by my method of teaching HTML the old-fashioned way—*by hand*. There's no better way to truly understand how markup works 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 marking up documents properly.

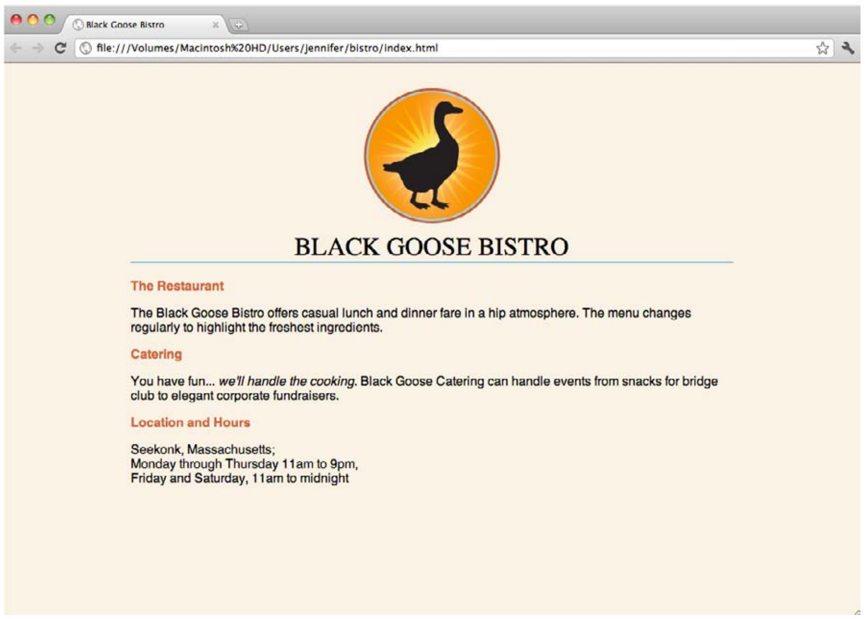
Although you may choose to use a web-authoring tool down the line, understanding HTML will make using your tools easier and more efficient. In addition, you will be glad that you can look at a source file and understand what you're seeing. It is also crucial for troubleshooting broken pages or fine-tuning the default formatting that web tools produce.

And for what it's worth, professional web developers tend to mark up content manually because it gives them better control over the code and allows them to make deliberate decisions about what elements are used.

Before We Begin, Launch a Text Editor

In this chapter and throughout the book, we'll be writing out HTML documents by hand, so the first thing we need to do is launch a text editor. The text editor that is provided with your operating system, such as Notepad (Windows) or TextEdit (Macintosh), will do for these purposes. Other text editors are fine as long as you can save plain text files with the *.html* extension. If you have a WYSIWYG web-authoring tool such as Dreamweaver, set it aside for now. I want you to get a feel for marking up a document manually (see the sidebar "HTML the Hard Way").

This section shows how to open new documents in Notepad and TextEdit. Even if you've used these programs before, skim through for some special settings that will make the exercises go more smoothly. We'll start with Notepad; Mac users can jump ahead.


Black Goose Bistro
file:///Volumes/Macintosh%20HD/Users/jennifer/bistro/index.html

BLACK GOOSE BISTRO

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

You have fun... *we'll handle the cooking*. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.

Location and Hours

Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm,
Friday and Saturday, 11am to midnight

Figure 4-1. In this chapter, we'll write the source document for this page step by step.

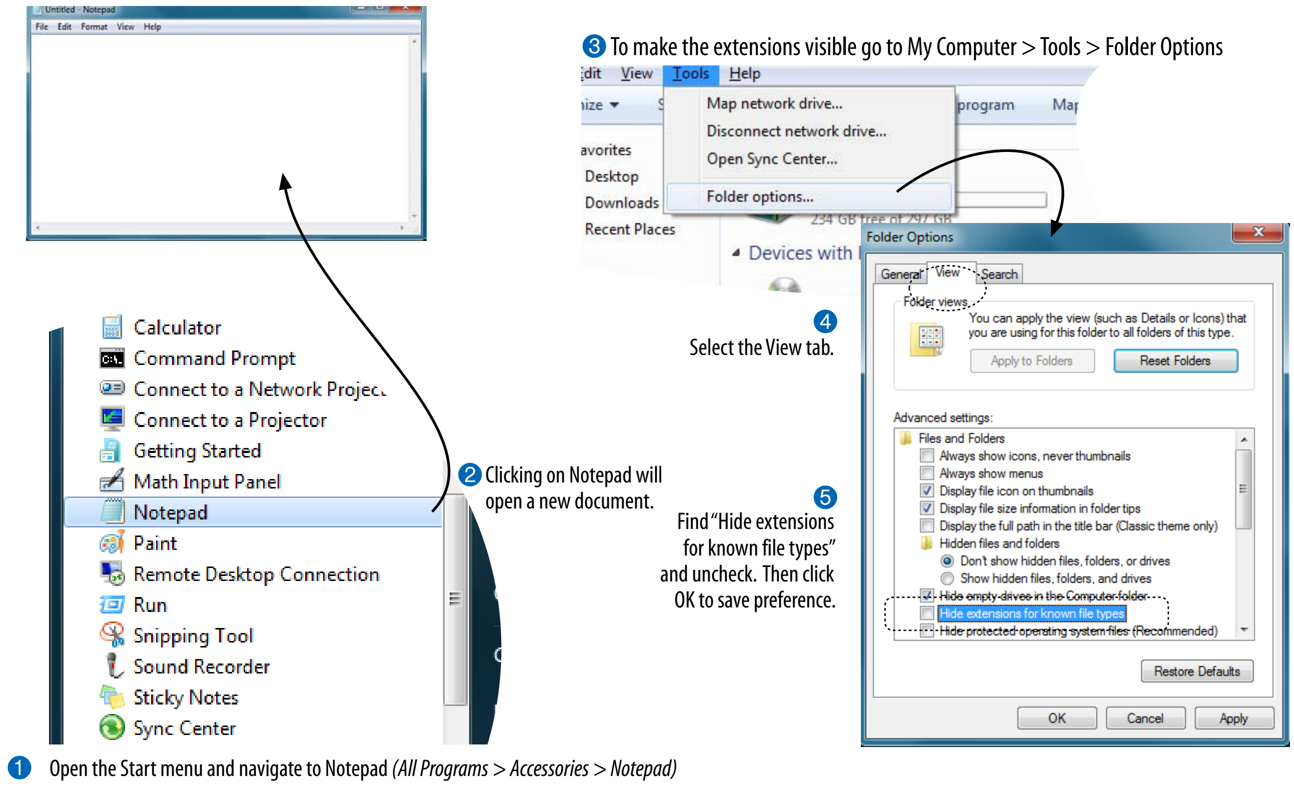
Creating a new document in Notepad (Windows)

These are the steps to creating a new document in Notepad on Windows 7 (Figure 4-2):

1. Open the Start menu and navigate to Notepad (in Accessories). ❶
2. Click on Notepad to open a new document window, and you're ready to start typing. ❷
3. Next, we'll make the extensions visible. This step is not required to make HTML documents, but it will help make the file types clearer at a glance. Select "Folder Options..." from the Tools menu ❸ and select the View tab ❹. Find "Hide extensions for known file types" and uncheck that option. ❺ Click OK to save the preference, and the file extensions will now be visible.

NOTE

In Windows 7, hit the ALT key to reveal the menu to access Tools and Folder Options. In Windows Vista, it is labeled "Folder and Search Options."



❸ To make the extensions visible go to My Computer > Tools > Folder Options

❹ Select the View tab.

❷ Clicking on Notepad will open a new document.

❺ Find "Hide extensions for known file types" and uncheck. Then click OK to save preference.

❶ Open the Start menu and navigate to Notepad *(All Programs > Accessories > Notepad)*

You can apply the view (such as Details or Icons) that you are using for this folder to all folders of this type.

Advanced settings:
- Files and Folders
  - [ ] Always show icons, never thumbnails
  - [ ] Always show menus
  - [✓] Display file icon on thumbnails
  - [✓] Display file size information in folder tips
  - [ ] Display the full path in the title bar (Classic theme only)
  - Hidden files and folders
    - ● Don't show hidden files, folders, or drives
    - ○ Show hidden files, folders, and drives
  - [✓] Hide empty drives in the Computer folder
  - [ ] Hide extensions for known file types
  - [✓] Hide protected operating system files (Recommended)

Figure 4-2. Creating a new document in Notepad.

Creating a new document in TextEdit (Mac OS X)

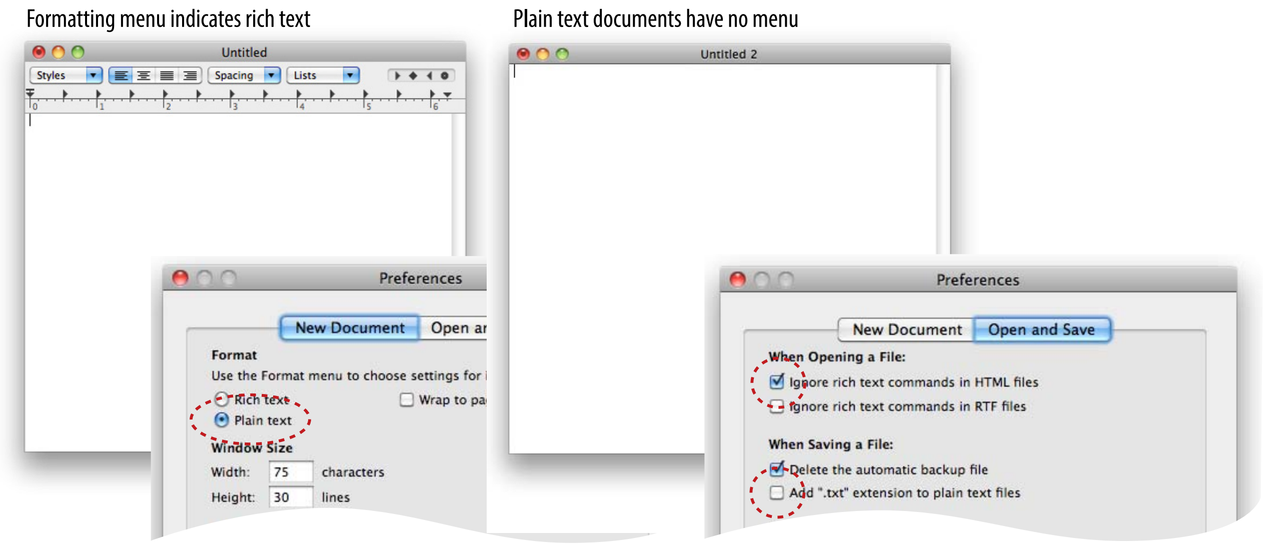
By default, TextEdit creates "rich text" documents, that is, documents that have hidden style formatting instructions for making text bold, setting font size, and so on. You can tell that TextEdit is in rich text mode when it has a formatting toolbar at the top of the window (plain text mode does not). HTML documents need to be plain text documents, so we'll need to change the Format, as shown in this example (Figure 4-3).

1. Use the Finder to look in the Applications folder for TextEdit. When you've found it, double-click the name or icon to launch the application.
2. TextEdit opens a new document. The text-formatting menu at the top shows that you are in Rich Text mode. Here's how you change it.
3. Open the Preferences dialog box from the TextEdit menu.
4. There are three settings you need to adjust:

   On the "New Document" tab, select "Plain text".

   On the "Open and Save" tab, select "Ignore rich text commands in HTML files" and turn off "Append '.txt' extensions to plain text files".
5. When you are done, click the red button in the top-left corner.
6. When you create a new document, the formatting menu will no longer be there and you can save your text as an HTML document. You can always convert a document back to rich text by selecting Format → Make Rich Text when you are not using TextEdit for HTML.

Figure 4-3. Launching TextEdit and choosing Plain Text settings in the Preferences.



Format
Use the Format menu to choose settings for
○ Rich text
◉ Plain text
☐ Wrap to pa
Window Size
Width: 75 characters
Height: 30 lines

When Opening a File:
☑ Ignore rich text commands in HTML files
☐ Ignore rich text commands in RTF files
When Saving a File:
☑ Delete the automatic backup file
☐ Add ".txt" extension to plain text files

Step 1: Start with Content

Now that we have our new document, it's time to get typing. A web page always starts with content, so that's where we begin our demonstration. Exercise 4-1 walks you through entering the raw text content and saving the document in a new folder.

exercise 4-1 | Entering content

1. Type the content below for the home page into the new document in your text editor. Copy it exactly as you see it here, keeping the line breaks the same for the sake of playing along. The raw text for this exercise is available online at *www.learningwebdesign.com/4e/materials/*.

```
Black Goose Bistro

The Restaurant
The Black Goose Bistro offers casual lunch and dinner fare in hip
atmosphere. The menu changes regularly to highlight the freshest
ingredients.

Catering
You have fun... we'll handle the cooking. Black Goose Catering
can handle events from snacks for bridge club to elegant corporate
fundraisers.

Location and Hours
Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm, Friday and Saturday, 11am to
midnight
```

2. Select "Save" or "Save as" from the File menu to get the Save As dialog box (Figure 4-4). The first thing you need to do is create a new folder that will contain all of the files for the site (in other words, it's the local root folder).

   Windows: Click the folder icon at the top to create the new folder.

   Mac: Click the "New Folder" button.

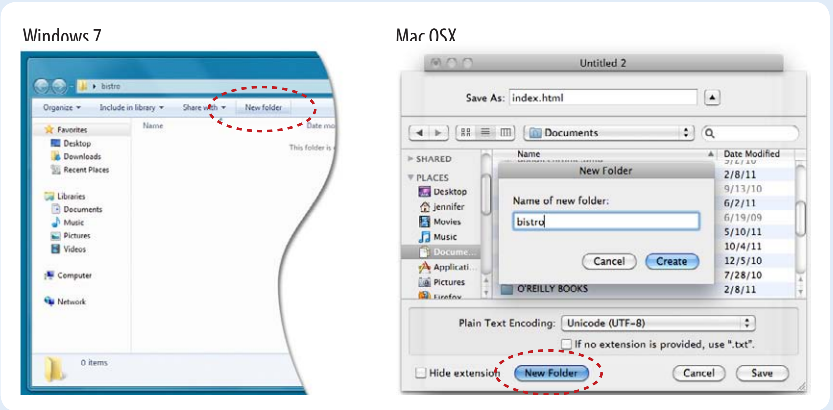


Figure 4-4. Saving index.html in a new folder called "bistro".

Naming Conventions

It is important that you follow these rules and conventions when naming your files:

Use proper suffixes for your files. HTML and XHTML files must end with .html. Web graphics must be labeled according to their file format: *.gif*, *.png*, or *.jpg* (*.jpeg* is also acceptable).

Never use character spaces within filenames. It is common to use an underline character or hyphen to visually separate words within filenames, such as *robbins_bio.html* or *robbins-bio.html*.

Avoid special characters such as **?**, **%**, **#**, **/**, **:**, **;**, **•**, etc. Limit filenames to letters, numbers, underscores, hyphens, and periods.

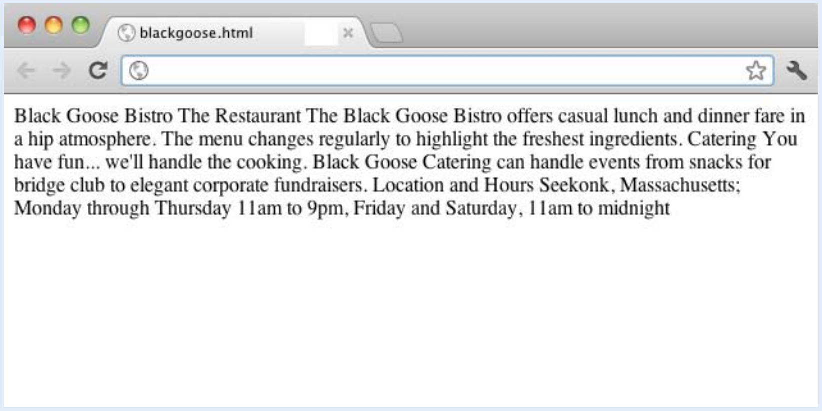
Filenames may be case-sensitive, depending on your server configuration. Consistently using all lowercase letters in filenames, although not necessary, is one way to make your filenames easier to manage.

Keep filenames short. Short names keep the character count and file size of your HTML file in check. If you really must give the file a long, multiword name, you can separate words with hyphens, such as *a-long-document-title.html*, to improve readability.

Self-imposed conventions. It is helpful to develop a consistent naming scheme for huge sites. For instance, always using lowercase with hyphens between words. This takes some of the guesswork out of remembering what you named a file when you go to link to it later.

Name the new folder ***bistro***, and save the text file as ***index.html*** in it. Windows users, you will also need to choose "All Files" after "Save as type" to prevent Notepad from adding a ".txt" extension to your filename. The filename needs to end in ***.html*** to be recognized by the browser as a web document. See the sidebar "Naming Conventions" for more tips on naming files.

3. Just for kicks, let's take a look at ***index.html*** in a browser. Launch your favorite browser (I'm using Google Chrome) and choose "Open" or "Open File" from the File menu. Navigate to ***index.html***, and then select the document to open it in the browser. You should see something like the page shown in Figure 4-5. We'll talk



Black Goose Bistro The Restaurant The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients. Catering You have fun... we'll handle the cooking. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers. Location and Hours Seekonk, Massachusetts; Monday through Thursday 11am to 9pm, Friday and Saturday, 11am to midnight

Figure 4-5. *A first look at the content in a browser.*

What Browsers Ignore

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

Multiple (white) spaces. When a browser encounters more than one consecutive blank character space, it displays a single space. So if the document contains:

```
long,     long      ago
```

the browser displays:

long, long ago

Line breaks (carriage returns). Browsers convert carriage returns to white spaces, so following the earlier "ignore multiple white spaces rule," line breaks have no effect on formatting the page. Text and elements wrap continuously until a new block element, such as a heading (`h1`) or paragraph (`p`), or the line break (`br`) element is encountered in the flow of the document text.

Tabs. Tabs are also converted to character spaces, so guess what? Useless.

Unrecognized markup. Browsers are instructed to ignore any tag they don't understand or that was specified incorrectly. Depending on the element and the browser, this can have varied results. The browser may display 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 `-->` tags used to denote a comment. See the Adding Hidden Comments sidebar later in this chapter.

Learning from step 1

Our content isn't looking so good (Figure 4-5). The text is all run together—that's not how it looked in the original document. There are a couple of things to be learned here. The first thing that is apparent is that the browser ignores line breaks in the source document. The sidebar "What Browsers Ignore" lists other information in the source that is not displayed in the browser window.

Second, we see that simply typing in some content and naming the document *.html* is not enough. While the browser can display the text from the file, we haven't indicated the *structure* of the content. That's where HTML comes in. We'll use markup to add structure: first to the HTML document itself (coming up in Step 2), then to the page's content (Step 3). Once the browser knows the structure of the content, it can display the page in a more meaningful way.

Step 2: Give the Document Structure

We have our content saved in an *.html* document—now we're ready to start marking it up.

Introducing…HTML elements

Back in Chapter 2, How the Web Works, you saw examples of HTML elements with an opening tag (**<p>** for a paragraph, for example) and closing tag (**</p>**). Before we start adding tags to our document, let's look at the anatomy of an HTML element (its syntax) and firm up some important terminology. A generic container element is labeled in Figure 4-6.

An element consists of both the content and its markup.

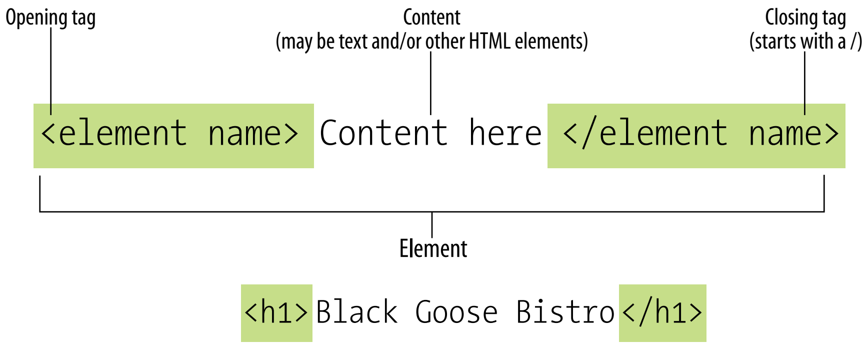

Opening tag
Content
(may be text and/or other HTML elements)
Closing tag
(starts with a /)
<element name> Content here </element name>
Element
<h1>Black Goose Bistro</h1>

***Figure 4-6.** The parts of an HTML container element.*

Elements are identified by tags in the text source. A tag consists of the element name (usually an abbreviation of a longer descriptive name) within angle brackets (**<** **>**). The browser knows that any text within brackets is hidden and not displayed in the browser window.

The element name appears in the opening tag (also called a start tag) and again in the closing (or end) tag preceded by a slash (**/**). The closing tag works something like an "off" switch for the element. Be careful not to use the similar backslash character in end tags (see the tip Slash vs. Backslash).

The tags added around content are referred to as the markup. It is important to note that an element consists of both the content *and* its markup (the start and end tags). Not all elements have content, however. Some are empty by definition, such as the **img** element used to add an image to the page. We'll talk about empty elements a little later in this chapter.

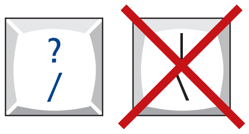
One last thing...capitalization. In HTML, the capitalization of element names is not important. So **<img>**, **<Img>**, and **<IMG>** are all the same as far as the browser is concerned. However, in XHTML (the stricter version of HTML) all element names must be all lowercase in order to be valid. Many web developers have come to like the orderliness of the stricter XHTML markup rules and stick with all lowercase, as I will do in this book.

TIP

Slash vs. Backslash

HTML tags and URLs use the slash character (/). The slash character is found under the question mark (?) on the standard QWERTY keyboard.

It is easy to confuse the slash with the backslash character (\), which is found under the bar character (|). The backslash key will not work in tags or URLs, so be careful not to use it.



Basic document structure

Figure 4-7 shows the recommended minimal skeleton of an HTML5 document. I say "recommended" because the only element that is *required* in HTML is the **`title`**. But I feel it is better, particularly for beginners, to explicitly organize documents with the proper structural markup. And if you are writing in the stricter XHTML, all of the following elements except **`meta`** must be included in order to be valid. Let's take a look at what's going on in Figure 4-7.

❶ I don't want to confuse things, but the first line in the example isn't an element at all; it is a document type declaration (also called DOCTYPE declaration) that identifies this document as an HTML5 document. I have a lot more to say about DOCTYPE declarations in Chapter 10, What's Up, HTML5?, but for this discussion, suffice it to say that including it lets modern browsers know they should interpret the document as written according to the HTML5 specification.

❷ The entire document is contained within an **`html`** element. The **`html`** element is called the root element because it contains all the elements in the document, and it may not be contained within any other element. It is used for both HTML and XHTML documents.

❸ Within the **`html`** element, the document is divided into a head and a body. The **`head`** element contains descriptive information about the document itself, such as its title, the style sheet(s) it uses, scripts, and other types of "meta" information.

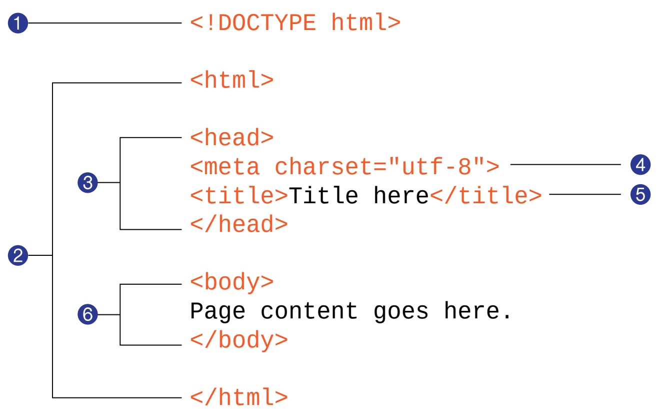
❹ The **`meta`** elements within the **`head`** element provide information *about* the document itself. A **`meta`** element can be used to provide all sorts of information, but in this case, it specifies the character encoding (the standardized collection of letters, numbers, and symbols) used in the document. I don't want to go into too much detail on this right now, but know that there are many good reasons for specifying the **`charset`** in every document, so I have included it as part of the minimal document structure.

NOTE

Prior to HTML5, the syntax for specifying the character set with the `meta` *element was a bit more elaborate. If you are writing your documents in HTML 4.01 or XHTML 1.0, your* `meta` *element should look like this:*

```
<meta http-equiv="content-
type" content="text/html;
charset=UTF-8">
```

Figure 4-7. The minimal structure of an HTML document.



```
<!DOCTYPE html>

<html>

<head>
<meta charset="utf-8">
<title>Title here</title>
</head>

<body>
Page content goes here.
</body>

</html>
```

❺ Also in the **`head`** is the mandatory **`title`** element. According to the HTML specification, every document must contain a descriptive title.

❻ Finally, the **`body`** element contains everything that we want to show up in the browser window.

Are you ready to add some structure to the Black Goose Bistro home page? Open the *index.html* document and move on to Exercise 4-2.

exercise 4-2 | Adding basic structure

1. Open the newly created document, *index.html*, if it isn't open already.
2. Start by adding the HTML5 DOCTYPE declaration:

```
<!DOCTYPE html>
```

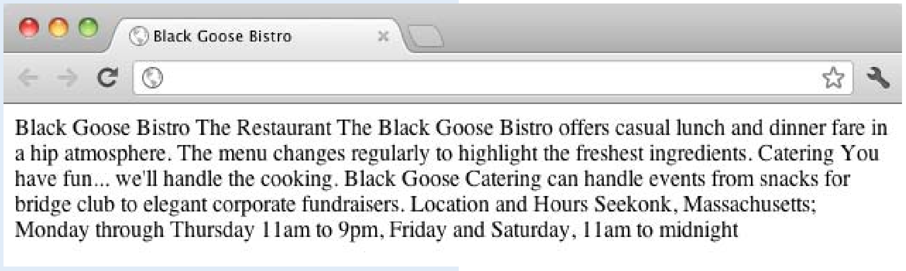
3. Put the entire document in an HTML root element by adding an **`<html>`** start tag at the very beginning and an end **`<html>`** tag at the end of the text.
4. Next, created the document head that contains the title for the page. Insert **`<head>`** and **`</head>`** tags before the content. Within the head element, add informatino about the character encoding **`<meta charset="utf-8">`**, and the title, "Black Goose Bistro", surrounded by opening and closing **`<title>`** tags.

   The correct terminology is to say that the **`title`** *element is nested within the* **`head`** *element. We'll talk about nesting more in later chapters.*

5. Finally, define the body of the document by wrapping the content in **`<body>`** and **`</body>`** tags. When you are done, the source document should look like this (the markup is shown in color to make it stand out):

```
<!DOCTYPE html>
<html>

<head>
<meta charset ="utf-8">
<title>Black Goose Bistro</title>
</head>

<body>
Black Goose Bistro

The Restaurant
The Black Goose Bistro offers casual lunch and dinner fare in
a hip atmosphere. The menu changes regularly to highlight the
freshest ingredients.

Catering Services
You have fun... we'll do the cooking. Black Goose catering can
handle events from snacks for bridge club to elegant corporate
fundraisers.
Location and Hours
Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm, Friday and Saturday, 11am to
midnight
</body>

</html>
```

6. Save the document in the bistro directory, so that it overwrites the old version. Open the file in the browser or hit "refresh" or "reload" if it is open already. Figure 4-8 shows how it should look now.



Black Goose Bistro The Restaurant The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients. Catering You have fun... we'll handle the cooking. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers. Location and Hours Seekonk, Massachusetts; Monday through Thursday 11am to 9pm, Friday and Saturday, 11am to midnight

Figure 4-8. The page in a browser after the document structure elements have been defined.

Not much has changed after structuring the document, except that the browser now displays the title of the document in the top bar or tab. If someone were to bookmark this page, that title would be added to his Bookmarks or Favorites list as well (see the sidebar Don't Forget a Good Title). But the content still runs together because we haven't given the browser any indication of how it should be structured. We'll take care of that next.

Step 3: Identify Text Elements

With a little markup experience under your belt, it should be a no-brainer to add the markup that identifies headings and subheads (**`h1`** and **`h2`**), paragraphs (**`p`**), and emphasized text (**`em`**) to our content, as we'll do in Exercise 4-3. However, before we begin, I want to take a moment to talk about what we're doing and not doing when marking up content with HTML.

Don't Forget a Good Title

Not only is a **`title`** element required for every document, it is quite useful as well. The title is what is displayed in a user's Bookmarks or Favorites list and on tabs in desktop browsers. Descriptive titles are also a key tool for improving accessibility, as they are the first thing a person hears when using a screen reader. Search engines rely heavily on document titles as well. For these reasons, it's important to provide thoughtful and descriptive titles for all your documents and avoid vague titles, such as "Welcome" or "My Page." You may also want to keep the length of your titles in check so they are able to display in the browser's title area. Another best practice is to put the part of the title with more specific information first (for example, the page description ahead of the company name) so that the page title is visible when multiple tabs are lined up in the browser window.

Introducing…semantic markup

The purpose of HTML is to add meaning and structure to the content. It is *not* intended to provide instructions for how the content should look (its presentation).

Your job when marking up content is to choose the HTML element that provides the most meaningful description of the content at hand. In the biz, we call this semantic markup. For example, the most important heading at the beginning of the document should be marked up as an **`h1`** because it is the most important heading on the page. Don't worry about what that looks like in the browser...you can easily change that with a style sheet. The important thing is that you choose elements based on what makes the most sense for the content.

In addition to adding meaning to content, the markup gives the document structure. The way elements follow each other or nest within one another creates relationships between the elements. You can think of it as an outline (its technical name is the DOM, for Document Object Model). The underlying document hierarchy is important because it gives browsers cues on how to handle the content. It is also the foundation upon which we add presentation instructions with style sheets and behaviors with JavaScript. We'll talk about document structure more in Part III, when we discuss Cascading Style Sheets, and in Part IV in the JavaScript overview.

Although HTML was intended to be used strictly for meaning and structure since its creation, that mission was somewhat thwarted in the early years of the web. With no style sheet system in place, HTML was extended to give authors ways to change the appearance of fonts, colors, and alignment using markup alone. Those presentational extras are still out there, so you may run across them if you view the source of older sites or a site made with old tools.

In this book, however, we'll focus on using HTML the right way, in keeping with the contemporary standards-based, semantic approach to web design.

OK, enough lecturing. It's time to get to work on that content in Exercise 4-3.

exercise 4-3 | Defining text elements

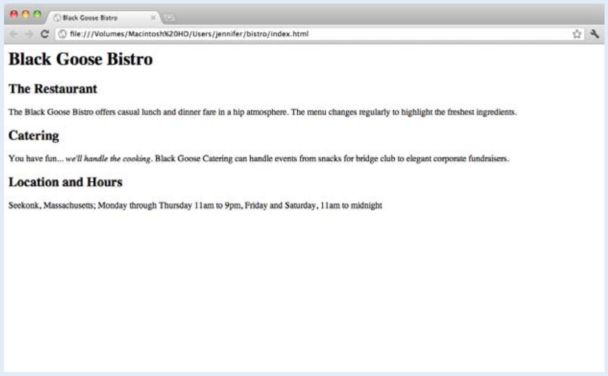
1. Open the document *index.html* in your text editor, if it isn't open already.
2. The first line of text, "Black Goose Bistro," is the main heading for the page, so we'll mark it up as a Heading Level 1 (**h1**) element. Put the opening tag, **<h1>**, at the beginning of the line and the closing tag, **</h1>**, after it, like this:

   ```
   <h1>Black Goose Bistro</h1>
   ```
3. Our page also has three subheads. Mark them up as Heading Level 2 (**h2**) elements in a similar manner. I'll do the first one here; you do the same for "Catering" and "Location and Hours".

   ```
   <h2>The Restaurant</h2>
   ```
4. Each **h2** element is followed by a brief paragraph of text, so let's mark those up as paragraph (**p**) elements in a similar manner. Here's the first one; you do the rest.

   ```
   <p>The Black Goose Bistro offers casual lunch and
   dinner fare in a hip atmosphere. The menu changes
   regularly to highlight the freshest ingredients.
   </p>
   ```
5. Finally, in the Catering section, I want to emphasize that visitors should just leave the cooking to us. To make text emphasized, mark it up in an emphasis element (**em**) element, as shown here.

   ```
   <p>You have fun... <em>we'll handle the cooking
   </em>. Black Goose Catering can handle events
   from snacks for bridge club to elegant corporate
   fundraisers.</p>
   ```
6. Now that we've marked up the document, let's save it as we did before, and open (or refresh) the page in the browser. You should see a page that looks much like the one in Figure 4-9. If it doesn't, check your markup to be sure that you aren't missing any angle brackets or a slash in a closing tag.


Black Goose Bistro
file:///Volumes/Macintosh%20HD/Users/jennifer/bistro/index.html

Black Goose Bistro

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

You have fun... *we'll handle the cooking*. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.

Location and Hours

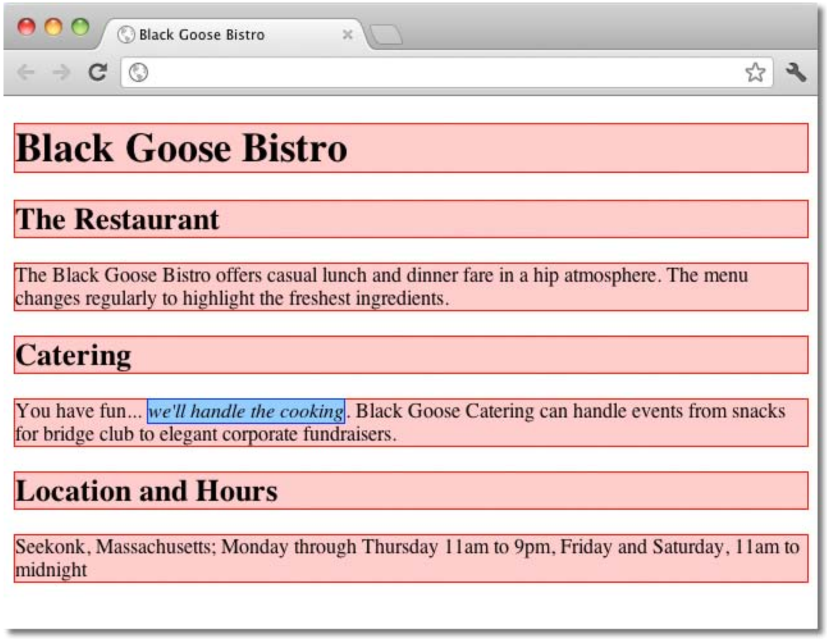
Seekonk, Massachusetts; Monday through Thursday 11am to 9pm, Friday and Saturday, 11am to midnight

Figure 4-9. The home page after the content has been marked up with HTML elements.

Now we're getting somewhere. With the elements properly identified, the browser can now display the text in a more meaningful manner. There are a few significant things to note about what's happening in Figure 4-9.

Block and inline elements

Although it may seem like stating the obvious, it is worth pointing out that the heading and paragraph elements start on new lines and do not run together as they did before. That is because by default, headings and paragraphs display as block elements. Browsers treat block elements as though they are in little rectangular boxes, stacked up in the page. Each block element begins on a new line, and some space is also usually added above and below the entire element by default. In Figure 4-10, the edges of the block elements are outlined in red.



Black Goose Bistro

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

You have fun... *we'll handle the cooking*. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.

Location and Hours

Seekonk, Massachusetts; Monday through Thursday 11am to 9pm, Friday and Saturday, 11am to midnight

Figure 4-10. The outlines show the structure of the elements in the home page.

Adding Hidden Comments

You can leave notes in the source document for yourself and others by marking them up as comments. Anything you put between comment tags (`<!-- -->`) will not display in the browser and will not have any effect on the rest of the source.

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

Comments are useful for labeling and organizing long documents, particularly when they are shared by a team of developers. In this example, comments are used to point out the section of the source that contains the navigation.

```
<!-- start global nav -->
<ul>
 ...
</ul>
<!-- end global nav -->
```

Bear in mind that although the browser will not display comments in the web page, readers can see them if they "view source," so be sure that the comments you leave are appropriate for everyone. It's probably a good idea just to strip out notes to your fellow developers before the site is published. It cuts some bytes off the file size as well.

By contrast, look at the text we marked up as emphasized (`em`). It does not start a new line, but rather stays in the flow of the paragraph. That is because the `em` element is an inline element. Inline elements do not start new lines; they just go with the flow. In Figure 4-10, the inline `em` element is outlined in light blue.

Default styles

The other thing that you will notice about the marked-up page in Figures 4-9 and 4-10 is that the browser makes an attempt to give the page some visual hierarchy by making the first-level heading the biggest and boldest thing on the page, with the second-level headings slightly smaller, and so on.

How does the browser determine what an `h1` should look like? It uses a style sheet! All browsers have their own built-in style sheets (called user agent style sheets in the spec) that describe the default rendering of elements. The default rendering is similar from browser to browser (for example, `h1`s are always big and bold), but there are some variations (long quotes may or may not be indented).

If you think the `h1` is too big and clunky as the browser renders it, just change it with a style sheet rule. Resist the urge to mark up the heading with another element just to get it to look better, for example, using an `h3` instead of an `h1` so it isn't as large. In the days before ubiquitous style sheet support, elements were abused in just that way. Now that there are style sheets for controlling the design, you should always choose elements based on how accurately they describe the content, and don't worry about the browser's default rendering.

We'll fix the presentation of the page with style sheets in a moment, but first, let's add an image to the page.

Step 4: Add an Image

What fun is a web page with no image? In Exercise 4-4, we'll add an image to the page using the **`img`** element. Images will be discussed in more detail in Chapter 7, Adding Images, but for now, it gives us an opportunity to introduce two more basic markup concepts: empty elements and attributes.

Empty elements

So far, nearly all of the elements we've used in the Black Goose Bistro home page have followed the syntax shown in Figure 4-1: a bit of text content surrounded by start and end tags.

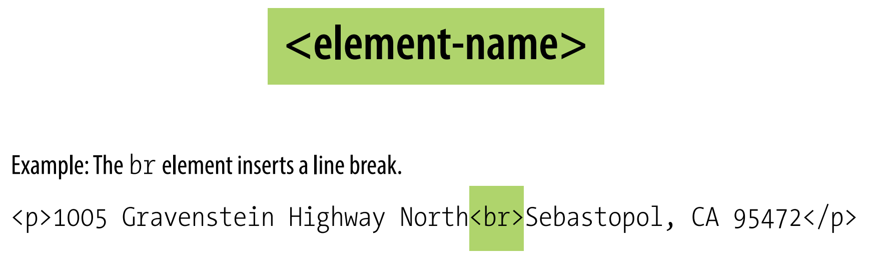
A handful of elements, however, do not have text content because they are used to provide a simple directive. These elements are said to be empty. The image element (**`img`**) is an example of such an element; it tells the browser to get an image file from the server and insert it at that spot in the flow of the text. Other empty elements include the line break (**`br`**), thematic breaks (**`hr`**), and elements that provide information about a document but don't affect its displayed content, such as the **`meta`** element that we used earlier.

Figure 4-11 shows the very simple syntax of an empty element (compare to Figure 4-4). If you are writing an XHTML document, the syntax is slightly different (see the sidebar Empty Elements in XHTML).

Empty Elements in XHTML

In XHTML, all elements, including empty elements, must be closed (or terminated, to use the proper term). Empty elements are terminated by adding a trailing slash preceded by a space before the closing bracket, like so: **`<img />`**, **`<br />`**, **`<meta />`**, and **`<hr />`**. Here is the line break example using XHTML syntax.

```
<p>1005 Gravenstein Highway
North <br />Sebastopol, CA
95472</p>
```



Example: The `br` element inserts a line break.

```
<p>1005 Gravenstein Highway North<br>Sebastopol, CA 95472</p>
```

Figure 4-11. *Empty element structure.*

Attributes

Let's get back to adding an image with the empty **`img`** element. Obviously, an **`<img>`** tag is not very useful by itself—there's no way to know which image to use. That's where attributes come in. Attributes are instructions that clarify or modify an element. For the **`img`** element, the **`src`** (short for "source") attribute is required, and specifies the location (URL) of the image file.

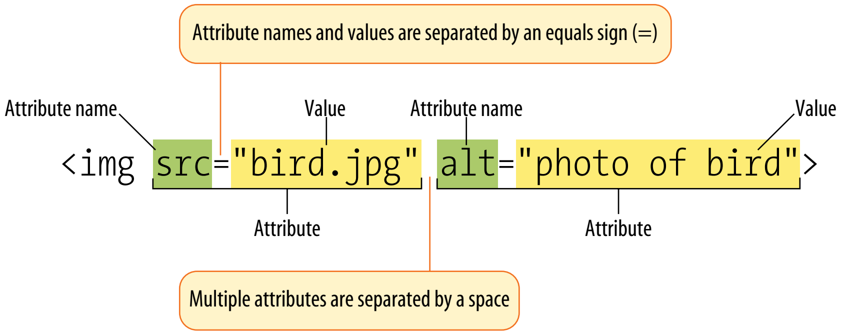

Attribute names and values are separated by an equals sign (=)
Attribute name
Value
Attribute name
Value
<img src="bird.jpg" alt="photo of bird">
Attribute
Attribute
Multiple attributes are separated by a space

Figure 4-12. An `img` element with attributes.

The syntax for an attribute is as follows:

```
attributename="value"
```

Attributes go after the element name, separated by a space. In non-empty elements, attributes go in the opening tag only:

```
<element attributename="value">
<element attributename="value">Content</element>
```

You can also put more than one attribute in an element in any order. Just keep them separated with spaces.

```
<element attribute1="value" attribute2="value">
```

For another way to look at it, Figure 4-12 shows an **`img`** element with its required attributes labeled.

Here's what you need to know about attributes:

- Attributes go after the element name in the opening tag only, never in the end tag.
- There may be several attributes applied to an element, separated by spaces in the opening tag. Their order is not important.
- Most attributes take values, which follow an equals sign (=). In HTML, some attribute values can be reduced to single descriptive words, for example, the **`checked`** attribute, which makes a checkbox checked when a form loads. In XHTML, however, all attributes must have explicit values (**`checked="checked"`**). You may hear this type of attribute called a Boolean attribute because it describes a feature that is either on or off.
- A value might be a number, a word, a string of text, a URL, or a measurement, depending on the purpose of the attribute. You'll see examples of all of these throughout this book.
- Some values don't have to be in quotation marks in HTML, but XHTML requires them. Many developers like the consistency and tidiness of quotation marks even when authoring HTML. Either single or double quotation marks are acceptable as long as they are used consistently; however,

double quotation marks are the convention. Note that quotation marks in HTML files need to be straight (") not curly (”).

- Some attributes are required, such as the **src** and **alt** attributes in the **img** element.
- The attribute names available for each element are defined in the HTML specifications; in other words, you can't make up an attribute for an element.

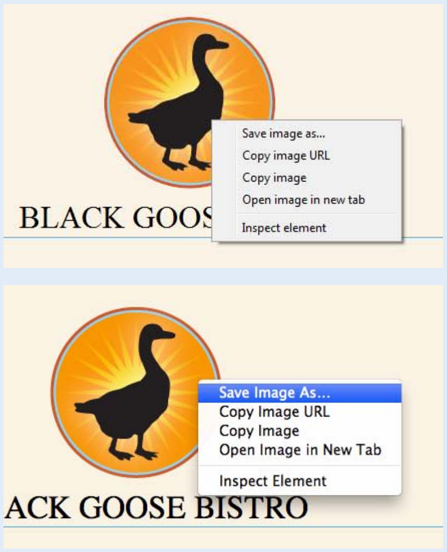
Now you should be more than ready to try your hand at adding the **img** element with its attributes to the Black Goose Bistro page in the next exercise. We'll throw a few line breaks in there as well.

exercise 4-4 | Adding an image

1. If you're working along, the first thing you'll need to do is get a copy of the image file on your hard drive so you can see it in place when you open the file locally. The image file is provided in the materials for this chapter. You can also get the image file by saving it right from the sample web page online at *www.learningwebdesign.com/4e/chapter04/bistro*. Right-click (or Ctrl-click on a Mac) on the goose image and select "Save to disk" (or similar) from the pop-up menu as shown in Figure 4-13. Name the file ***blackgoose.png***. Be sure to save it in the ***bistro*** folder with ***index.html***.
2. Once you have the image, insert it at the beginning of the first-level heading by typing in the **img** element and its attributes as shown here:

```
<h1><img src="blackgoose.png" alt="Black Goose logo">Black Goose
Bistro</h1>
```

The **src** attribute provides the name of the image file that should be inserted, and the **alt** attribute provides text that should be displayed if the image is not available. Both of these attributes are required in every **img** element.



Windows:
Right-click on the image to access the pop-up menu

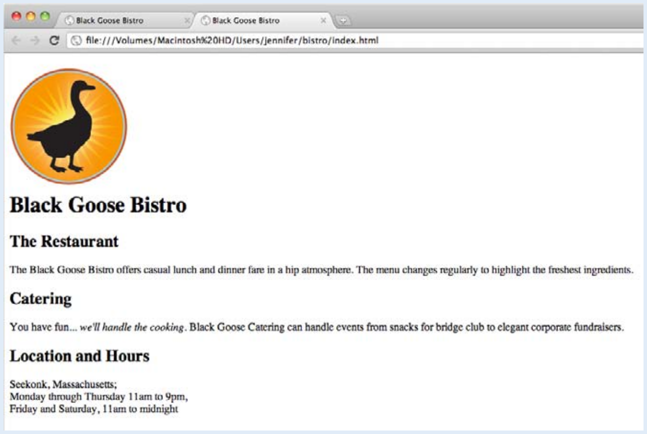
Mac:
Control-click on the image to access the popup menu. The options my vary by browser.

Figure 4-13. Saving an image file from a page on the Web.

3. I'd like the image to appear above the title, so lets add a line break (**`br`**) after the **`img`** element to start the headline text on a new line.

```
<h1><img src="blackgoose.png" alt="Black Goose logo"><br>Black
Goose Bistro</h1>
```

4. Let's break up the last paragraph into three lines for better clarity. Drop a **`<br>`** tag at the spots you'd like the line breaks to occur. Try to match the screenshot in Figure 4-14.
5. Now save ***index.html*** and open or refresh it in the browser window. The page should look like the one shown in Figure 4-14. If it doesn't, check to make sure that the image file, ***blackgoose.png***, is in the same directory as ***index.html***. If it is, then check to make sure that you aren't missing any characters, such as a closing quote or bracket, in the **`img`** element markup.


Black Goose Bistro
Black Goose Bistro
file:///Volumes/Macintosh%20HD/Users/jennifer/bistro/index.html

Black Goose Bistro

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

You have fun... *we'll handle the cooking*. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.

Location and Hours

Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm,
Friday and Saturday, 11am to midnight

Figure 4-14. *The Black Goose Bistro page with the logo image.*

Step 5: Change the Look with a Style Sheet

Depending on the content and purpose of your website, you may decide that the browser's default rendering of your document is perfectly adequate. However, I think I'd like to pretty up the Black Goose Bistro home page a bit to make a good first impression on potential patrons. "Prettying up" is just my way of saying that I'd like to change its presentation, which is the job of Cascading Style Sheets (CSS).

In Exercise 4-5, we'll change the appearance of the text elements and the page background using some simple style sheet rules. Don't worry about understanding them all right now; we'll get into CSS in more detail in Part III. But I want to at least give you a taste of what it means to add a "layer" of presentation onto the structure we've created with our markup.

exercise 4-5 | Adding a style sheet

1. Open *index.html* if it isn't open already.
2. We're going to use the **style** element to apply a very simple embedded style sheet to the page. (This is just one of the ways to add a style sheet; the others are covered in Chapter 11, Style Sheet Orientation.)

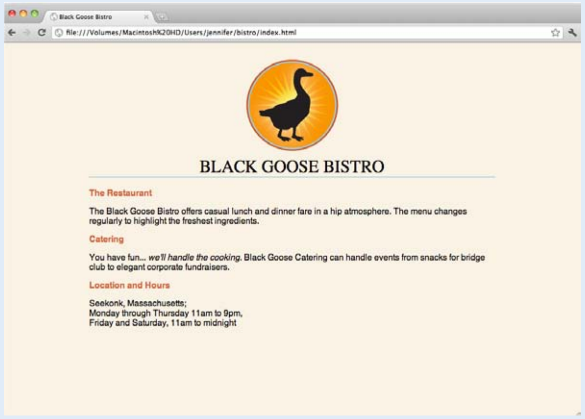
   The **style** element is placed inside the **head** of the document. Start by adding the **style** element to the document as shown here:

```
<head>
  <meta charset="utf-8">
  <title>Black Goose Bistro</title>
  <style>

  </style>
</head>
```

3. Now, type the following style rules within the **style** element just as you see them here. Don't worry if you don't know exactly what is going on (although it is fairly intuitive). You'll learn all about style rules in Part III.

```
<style>

body {
  background-color: #faf2e4;
  margin: 0 15%;
  font-family: sans-serif;
  }

h1 {
  text-align: center;
  font-family: serif;
  font-weight: normal;
  text-transform: uppercase;
  border-bottom: 1px solid #57b1dc;
  margin-top: 30px;
}

h2 {
  color: #d1633c;
  font-size: 1em;
}

</style>
```

4. Now it's time to save the file and take a look at it in the browser. It should look like the page in Figure 4-15. If it doesn't, go over the style sheet code to make sure you didn't miss a semicolon or a curly bracket.


Black Goose Bistro
file:///Volumes/Macintosh%20HD/Users/jennifer/bistro/index.html

BLACK GOOSE BISTRO

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

You have fun... *we'll handle the cooking*. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.

Location and Hours

Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm,
Friday and Saturday, 11am to midnight

Figure 4-15. The Black Goose Bistro page after CSS style rules have been applied.

We're finished with the Black Goose Bistro page. Not only have you written your first web page, complete with a style sheet, but you've learned about elements, attributes, empty elements, block and inline elements, the basic structure of an HTML document, and the correct use of markup along the way. Not bad for one chapter!

When Good Pages Go Bad

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

NOTE

Omitting the slash in the closing tag (or even omitting the closing tag itself) for block elements, such as headings or paragraphs, may not be so dramatic. Browsers interpret the start of a new block element to mean that the previous block element is finished.

What if I had forgotten to type the slash (/) in the closing emphasis tag (`</em>`)? With just one character out of place (Figure 4-16), the remainder of the document displays in emphasized (italic) text. That's because without that slash, there's nothing telling the browser to turn "off" the emphasized formatting, so it just keeps going.

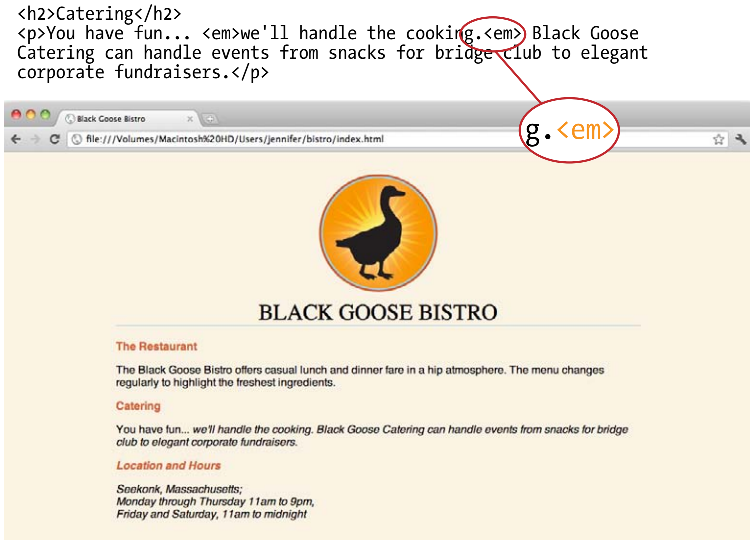
I've fixed the slash, but this time, let's see what would have happened if I had accidentally omitted a bracket from the end of the first `<h2>` tag (Figure 4-17).

See how the headline 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 the `<h2>` opening tag. Browsers don't display any text within a tag, so my heading disappeared. The browser just ignored the foreign-looking element name and moved on to the next element.

Making mistakes in your first HTML documents 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.

Figure 4-16. *When a slash is omitted, the browser doesn't know when the element ends, as is the case in this example.*

```
<h2>Catering</h2>
<p>You have fun... <em>we'll handle the cooking.<em> Black Goose
Catering can handle events from snacks for bridge club to elegant
corporate fundraisers.</p>
```


g.<em>
Black Goose Bistro
file:///Volumes/Macintosh%20HD/Users/jennifer/bistro/index.html

BLACK GOOSE BISTRO

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

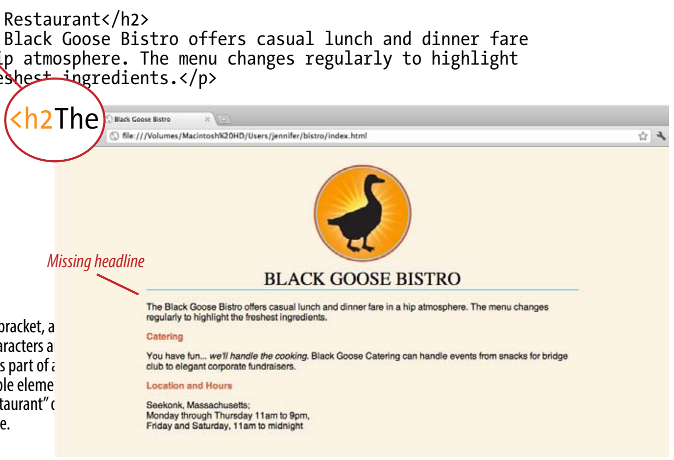
You have fun... *we'll handle the cooking. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.*

Location and Hours

Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm,
Friday and Saturday, 11am to midnight

Figure 4-17. *A missing end bracket makes all the following content part of the tag, and therefore it doesn't display.*

```
<h2The Restaurant</h2>
<p>The Black Goose Bistro offers casual lunch and dinner fare
in a hip atmosphere. The menu changes regularly to highlight
the freshest ingredients.</p>
```


<h2The
Black Goose Bistro
file:///Volumes/Macintosh%20HD/Users/jennifer/bistro/index.html
Missing headline

BLACK GOOSE BISTRO

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

You have fun... *we'll handle the cooking.* Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.

Location and Hours

Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm,
Friday and Saturday, 11am to midnight

Without the bracket, a
following characters a
interpreted as part of a
unrecognizable eleme
and "The Restaurant" c
from the page.

Validating Your Documents

One way that professional web developers catch errors in their markup is to validate their documents. What does that mean? To validate a document is to check your markup to make sure that you have abided by all the rules of whatever version of HTML you are using (there are more than one, as we'll discuss in Chapter 10, What's Up, HTML5?). Documents that are error-free are said to be valid. It is strongly recommended that you validate your documents, especially for professional sites. Valid documents are more consistent on a variety of browsers, they display more quickly, and they are more accessible.

Right now, browsers don't require documents to be valid (in other words, they'll do their best to display them, errors and all), but any time you stray from the standard you introduce unpredictability in the way the page is displayed or handled by alternative devices.

So how do you make sure your document is valid? You could check it yourself or ask a friend, but humans make mistakes, and you aren't really expected to memorize every minute rule in the specifications. Instead, you use a validator, software that checks your source against the HTML version you specify. These are some of the things validators check for:

- The inclusion of a DOCTYPE declaration. Without it the validator doesn't know which version of HTML or XHTML to validate against.
- An indication of the character encoding for the document.
- The inclusion of required rules and attributes.
- Non-standard elements.
- Mismatched tags.
- Nesting errors.
- Typos and other minor errors.

Developers use a number of helpful tools for checking and correcting errors in HTML documents. The W3C offers a free online validator at *validator.w3.org*. For HTML5 documents, use the online validator located at *html5.validator.nu*. Browser developer tools like the Firebug plug-in for Firefox or the built-in developer tools in Safari and Chrome also have validators so you can check your work on the fly. If you use Dreamweaver to create your sites, there is a validator built into that as well.

Having Problems?

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

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

It could be you didn't save your document before reloading, or you may have saved it in a different directory.

Half my page disappeared.

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 by hand.

I put in a graphic using the `img` element, but all that shows up is a broken image icon.

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 image file is correct. (We'll discuss URLs further in Chapter 6, Adding Links.) Make sure that the image file is actually in the directory you've specified. If the file is there, make sure it is in one of the formats that web browsers can display (GIF, JPEG, or PNG) and that it is named with the proper suffix (*.gif*, *.jpeg* or *.jpg*, or *.png*, respectively).

Test Yourself

Now is a good time to make sure you understand the basics of markup. Use what you've learned in this chapter to answer the following questions. Answers are in Appendix A.

1. What is the difference between a tag and an element?

2. Write out the recommended minimal structure of an HTML5 document.

3. Indicate whether each of these filenames is an acceptable name for a web document by circling "Yes" or "No." If it is not acceptable, provide the reason why.

a. *Sunflower.html*	Yes	No
b. *index.doc*	Yes	No
c. *cooking home page.html*	Yes	No
d. *Song_Lyrics.html*	Yes	No
e. *games/rubix.html*	Yes	No
f. *%whatever.html*	Yes	No

4. All of the following markup examples are incorrect. Describe what is wrong with each one, and then write it correctly.

   a. `<img "birthday.jpg">`

   b. `<i>Congratulations!<i>`

   c. `<a href="file.html">linked text</a href="file.html">`

   d. `<p>This is a new paragraph<\p>`

5. How would you mark up this comment in an HTML document so that it doesn't display in the browser window?

```
product list begins here
```

Element Review: Document Structure

This chapter introduced the elements that establish the structure of the document. The remaining elements introduced in the exercises will be treated in more depth in the following chapters.

Element	**Description**
`body`	Identifies the body of the document that holds the content
`head`	Identifies the head of the document that contains information about the document
`html`	The root element that contains all the other elements
`meta`	Provides information about the document
`title`	Gives the page a title