Chapter 1: Sound HTML Foundations

In This Chapter

- Creating a basic web page
- Understanding the most critical HTML tags
- Setting up your system to work with HTML
- Viewing your pages

This chapter is your introduction to building web pages. Before this slim chapter is finished, you'll have your first page up and running. It's a humble beginning, but the basic web technology you learn here is the foundation of everything happening on the web today.

In this minibook, you discover the modern form of web design using HTML5. Your web pages will be designed from the ground up, which makes them easy to modify and customize. Although you figure out more advanced techniques throughout this book, you'll take the humble pages you discover in this chapter and make them do all kinds of exciting things.

Creating a Basic Page

Here's the great news: The most important web technology you need is also the easiest. You don't need any expensive or complicated software, and you don't need a powerful computer. You probably have everything you need to get started already.

No more talking! Fire up a computer and build a web page!

1. **Open a text editor.**

   You can use any text editor you want, as long as it lets you save files as plain text. If you're using Windows, Notepad is fine for now. If you're using Mac, you'll really need to download a text editor. I like Komodo Edit (`www.activestate.com/komodo-edit`) or TextWrangler (`www.barebones.com/products/textwrangler/`). It's possible to make TextEdit work correctly, but it's probably easier to just download something made for the job. I explain text editors more completely in Chapter 3 of this mini-book.

Don't use a word processor like Microsoft Word or Mac TextEdit. These are powerful tools, but they don't save things in the right format. The way these tools do things like centering text and changing fonts won't work on the web. I promise that you'll figure out how to do all that stuff soon, but a word processing program won't do it correctly. Even the Save as HTML feature doesn't work right. You really need a very simple text editor, and that's it. In Chapter 3 of this minibook, I show you a few more editors that make your life easier. You should not use Word or TextEdit.

2. **Type the following code.**

   Really. Type it in your text editor so you get some experience writing the actual code. I explain very soon what all this means, but type it now to get a feel for it:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset="UTF-8">
<!-- myFirst.html -->

<title>My very first web page!</title>
</head>

<body>

<h1>This is my first web page!</h1>

<p>
This is the first web page I've ever made,
and I'm extremely proud of it.
It is so cool!
</p>

</body>
</html>
```

3. **Save the file as `myFirst.html`.**

   It's important that your filename has no spaces and ends with the `.html` extension. Spaces cause problems on the Internet (which is, of course, where all good pages go to live), and the `.html` extension is how most computers know that this file is an HTML file (which is another name for a web page). It doesn't matter where you save the file, as long as you can find it in the next step.

4. **Open your web browser.**

   The *web browser* is the program used to look at pages. After you post your page on a web server somewhere, your Great Aunt Gertrude can use her web browser to view your page. You also need one (a browser, not a Great Aunt Gertrude) to test your page. For now, use whatever browser you ordinarily use. Most Windows users already have Internet Explorer installed. If you're a Mac user, you probably have Safari. Linux folks generally have Chrome or Firefox. Any of these are fine. In Chapter 3 of this minibook, I explain why you probably need more than one browser and how to configure them for maximum usefulness.

5. **Load your page into the browser.**

   You can do this a number of ways. You can use the browser's File menu to open a local file, or you can simply drag the file from your Desktop (or wherever) to the open browser window.

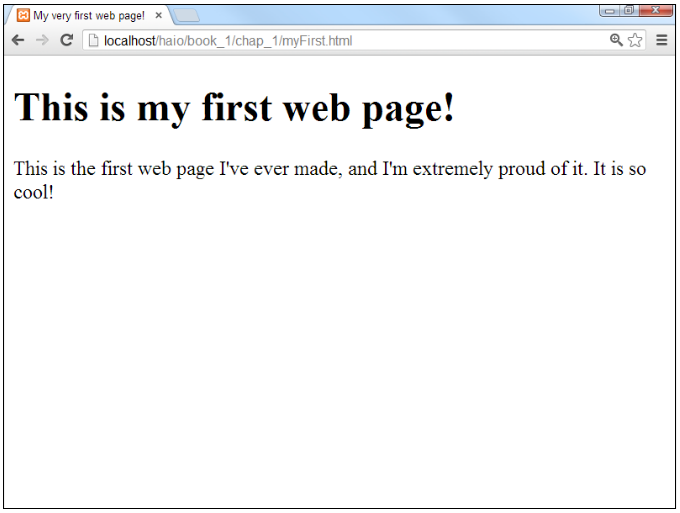
6. **Bask in your newfound genius.**

   Your simple text file is transformed! If all went well, it looks like Figure 1-1.

Understanding the HTML in the Basic Page

The page you created in the previous section uses an extremely simple notation — HTML (HyperText Markup Language), which has been around since the beginning of the web. HTML is a terrific technology for several reasons:

✦ **It uses plain text.** Most document systems (like word processors) use special *binary encoding schemes* that incorporate formatting directly into the computer's internal language, which locks a document into a particular computer or software. That is, a document stored in Word format can't be read without a program that understands Word formatting. HTML gets past this problem by storing everything in plain text.



This is my first web page!

This is the first web page I've ever made, and I'm extremely proud of it. It is so cool!

Figure 1-1: Congratulations! You're now a web developer!

- **It works on all computers.** The main point of HTML is to have a universal format. Any computer should be able to read and write it. The plain-text formatting aids in this.
- **It describes what documents *mean*.** HTML isn't really designed to indicate how a page or its elements look. HTML is about describing the meaning of various elements (more on that very soon). This has some distinct advantages when you figure out how to use HTML properly.
- **It *doesn't* describe how documents *look*.** This one seems strange. Of course, when you look at Figure 1-1, you can see that the appearance of the text on the web page has changed from the way the text looked in your text editor. Formatting a document in HTML does cause the document's appearance to change. That's not the point of HTML, though. You discover in Book II and Book III how to use another powerful technology — *CSS* — to change the appearance of a page after you define its meaning. This separation of meaning from layout is one of the best features of HTML.
- **It's easy to write.** Sure, HTML gets a little more complicated than this first example, but you can easily figure out how to write HTML without any specialized editors. You only have to know a handful of elements, and they're pretty straightforward.
- **It's free.** HTML doesn't cost anything to use, primarily because it isn't owned by anyone. No corporation has control of it (although a couple have tried), and nobody has a patent on it. The fact that this technology is freely available to anyone is a huge advantage.

Meeting Your New Friends, the Tags

The key to writing HTML code is the special text inside angle braces (<>). These special elements are *tags*. They aren't meant to be displayed on the web page, but offer instructions to the web browser about the meaning of the text. The tags are meant to be embedded into each other to indicate the organization of the page. This basic page introduces you to all the major tags you'll encounter. (There are more, but they can wait for a chapter or two.) Each tag has a beginning and an end tag. The end tag is just like the beginning tag, except the end tag has a slash (/):

- **`<!DOCTYPE HTML>`:** This special tag is used to inform the browser that the document type is HTML. This is how the browser knows you'll be writing an HTML5 document. You will sometimes see other values for the doctype, but HTML5 is the way to go these days.
- **`<html lang = "en"></html>`:** The `<html>` tag is the foundation of the entire web page. The tag begins the page. Likewise, `</html>` ends the page. For example, the page begins with `<html>` and ends with `</html>`. The `<html></html>` combination indicates that everything in the page is defined as HTML code. In HTML5, you're expected to tell

the browser which language the page will be written in. Because I write in English, I'm specifying with the code "en."

TIP

Some books teach you to write your HTML tags in uppercase letters. This was once a standard, but it is no longer recommended.

✦ `<head></head>`: These tags define a special part of the web page called the *head* (or sometimes *header*). This part of the web page reminds me of the engine compartment of a car. This is where you put some great stuff later, but it's not where the main document lives. For now, the only thing you'll put in the header is the document's title. Later, you'll add styling information and programming code to make your pages sing and dance.

✦ `<meta charset="UTF-8">`: The meta tag is used to provide a little more information to the browser. This command gives a little more information to the browser, telling it which character set to use. English normally uses a character set called (for obscure reasons) UTF-8. You don't need to worry much about this, but every HTML5 page written in English uses this code.

✦ `<!--/-->`: This tag indicates a *comment,* which is ignored by the browser. However, a comment is used to describe what's going on in a particular part of the code.

✦ `<title></title>`: This tag is used to determine the page's title. The title usually contains ordinary text. Whatever you define as the title will appear in some special ways. Many browsers put the title text in the browser's title bar. Search engines often use the title to describe the page.

Throughout this book, I use the filename of the HTML code as the title. That way, you can match any figure or code listing to the corresponding file on the web site that accompanies this book. Typically, you'll use something more descriptive, but this is a useful technique for a book like this.

WARNING!

It's not quite accurate to say that the title text always shows up in the title bar because a web page is designed to work on lots of different browsers. Sure, the title does show up on most major browsers that way, but what about cellphones and tablets? HTML never legislates what will happen; it only suggests. This may be hard to get used to, but it's a reality. You trade absolute control for widespread capability, which is a good deal.

✦ `<body></body>`: The page's main content is contained within these tags. Most of the HTML code and the stuff the user sees are in the body area. If the header area is the engine compartment, the body is where the passengers go.

✦ `<h1></h1>`: H1 stands for *heading level one.* Any text contained within this markup is treated as a prominent headline. By default, most browsers add special formatting to anything defined as H1, but there's no guarantee. An H1 heading doesn't really specify any particular font or formatting, just the *meaning* of the text as a level one heading. When you find out how to use CSS in Book II, you'll discover that you can make your headline look however you want. In this first minibook, keep all the default layouts for now and make sure you understand that HTML is about semantic meaning, not about layout or design. There are other levels of headings, of course, through `<h6>` where `<h2>` indicates a heading slightly less important than `<h1>`, `<h3>` is less important than `<h2>`, and so on.

Beginners are sometimes tempted to make their first headline an `<h1>` tag and then use an `<h2>` for the second headline and an `<h3>` for the third. That's not how it works. Web pages, like newspapers and books, use different headlines to point out the relative importance of various elements on the page, often varying the point size of the text. You can read more about that in Book II.

✦ **`<p></p>`:** In HTML, `p` stands for the paragraph tag. In your web pages, you should enclose each standard paragraph in a `<p></p>` pair. You might notice that HTML doesn't preserve the carriage returns or white space in your HTML document. That is, if you press Enter in your code to move text to a new line, that new line isn't necessarily preserved in the final web page.

The `<p></p>` structure is one easy way to manage spacing before and after each paragraph in your document.

Some older books recommend using `<p>` without a `</p>` to add space to your documents, similar to pressing the Enter key. This way of thinking could cause you problems later because it doesn't accurately reflect the way web browsers work. Don't think of `<p>` as the carriage return. Instead, think of `<p>` and `</p>` as defining a paragraph. The paragraph model is more powerful because soon enough, you'll figure out how to take any properly defined paragraph and give it yellow letters on a green background with daisies (or whatever else you want). If things are marked properly, they'll be much easier to manipulate later.

A few notes about the basic page

Be proud of this first page. It may be simple, but it's the foundation of greater things to come. Before moving on, take a moment to ponder some important HTML principles shown in this humble page you've created:

- **All tags are lowercase.** Although HTML does allow uppercase tags, modern developers have agreed on lowercase tags in most cases. (`<!DOCTYPE>` is one notable exception to this rule.)
- **Tag pairs are containers, with a beginning and an end.** Tags contain other tags or text.
- **Some elements can be repeated.** There's only one `<html>`, `<title>`, and `<body>` tag per page, but a lot of the other elements (`<h1>` and `<p>`) can be repeated as many times as you like.
- **Carriage returns are ignored.** In the Notepad document, there are a number of carriage returns. The formatting of the original document has no effect on the HTML output. The markup tags indicate how the output looks.

Setting Up Your System

You don't need much to make web pages. Your plain text editor and a web browser are about all you need. Still, some things can make your life easier as a web developer.

Displaying file extensions

The method discussed in this section is mainly for Windows users, but it's a big one. Windows uses the *extension* (the part of the filename after the period) to determine what type of file you're dealing with. This is very important in web development. The files you create are simple text files, but if you store them with the ordinary `.txt` extension, your browser can't read them properly. What's worse, the default Windows setting hides these extensions from you, so you have only the icons to tell you what type of file you're dealing with, which causes all kinds of problems. I recommend you have Windows explicitly describe your file extensions. Here's how to set that up in Windows 7:

1. **Click the Start button.**

   This opens the standard Start menu.

2. **Open the Control Panel.**

   The Control Panel application allows you to modify many parts of your operating system.

3. **Find Appearance and Personalization.**

   This section allows you to modify the visual look and feel of your operating system.

4. **Choose Folder Options.**

   This dialog box lets you modify the way folders look throughout the visual interface.

5. **Find Advanced Settings.**

   Click the View tab and then look under Advanced Settings.

6. **Display file extensions.**

   By default, the Hide Extensions for Known File Types check box is selected. Deselect this check box to display file extensions.

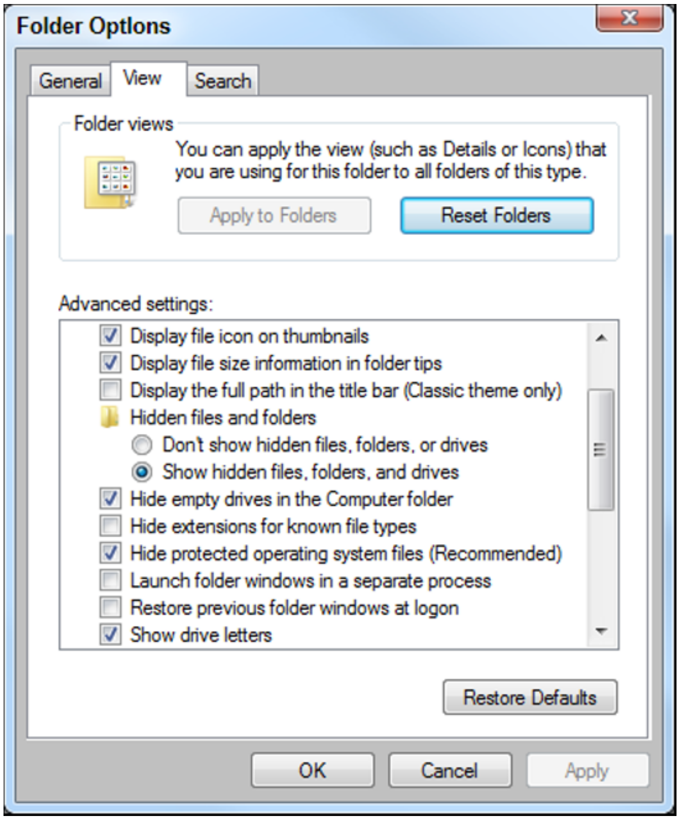
The process for displaying file types is similar in Windows 8:

1. **Go to Windows Explorer.**

   Use the Windows Explorer tile to view Windows Explorer — the standard file manager for Windows.

2. **Click the View tab.**

   This tab allows you to modify how directories look.



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:

- [✓] 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)
- [ ] Launch folder windows in a separate process
- [ ] Restore previous folder windows at logon
- [✓] Show drive letters

Figure 1-2: Don't hide file extensions (deselect that Hide Extensions check box).

3. **De-select filename extensions.**

   If this button is checked, file extensions are shown (which is what you want.) (See Figure 1-2.) Note this is the opposite of Windows 7's behavior.

TIP

Although my demonstration uses Windows 7 and 8, the technique is similar in older versions of Windows. Just do a quick search for "displaying file extensions."

Setting up your software

You'll write a lot of web pages, so it makes sense to set up your system to make that process as easy as possible. I talk a lot more about some software you should use in Chapter 3 of this minibook, but for now, here are a couple of easy suggestions:

✦ **Put a Notepad icon on your Desktop.** You'll edit a lot of text files, so it's helpful to have an icon for Notepad (or whatever other text editor you use) available directly on the Desktop. That way, you can quickly edit any web page by dragging it to the Desktop. When you use more sophisticated editors than Notepad, you'll want links to them, too.

✦ **Get another web browser.** You may just *love* your web browser, and that's fine, but you can't assume that everybody likes the same browser you do. You need to know how other browsers interpret your code. Chrome is an incredibly powerful browser, and it's completely free, as well has having a lot of great programmer's features. If you don't already, I suggest having links to at least two browsers directly on your Desktop.

Understanding the magic

Most of the problems people have with the web are from misunderstandings about how this medium really works. Most people are comfortable with word processors, and we know how to make a document look how we want. Modern applications use WYSIWYG technology, promising that *what you see is what you get.* That's a reasonable promise when it comes to print documents, but it doesn't work that way on the web.

How a web page looks depends on a lot of things that you don't control. The user may read your pages on a smaller or larger screen than you. She may use a different operating system than you. She may have a slower connection or may turn off the graphics for speed. She may be blind and use screen-reader technology to navigate web pages. She may be reading your page on a tablet, smart phone, or even an older (not so smart) cellphone. You can't make a document that looks the same in all these situations.

A good compromise is to make a document that clearly indicates how the information fits together and makes suggestions about the visual design. The user and her browser can determine how much of those suggestions to use.

You get some control of the visual design but never complete control, which is okay because you're trading total control for accessibility. People with devices you've never heard of can visit your page.

Practice a few times until you can easily build a page without looking anything up. Soon enough, you're ready for the next step — building pages like the pros.