1

An Overview of HTML and CSS on the Web

In This Chapter

- Bringing web pages to life
- Understanding the role that HTML plays on web pages
- Appreciating what CSS does to give web pages style
- Exploring and analyzing simple markup examples

Welcome to the wonderful world of the web, HTML, and CSS. With just a little knowledge, some practice, and something to say, you can create your own little virtual acre of cyberspace or improve on existing work.

We use the term HTML throughout this book. Using this term lets us refer to the HyperText Markup Language in general, including both HTML4 and Markup Language), all in one go. Although HTML4 and HTML5 are different (and XHTML differs from both of them, too), they're all enough alike for this reference to make sense.

This book is your down-and-dirty guide to understanding web documents, sprucing up existing web pages, and crafting complex and exciting pages that use intricate designs, multimedia, and scripting.

The best way to start working with HTML is to jump right in, so that's what this chapter does: It explains the basics of how HTML and CSS work behind the scenes inside web pages, and it introduces you to their underlying building blocks. When you're done with this chapter, you'll know how HTML and CSS work so you can start creating or editing web pages right away — albeit very, very simple ones.

How and Where Web Pages Come to Life Online

Web pages can accommodate many kinds of content, such as text, graphics, forms, audio and video files, streaming media, and even interactive games.

Browse the web for only a moment or two, and you see a smorgasbord of information and content displayed in many ways. Every website is different, but all have one thing in common: HyperText Markup Language (also known as HTML). You also run into Cascading Style Sheets (CSS) regularly.

Regardless of what information a web page contains, every page is created using some form of HTML. HTML is the mortar that holds web pages together: graphics, text, and other information are the bricks. CSS tells web pages how they should look (and to some extent, behave) when on display.

HTML files that produce web pages are simple text files, whether those files contain HTML4, HTML5, or even XHTML. Same thing goes for CSS. Reliance on simple text files, or documents, explains why the web works as well as it does. Text is a universal way of representing data for computers. Any text file you create on a Windows PC — including any HTML or CSS file — works equally well on a Mac, Linux/Unix, or any other operating system.

But web pages aren't *merely* text documents. Web pages are made using special, attention-starved, sugar-loaded text called HTML or CSS. Each web page uses its own specific sets of instructions and directives that you include (along with your content) inside text files to specify what's on the page and how that page should look and behave. Stick with us to uncover everything you need to know about HTML and CSS!

HyperText

Special instructions in HTML permit lines of text to point to (that is, *link*) something else in cyberspace. Such pointers are called *hyperlinks.* Hyperlinks are the glue that holds the World Wide Web together. In your web browser, hyperlinks usually appear in blue and are underlined. When you click a hyperlink, it takes you somewhere else.

Hypertext or not, a web page is a text file, which means you can create and edit a web page in any application that creates plain text (such as Notepad or TextEdit). Some software tools offer fancy options (covered in Chapter 23) to help you create web pages, but they generate the same text files you can create using a plain-text editor. We recommend you start with a simple, free web page editor named Aptana Studio. Visit `www.aptana.com`, where you can download the program. (You can also find instructions for Windows, Mac OS, and Linux.)

Steer clear of word processors such as WordPad or Microsoft Word when creating HTML. These tools introduce all kinds of extra markup to web pages that you don't want gunking up your work. If you don't believe us, try creating a web page inside Word and then look at all the stuff it adds inside some other editor. You won't believe your eyes!

The World Wide Web comes by its name honestly. It's literally a web of online pages hosted on web servers around the world, connected in trillions of ways by hyperlinks that tie individual pages together. Without such links, the web would be just a bunch of isolated, stand-alone documents. Boo hoo!

Much of the web's value comes from its ability to link pages and other resources (such as images, downloadable files, and media of all kinds) on a single website, or across many websites. For example, USA.gov (`www.usa.gov`) is a *gateway* website — its primary function is to provide access to other websites. If you aren't sure which government agency handles first-time loans for homebuyers, or you want to take a tour of the Capitol, visit the site shown in Figure 1-1 for information.

Figure 1-1: USA.gov uses hyperlinks to help visitors locate government information.

Web browsers were created specifically for the purpose of reading HTML markup and displaying the resulting web pages such markup describes. Markup lives in a text file (along with your content) to give orders to a browser. For example, look at the web page shown in Figure 1-2. You can

see how the page is made up by examining its underlying HTML; its underlying CSS governs its formatting, layout, and appearance.

This page includes a graphic, a title that describes the page (HTML5 Cafe: Home), a brief welcome, navigation text, and not much else.

Here, different components of the page use different formatting:

- The title for the page appears in its browser tab.
- A brief and simple text navigation bar (HOME | ABOUT US | MENU | CONTACT US) appears at the top border.
- The welcome statement is a text heading in large-format type, followed by a brief description of what's there.
- A coffee cup image appears next, followed by our favorite morning slogan (powered by coffee).

Figure 1-2: This page incorporates multiple parts and numerous bits of HTML and CSS.

The browser knows to display these components of the page in specific ways thanks to the somewhat simplified HTML markup for this page we present in Listing 1-1. Eventually we get around to all the real stuff that's on the actual web page, but for the moment, we present a stick-figure equivalent.

Listing 1-1: The HTML5 Cafe Home Page

```
<!DOCTYPE html>
    <head>
        <meta charset="utf-8">
        <title>HTML5 Cafe: Home</title>
        <meta name="description" content="sample site for 9781118657201">
        <meta name="viewport" content="width=device-width">
        <link rel="stylesheet" href="css/normalize.css">
        <link rel="stylesheet" href="css/main.css">
    </head>
    <body>
        <div id="container">
          <nav id="topnav">
            <a href="index.html">HOME</a> |
            <a href="about.html">ABOUT US</a> |
            <a href="menu.html">MENU</a> |
            <a href="contact.html">CONTACT US</a>
          </nav>
        <div id="content">
          <h1>Welcome to HTML5 Cafe!</h1>
          <p>Here you will find all sorts of delicious HTML5 and CSS3 treats.
             This is the sample site for <a href=
             "http://www.amazon.com/Beginning-HTML5-CSS3-Dummies-
              Computer/dp/1118657209">Beginning HTML5 and CSS3 for Dummies</a>,
              by <a href="http://www.edtittel.com">Ed Tittel</a> and
              <a href="http://www.chrisminnick.com">Chris Minnick</a>. To view
              all of the code samples from the book, visit the
              <a href="menu.html">Menu</a>.
             </p>
          <figure id="home-image">
            <img src="img/pitr_Coffee_cup_icon.png"
             width="400" height="400" alt="delicious coffee">
            <figcaption class="warning">powered by coffee.</figcaption>
           </figure>
         </div>
         <footer>
          copyright &copy; dummieshtml.com
         </footer>
       </div>
    </body>
</html>
```

Nearly all text enclosed between angle brackets (less-than and greater-than signs, or < >) is an HTML *tag* (often called *markup*). For example, the *p* within brackets (tags) identifies text inside paragraphs. The markup between <head> and </head> at the beginning of the document defines data that describes the entire document, including the character set it uses (charset="uft-8"), the title that appears on the browser tab, description and display information, and links to some standard style sheets to manage the look and feel. The markup between <body> and </body> contains everything you can actually see on the page (and some values that control how big the included coffee cup image appears). That's really all there is to it. You embed the markup in a text file, along with text for readers to see, to instruct the browser how to display your web page.

Tags and the content between (and inside) them are also called *elements*. Angle brackets < > enclose HTML markup; curly braces { } enclose CSS markup. (You haven't seen those yet, but they show up in the next chapter.)

Content versus presentation

Simply put, *content* is stuff you can see on a web page. When developers talk about "web page content," they often mean text information that appears on a web page. But images are content, too, as is any of the various types of multimedia that you find on many web pages nowadays, such as music, videos, animations, slide shows, and all kinds of other stuff. In general, HTML handles and packages content on web pages.

Equally simply, presentation is what stuff on a web page looks like when you see it. When web developers talk about "presentation," they're referring to a multitude of characteristics. These include a plethora of typography controls for text (font family, font weight, font size, font color, and much more) but also precise positioning controls that can determine exactly where elements will appear as they're displayed. CSS includes hundreds of presentation controls, which define how web content looks and behaves when it's displayed somewhere, or printed, or even spoken (for those people making use of text-to-speech rendering facilities).

Web browsers

The user's primary tool in the web puzzle is called a *web browser*. Web browsers are programs that read HTML and CSS instructions and then use those instructions to make web page content appear on a screen.

Always write your HTML with the idea that people will view the content using a web browser. Just remember that there's more than one kind of browser out there, and each one comes in several versions.

Usually, web browsers request and display web pages that come from a web server on the Internet. But you can also display HTML pages you've saved on your own device before making them available on an Internet web server. When you develop your own HTML documents (web pages), you view those pages (called *local* pages) in your browser. You can use local pages to get a good idea of what people will see after those pages go live on the Internet.

Each web browser interprets HTML in its own way (though HTML5 is designed to improve this situation). Thus, the same HTML may not look exactly alike from one browser to the next. When you work with basic HTML, variations will be minor, but as you add other elements (such as scripting and multimedia), rendering markup can get hairy. Again, HTML5 is supposed to fix many such problems, but HTML5 isn't completely finished yet as we write this book, so it's still too early to tell whether that promise in theory will be kept in practice.

Chapter 3 explains how to use a web browser to view a local copy of your very first web page, in case you don't already know how to do this.

Some people use text-only web browsers such as Lynx because either:

- They're visually impaired and can't use a graphical display.
- They like a lean, fast web browser that displays only text.

A bevy of browsers

The web may be viewed through browsers of many types, each in its own versions, and each with its own feature sets. Some of the most popular web browsers include Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, and Google Chrome. Other browsers, such as Lynx and Opera, are also widely used. As an HTML developer, you must think beyond your own browser experience and preferences. That's because every user has his or her own personal browser preferences and settings, and they are by no means all alike — not even close!

Each browser renders HTML a bit differently. Every browser handles JavaScript, multimedia, style sheets, and other add-ins differently, too. Throw in different operating systems, and a mix of smartphones and tablets, plus notebook and desktop PCs, and things get really interesting.

Usually differences between browsers are minor. But sometimes a combination of HTML, text, and media can bring a specific browser to its knees. When you work with HTML, test your pages on as many different browsers as you can. Install at least four browsers on your system for testing. We recommend the latest versions of Internet Explorer, Safari, Chrome, and Firefox.

If you want information about more browsers, Yahoo! maintains a fairly complete list (over 60 items altogether):

```
http://dir.yahoo.com/computers_
    and_internet/software/
    internet/world_wide_web/
    browsers
```

Getting to know Internet protocols

Under the hood, the Internet works because of extraordinarily durable and capable sets of rules and formats for networked communication. These things are called protocols, and they define the ways in which computers can talk to each other across the Internet.

In fact the web is made up of billions of resources, each of them linkable. A resource's exact location is the key to linking to it. Without an exact address (a *Uniform Resource Locator,* or URL), you can't use the address bar in a web browser to visit a web page directly.

URLs are the standard addressing system for web resources. Each resource (web page, site, or individual file) has a unique URL. URLs work a lot like your postal address. Figure 1-3 identifies the components of a URL.

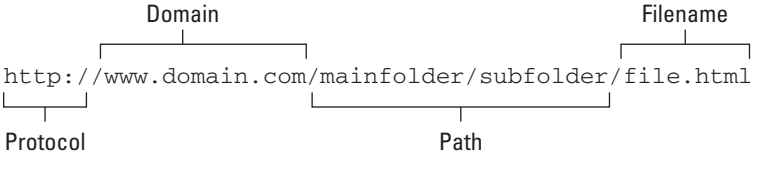

Figure 1-3: The components of a URL help it define an exact location for a file on the web.

The devil is in the protocol details

A collection of related protocols is often called a *protocol suite*. For the Internet, that protocol suite is TCP/IP taken from the abbreviation for the names of two of its most important protocols — namely the Transmission Control Protocol (TCP) and the Internet Protocol (IP). Together, in fact, both TCP and IP transport web communications safely across the Internet. They also support the HyperText Transfer Protocol, also known as HTTP, which is what moves web pages and ancillary materials (images, graphics, media, and so forth) around the Internet.

HTTP isn't the only protocol at work on the Internet riding atop TCP and IP. The Simple Mail Transfer Protocol (SMTP) and the Post Office Protocol (POP) make e-mail possible, and the File Transfer Protocol (FTP) allows you to upload, download, move, copy, and delete files and folders across the Internet. The good news is that web browsers and web servers do all the HTTP work for you, so you need only put your pages on a server or type a URL into a browser.

To see how HTTP works, check out David Gourley and Brian Totty's chapter on HTTP Messages, available through Google Books. Go to `http://books.google.com`, search for *"understanding http transactions,"* double-click *HTTP: The Definitive Guide* in the results, and browse around inside this excellent reference.

Each URL component helps to define the location of a web page or resource:

- **Protocol:** Specifies the protocol the browser should use to request the resource. This is usually HTTP but could be HTTPS (Secure HTTP), FTP, or something else.
- **Domain:** Points to the General website, such as `www.usa.gov`, where the resource resides. A domain may host a few files (like a personal website) or thousands of files (like a large government or corporate site, such as `www.usa.gov` or `www.ibm.com`).
- **Path:** Names the sequence of folders through which to navigate to get to a specific file or resource.

  For example, to get to a file in the `services` folder that resides in the `system` folder, use the `/system/services/` path.
- **Filename:** Specifies which file in a directory path the browser is to access.

Although the URL shown in Figure 1-3 is not publicly accessible, it points to a domain and defines a path that leads to a specific resource named `file.html`:

```
http://www.domain.com/mainfolder/subfolder/file.html
```

Chapter 8 provides the complete details on how to use HTML and URLs to add hyperlinks to your web pages, and Chapter 3 shows how to obtain a URL for your website after you're ready to move it to a web server.

Understanding HTML and Its Versions

You already know that HTML's primary job is to label and accommodate content on web pages. But HTML comes in various versions, each of which handles content, but each of which is slightly different from the other. The basic rules and components stay more or less the same, but some important details differ. The following sections explore those versions and explain what makes them different.

Different versions of HTML

HTML stands for HyperText Markup Language, markup developed in the late 1980s and early 1990s to describe web pages. HTML is now enshrined in numerous standard descriptions called *specifications* from the World Wide Web Consortium (W3C) and the Web HyperText Application Technology Working Group (WHATWG). Work on HTML specifications for versions 1–4 ended in 1999.

When you add an X in front of HTML, you get XHTML, a reworked version of HTML based on the *eXtensible Markup Language* (XML). XML was designed to work and behave well with computers, software, and the Internet.

The original versions (1–4, that is) of HTML included some irregularities that could cause heartburn for software that reads HTML documents. XHTML was designed to use an extremely regular and predictable syntax that's easier for software to handle. XHTML was supposed to replace HTML, but increasing technical complexity in later versions caused it to fall by the wayside. (XHTML 2.0 was so complicated, it was neither widely adopted nor used very much at all.)

In 2004, the WHATWG began work on what is called a "Living Standard" for what is called HTML5 today. It's been in process for a long time now, and the standard is finally nearing completion as we write this book. Some areas of HTML5 are still under development or subject to unresolved controversy. We steer clear of them in this book so that we can provide you with a solid foundation for your web pages for the foreseeable future.

HTML5 already appears to be succeeding where XHTML did not. Even though the standard is still under construction, HTML5 is widely adopted and used on the web today. In fact, the HTML5 specification is in what's called "Candidate Recommendation" form as of December 2012. That's one step before final Recommendation status is reached; most experts expect that final version to be approved and ratified in late 2013 or early 2014.

This book concentrates on the safe parts of HTML5, which use the same kind of regular and straightforward syntax that XHTML offered, but is much simpler to understand and use. Earlier books we've written show how to create both HTML and XHTML; in this book, we stick to HTML5, period.

Creating HTML markup

HTML is a straightforward language for describing web page contents. Its components are easy to use and come in three basic types:

- **Elements:** Identify different parts of an HTML document using tags.
- **Attributes:** Provide additional information about a particular instance of an element.
- **Entities:** Non-ASCII text characters, such as the copyright symbol (©) and accented letters (É). Entities come from the Standard Generic Markup Language (SGML) used to define early HTML versions.

This chapter covers basic form and syntax for elements, attributes, and entities. Parts II through V of this book show how elements and attributes do the following:

- Describe various kinds of text (such as paragraphs, articles, or tables).
- Create effects on a web page (such as changing fonts or colors, or creating buttons with rounded corners and beveled edges).
- Add images and links to a page.

We provide links to some tables of basic entities on our companion website, or you can consult the complete Unicode Character Code Charts at `www.unicode.org/charts`, where you can find codes for nearly every known human language, and a huge collection of abstract shapes and symbols. Find this information by browsing around at this site:

```
www.dummieshtml.com/html5cafe
```

Building HTML documents

Building an HTML document requires assembling a sequence of elements. Some of that sequence is prescribed, which means certain elements always appear in a specific order. Other aspects of the sequence are optional, which gives you the ability to pick and choose the elements for a particular page that are best-suited to accommodate and deliver your content.

Hopefully, this helps to explain why building HTML documents often proceeds from predefined skeletons called *templates.* Because everybody knows in advance what the prescribed HTML elements are and in what order they must appear, there's no reason why work on a web page can't start with such a skeleton — which is more or less content-free when you start work on it anyway. Templates make it a little quicker and easier for you to flesh out your web page (and to make sure you don't forget any of its obligatory elements).

Human error on web pages is inevitable, so web browsers are designed to do their best to compensate for errors and omissions in those pages. But even though HTML5 is simple and straightforward, it comes with certain basic requirements that must be met for a web page to display properly. We get into those requirements in Chapter 2 and Parts II and III of this book. For now, please accept that there are good reasons for following HTML's rules of the road, the best of which is that if you do, your pages should work (and look good) in almost any web browser.

Basically, building a web page consists of inserting a sequence of HTML elements into a document, along with text and pointers to resources, to give the page some content. This means adding elements (and sometimes providing them with attributes and values), writing text, preparing images or media, and so forth. When all the pieces are put together, you can check your work to make sure that the page says what you want, does what you want, and looks the way you want.

Understanding the Role of CSS

Cascading Style Sheets (CSS) manage web page presentation, and govern how pages look and behave when on display to users (or being printed to paper or listened to in a text-to-speech converter). CSS is another markup language that mixes special symbols and keywords to define rules for handling specific HTML elements (and even, specific instances of HTML elements, when "special handling" is needed). CSS is best understood as a tool to manage formatting, layout, and behavior on web pages.

CSS offers an incredible array of presentation controls, including positioning and layout of document elements, identification and assignment of colors for text and backgrounds, and selection and manipulation of specific typefaces, called *fonts,* for textual information. CSS provides methods so that a single page of markup can be presented in different styles for different forms of rendering, so that a document can be tweaked and tuned for delivery on a screen, a printed page, by voice, or even on a Braille-based tactile device.

When an author builds a web page, he or she can define a style sheet for that document. Nevertheless, the reader's web browser can override its definitions with a different style sheet if the reader so chooses. CSS defines a priority scheme, called the *cascade,* that defines which style rule should be applied to individual HTML elements in a document. Such priorities or weights are calculated to apply to style rules so that results are predictable and repeatable.

Different versions here, too . . .

Like HTML, CSS has been around for a while. Today, three finalized versions of CSS have been defined:

- **Version 1, also called CSS Level 1,** was published by the W3C in December, 1996. This version defines all basic CSS capabilities, including font properties such as typeface and emphasis, color for text and backgrounds, text attributes for spacing between letters, words, and lines

of text, alignment values for text, images, and tables, margins, borders, padding and positioning of elements, and unique identifiers and generic classifications for groups of attributes.

- **Version 2 (CSS Level 2),** published in May 1998, adds absolute, relative, and fixed positioning for elements, and a z-index to position multiple layers on a document, along with media types, aural (sound) styles, bidirectional text, and new font properties such as drop shadows.

  Version 2.1 (CSS Level 2 Revision 1, also known as CSS2.1) repaired errors in CSS2. It went through many versions and revisions itself, first reaching Candidate Recommendation status in February 2004. It wasn't finally published as a Recommendation until June 2011.

- **Version 3 (CSS Level 3)** is broken into a collection of items called modules, each of which extends features defined in CSS2. At present, 50 CSS modules have issued from the W3C's CSS Working Group, but only four have attained Recommendation status — namely, Media Queries, Namespaces, Selectors Level 3, and Color. Others are relatively stable and have reached Candidate Recommendation status, including the Backgrounds and Borders module, as well as Multi-Column Layout. The rest are still in various stages of completion.

CSS4 follows the module approach introduced with CSS 3 so there is no single, monolithic CSS4 specification. Only a few Level 4 modules are currently in development, because so few Level 3 modules have been completed. (Image Values, Backgrounds and Borders, and Selectors are the best-known Level 4 modules being worked on at present.)

Most modern web browsers, such as Internet Explorer versions 9 and 10, Chrome versions 20 and later, Firefox version 17 or later, and so forth, fully support CSS2.1. Support for CSS3 varies by module, where all of the Recommendation status items are typically supported, but with varying degrees of support for other modules. CSS4 enjoys little or no support from these same browsers (except for experimental or beta implementation).

As with HTML5, we provide information only about widely adopted and used CSS markup in this book. We assume you want to build workable and predictable web pages and sites. That's why we steer clear of specifications and modules that aren't well understood and widely implemented here.

Creating CSS markup

Interestingly, any HTML document can include style information written using CSS markup. Nevertheless, most web developers isolate CSS markup in separate style sheet documents and use links to those independent external

style sheets in their web pages (HTML documents). This technique helps keep content separate from presentation, encourages reuse of style sheets, and makes it easy to update presentation for multiple pages by editing the style sheets they reference rather than having to incorporate changes into a whole raft of web pages. Another important advantage to this approach is that it encourages use of standard style sheets — like the ones you see referenced in Listing 1-1 in this very chapter, in fact — where local customization comes from reference to local style sheets.

Building a style sheet requires some knowledge of the HTML elements that will appear on your web pages, and it requires you to define properties and values to manage how those elements will look and where they should be positioned on those pages. CSS offers incredible control over presentation, which in turn requires extensive testing and tweaking to get things just right. Furthermore, CSS permits classes or unique identifiers to be used, so you can associate a set of style rules with a single type or even a single instance of an HTML element on your pages. Thus, you can define basic style rules for entire HTML elements and then override them with specific rules for things such as page headers, page footers, certain types of paragraphs, and even individual instances of HTML elements. This provides incredible power over layout, look, and feel on web pages.

Dissecting a Simple Markup Example

Flip back to take another look at Listing 1-1. Careful examination of this short listing shows quite a bit of HTML, but only indirect references to CSS.

Where's the HTML?

The HTML elements you see in Listing 1-1 are as follows, in their order of appearance:

- The `<html>` tag starts the web page, and `</html>` ends it.
- The markup between `<head>` and `</head>` defines general information for the entire web page.
- The text inside the `<title></title>` element provides the page title.
- The `<meta>` element provides information about page content and display layout.
- A `<link>` element establishes a link to an external resource; in this case, to two different CSS style sheets.

- The markup between <body> and </body> supplies actual page content.
- The element defines two different content divisions on the page, one for navigation, the other for page content.
- The navigation <nav></nav> element defines a navigation bar.
- The anchor <a> element defines hypertext links.
- The heading1 <h1></h1> element defines a level-1 heading.
- The paragraph element defines a paragraph of text.
- A figure <figure></figure> element defines a graphic with a caption.
- The image <img> element links to a graphic for display, with horizontal and vertical dimensions and alternative text in case the image doesn't appear.
- A figure caption <figcaption></figcaption> element labels the figure caption.
- A document footer <footer></footer> element defines text for the bottom of the page.

Put all these elements together, add attribute values and text, and you have the web page shown in Figure 1-2.

Where's the CSS?

There is no CSS per se in Listing 1-1. Rather, you find links to two external style sheets, one named `main.css` and the other `normalize.css`. As it happens, these two style sheets are the results of considerable work from the HTML5 community to create standard HTML styling that looks the same (or at least, very close) across multiple browsers. This project is called HTML5 Boilerplate, and it describes itself as "a professional front-end template for building fast, robust, and adaptable web apps or sites." Check it out at `http://html5boilerplate.com`; you can also find a nice showcase of cool examples based on this template at `http://h5bp.net`.

A partnership of equals

It's tempting to treat CSS as an afterthought to HTML or somehow secondary to HTML. You must have content before you can have presentation, right? Although that's true, you can't deliver content without presentation, either, and a good presentation is just as important to the success and usability of a web page as is the content that it handles in a web browser.

That's why it's important to understand both HTML and CSS. You use HTML to control what goes into a web page, and you use CSS to control where and how it appears; what it looks (or sounds, or even feels) like; and how it behaves. Both HTML and CSS are essential to a well-crafted web page, so you should devote equal attention and energy to understanding both HTML and CSS, neither one to the detriment of the other.

'Nuff said!