Description: Introduction to gain understanding of HTML

Length: 2 days / 2 hours

Text book: Sam’s tutorial to html/css and javascript

Difficulty: very easy

**A brief introduction to html:**

Once upon a time, back when there weren’t any footprints on the moon, some farsighted folks decided to see whether they could connect several major computer networks together. I’ll spare you the names and stories (there are plenty of both), but the eventual result was the “mother of all networks,” which we call the Internet.

Until 1990, accessing information through the Internet was a rather technical affair. It was so hard, in fact, that even Ph.D.-holding physicists were often frustrated when trying to swap data. One such physicist, the now-famous (and knighted) Sir Tim Berners-Lee, cooked up a way to easily crossreference text on the Internet through hypertext links.

This wasn’t a new idea, but his simple HTML managed to thrive while more ambitious hypertext projects floundered. *Hypertext* originally meant text stored in electronic form with cross-reference links between pages. It is now a broader term that refers to just about any object (text, images, files, and so on) that can be linked to other objects. *Hypertext Markup Language* is a language for describing how text, graphics, and files containing other information are organized and linked together. By 1993, only 100 or so computers throughout the world were equipped to serve up HTML pages.

Those interlinked pages were dubbed the *World Wide Web (WWW)*, and several web browser programs had been written to allow people to view web pages. Because of the growing popularity of the Web, a few programmers soon wrote web browsers that could view graphical images along with

text. From that point forward, the continued development of web browser software and the

standardization of the HTML—and XHTML—languages has lead us to the world we live in today, one in which more than 110 million web servers answer requests for more than 25 billion text and multimedia files.

These few paragraphs really are a brief history of what has been a remarkable period. Today’s college freshmen have never known a time in which the Web didn’t exist, and the idea of always-on information and ubiquitous computing will shape all aspects of our lives moving forward. Instead of seeing web content creation and management as a set of skills possessed only by a few technically oriented folks (okay, call them geeks if you will), by the end of this book, you will see that these are skills that anyone can master, regardless of inherent geekiness.

**Creating Web Content**

You might have noticed the use of the term *web content* rather than *web pages*—that was intentional. Although we talk of “visiting a web page,” what we really mean is something like “looking at all the text and the images at one address on our computer.” The text that we read, and the images that we see, are rendered by our web browsers, which are given certain instructions found in individual files. Those files contain text that is *marked up*, or surrounded by, HTML codes that tell the browser how to display the text—as a heading, as a paragraph, in a red font, and so on. Some HTML markup tells the browser to display an image or video file rather than plain text, which brings me back to the point:

Different types of content are sent to your web browser, so simply saying *web page* doesn’t begin to cover it. Here we use the term *web content* instead, to cover the full range of text, image, audio, video, and other media found online.

In later chapters, you will learn the basics of linking to or creating the various types of multimedia web content found in websites. All you need to remember at this point is that *you* are in control of the content a user sees when visiting your website. Beginning with the file that contains text to display or codes that tell the server to send a graphic along to the user’s web browser, you have to plan, design, and implement all the pieces that will eventually make up your web presence.

As you will learn throughout this book, it is not a difficult process as long as you understand all the little steps along the way. In its most fundamental form, web content begins with a simple text file containing HTML or XHTML markup. XHTML is another flavor of HTML; the “X” stands for eXtensible, and you will learn more about it as you continue through the chapters. The most important thing to know from the outset is that all the examples in this book are HTML 4 and XHTML compatible, meaning that they will be rendered similarly both now and in the future by any newer generations of web browsers. That is one of the benefits of writing standards-compliant code: You do not have to worry about going back to your code sometime in the future and changing it because it doesn’t work. Your code will likely always work for as long as web browsers adhere to standards (hopefully a long time).

**Understanding Web Content Delivery**

Several processes occur, in many different locations, to eventually produce web content that you cancsee. These processes occur very quickly—on the order of milliseconds—and occur behind the scenes. In other words, although we might think all we are doing is opening a web browser, typing in a web address, and instantaneously seeing the content we requested, technology in the background is working hard on our behalf.

**Chapter 2. Understanding HTML and XHTML Connections**

WHAT YOU’LL LEARN IN THIS CHAPTER:

• How to create a simple web page in HTML

• How to include all the HTML Tags that every web page must have

• How to organize a page with paragraphs and line breaks

• How to organize your content with headings

• How to validate your web content

• How to differentiate between HTML, XML, XHTML, and HTML5

The first chapter gave you a basic idea of the process behind creating web content and viewing it online (or locally, if you do not yet have a web hosting provider). In this chapter, we’ll get down to the business of explaining the various elements that must appear in an HTML file so that it is displayed appropriately in your web browser.

By the end of the chapter, you’ll learn how HTML differs from XHTML and why there are two different languages designed to do the same thing—create web content. In general, this chapter provides a quick summary of HTML and XHTML basics and gives some practical tips to make the most of your time as a web page author and publisher. It’s not all theory, however; you do get to see a real web page and the HTML code behind it.

**Getting Prepared**

Here’s a review of what you need to do before you’re ready to use the rest of this book:

**1.** Get a computer. I used a computer running Ubuntu (Linux) to test the sample web content and capture the figures in this book, but you can use any Windows, Macintosh, or Linux/UNIX machine to create and view your web content.

**2.** Get a connection to the Internet. Whether you have a dial-up, wireless, or broadband connection doesn’t matter for the creation and viewing of your web content, but the faster the connection, the

better for the overall experience. The Internet service provider (ISP), school, or business that

provides your Internet connection can help you with the details of setting it up properly.

Additionally, many public spaces such as coffee shops, bookstores, and libraries offer free wireless Internet service that you can use if you have a laptop computer with Wi-Fi network support.

**3.** Get web browser software. This is the software your computer needs to retrieve and display web content. As you learned in the first chapter, the most popular browser software (in alphabetical order) is Apple Safari, Google Chrome, Microsoft Internet Explorer, Mozilla Firefox, and Opera.

It’s a good idea to install several of these browsers so that you can experiment and make sure that your content looks consistent across them all; you can’t make assumptions about the browsers other people are using.

**Caution**

Although all web browsers process and handle information in the same general way, there are some specific differences among them that result in things not always looking the same in different browsers.

Be sure to check your web pages in multiple browsers to make sure that they look reasonably consistent.

**Getting Started with a Simple Web Page**

In the first chapter, you learned that a web page is just a text file that is marked up by (or surrounded by) HTML codes that tell the browser how to display the text. To create these text files, use a *text editor* such as Notepad (on Windows) or TextEdit (on a Mac)—do not use WordPad, Microsoft Word, or other full-featured word-processing software because those create different sorts of files than the plain-text files we use for web content the Internet 24 hours a day. The same company or school that provides you with Internet access might also provide web space; if not, you might need to pay a hosting provider for the service.

Before you begin working, you should start with some text that you want to put on a web page:

**1.** Find (or write) a few paragraphs of text about yourself, your family, your company, your softball team, or some other subject in which you’re interested.

**2.** Save this text as plain, standard ASCII text. Notepad and most simple text editors always save files as plain text, but if you’re using another program, you might need to choose this file type as an option (after selecting File, Save As).

As you go through this chapter, you will add HTML markup (called *tags*) to the text file, thus making it into web content.

When you save files containing HTML tags, always give them a name ending in .html. This is important: If you forget to type the .html at the end of the filename when you save the file, most text editors will give it some other extension (such as .txt). If that happens, you might not be able to find the file when you try to look at it with a web browser; if you find it, it certainly won’t display properly. In other words, web browsers expect a web page file to have a file extension of .html.

**Caution**

To reiterate, because it is very important both to the outcome and the learning process itself: Do not create your first HTML file with Microsoft Word or any other HTML-compatible word processor; most of these programs attempt to rewrite your HTML for you in strange ways, potentially leaving you totally confused. Additionally, I recommend that you do *not* use a graphical, what-you-see-is-what-you-get (WYSIWYG) editor, such as Microsoft FrontPage or Adobe Dreamweaver. You’ll likely find it easier and more educational to start out with a simple text editor while you’re just learning HTML. You can move to visual tools (such as FrontPage and Dreamweaver) after you have a better understanding of what’s going on under the hood.

When visiting websites, you might also encounter pages with a file extension of .htm, which is also an acceptable file extension to use. You might find other file extensions used on the Web, such as .jsp (Java Server Pages), .asp (Microsoft Active Server Pages), or .php (PHP: Hypertext Preprocessor), but these file types use server-side technologies that are beyond the scope of HTML and the chapters throughout this book. However, these files also contain HTML in addition to the programming language; although the programming code in those files is compiled on the server side and all you would see on the client side is the HTML output, if you were to look at the source files, you would likely see some intricate weaving of programming and markup codes..

Every web page you create must include the <html></html>, <head></head>, <title></title>, and <body></body> tag pairs. In the beginning, Tim created the HyperText Markup Language. The Internet was without form and void, and text was upon the face of the monitor and the Hands of Tim were moving over the face of the keyboard. And Tim said,

Let there be links; and there were links. And Tim saw that the links were good; and Tim separated the links from the text. Tim called the links Anchors, and the text He called Other Stuff. And the whole thing together was the first Web Page.

</p>

</body>

</html>

These coded commands are called *HTML tags* because they “tag” pieces of text and tell the web browser what kind of text it is. This allows the web browser to display the text appropriately.

**Try It Yourself: Creating and Viewing a Basic Web Page**

Before you learn the meaning of the HTML tags used earlier, you might want to see exactly how I went about creating and viewing the document itself. Follow these steps:

**1.** Type all the text in the example, including the HTML tags, in Windows Notepad (or use Macintosh TextEdit or another text editor of your choice).

**2.** Select File, Save As. Be sure to select plain text (or ASCII text) as the file type.

**3.** Name the file **firstpage.html**.

**4.** Choose the folder on your hard drive where you would like to keep your web pages—and remember which folder you choose!

Click the Save or OK button to save the file.

**5.** Now start your favorite web browser. (Leave Notepad running, too, so you can easily switch between viewing and editing your page.)

In Internet Explorer, select File, Open and click Browse. If you’re using Firefox, select File, Open File. Navigate to the appropriate folder and select the firstpage.html file. Some browsers and operating systems will also enable you to drag and drop the firstpage.html file onto the browser window to view it.

*Voilà!* You should see the page. If you have obtained a web hosting account, you could use FTP at this point to transfer the firstpage.html file to the web server. In fact, from this chapter forward, the instructions will assume you have a hosting provider and are comfortable sending files back and forth via FTP; if that is not the case, please review the first chapter before moving on. Or, if you are consciously choosing to work with files locally (without a web host), be prepared to adjust the instructions to suit your particular needs (such as ignoring the commands “transfer the files” and “type in the URL”).

**Note**

You don’t need to be connected to the Internet to view a web page stored on your own computer. By default, your web browser tries to connect to the Internet every time you start it, which makes sense most of the time. However, this can be a hassle if you’re developing pages locally on your hard drive (offline) and you keep getting errors about a page not being found. If you have a full-time web connection via a LAN, cable modem, or DSL, this is a moot point because the browser will never complain about being offline. Otherwise, the appropriate disciplinary action will depend on your breed of browser; check the options under your browser’s Tools menu.

**HTML Tags Every XHTML Web Page Must Have**

The time has come for the secret language of HTML tags to be revealed to you. When you understand this language, you will have creative powers far beyond those of other humans. Don’t tell the other humans, but it’s really pretty easy.

Before you get into the HTML tags, let’s first address the messy-looking code at the top of the previous example The first line indicates that the HTML document is, in fact, an XML document: <?xml version="1.0" encoding="UTF-8"?>

**Note**

It isn’t terribly important that you understand concepts such as character encoding at this point. What is important is that you include the appropriate boilerplate code in your pages so that they adhere to the latest web standards. As of this writing, XHTML 1.1 is a web standard. HTML5 is not yet a web standard, but if you were creating an HTML5 document, these lines at the beginning of your HTML file would not be necessary.

The version of XML is set to 1.0, which is fairly standard, as is the type of character encoding (UTF-8).

The second and third lines of code are even more complicated looking: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

Again, the specifics of this code aren’t terribly important as long as you remember to include the code at the start of your pages. This code identifies the document as being XHTML 1.1, which then allows web browsers to make sure the code meets all the requirements of XHTML 1.1.

Most HTML tags have two parts: an *opening tag*, which indicates where a piece of text begins, and a *closing tag*, which indicates where the piece of text ends. Closing tags start with a / (forward slash) just after the < symbol.

**Note**

The XML/XHTML boilerplate code isn’t strictly required for you to create web pages. You can delete the opening lines of code in the example so that the page starts with the <html> tag and it will still open fine in a web browser. The extra code is included to ensure your pages are up to date with the current web standards. Additionally, the extra code enables you to validate your web pages for accuracy, which you’ll learn how to do a bit later in this chapter.

Another type of tag is the *empty tag*, which is unique in that it doesn’t include a pair of matching opening and closing tags. Instead, an empty tag consists of a single tag that starts with a < and ends with a / just before the > symbol.

Following is a quick summary of these three tags just to make sure you understand the role each plays:

• An *opening tag* is an HTML tag that indicates the start of an HTML command; the text affected bythe command appears after the opening tag. Opening tags always begin with < and end with >, as in

<html>.

• A *closing tag* is an HTML tag that indicates the end of an HTML command; the text affected by the command appears before the closing tag. Closing tags always begin with </ and end with >, as in

</html>.

• An *empty tag* is an HTML tag that issues an HTML command without enclosing any text in the page. Empty tags always begin with < and end with />, as in <br /> and <img />.

For example, the <body> tag in the example tells the web browser where the actual body text of the pagebegins, and </body> indicates where it ends. Everything between the <body> and </body> tags will appear in the main display area of the web browser window, as shown in the example

**Note**

You no doubt noticed in The example that there is some extra code associated with the <html> tag. This code consists of two attributes (xmlns and xml:lang), which are used to specify additional information related to the tag. These two attributes are standard requirements of all XHTML web pages; the former defines the XML namespace, whereas the latter defines the language of the content.

Throughout this book, a standard namespace is defined, and the English language is used. If you are writing in a different language, replace the "en" (for English) with the language identifier relevant to you.

The very top of the browser window shows title text, which is any text that is located between <title> and </title>. The title text is also used to identify the page on the browser’s Bookmarks or Favorites menu, depending on which browser you use. It’s important to provide titles for your pages so that visitors to the page can properly bookmark them for future reference.

You will use the <body> and <title> tag pairs in every HTML page you create because every web page needs a title and body text. You will also use the <html> and <head> tag pairs, which are the other two tags shown in The example. Putting <html> at the very beginning of a document simply indicates that the document is a web page. The </html> at the end indicates that the web page is over.

Within a page, there is a head section and a body section. Each section is identified by <head> and <body> tags. The idea is that information in the head of the page somehow describes the page but isn’t actually displayed by a web browser. Information placed in the body, however, is displayed by a web browser. The <head> tag always appears near the beginning of the HTML code for a page, just after the opening <html> tag.

The <title> tag pair used to identify the title of a page appears within the head of the page, which means it is placed after the opening <head> tag and before the closing </head> tag. In upcoming chapters, you’ll learn about some other advanced header information that can go between <head> and </head>, such as style sheet rules that are used to format the page, as well as the JavaScript you’ll learn to write and embed.

**Tip**

You might find it convenient to create and save a bare-bones page (also known as a *skeleton* page, or *template*) with just the opening and closing <html>, <head>, <title>, and <body> tags, similar to the document used in The example. You can then open that document as a starting point whenever you want to make a new web page and save yourself the trouble of typing all those obligatory tags every time.

The <p> tag used in The example encloses a paragraph of text. You should enclose your chunks of text in the appropriate container tags whenever possible.