The nuts and bolts of HTML

Lesson 2

A few notes about your first web page Page 1 of 8

Before we move on to the nitty-gritty of how HTML works, pull out the 'my first Web page' Web page you created in assignment 2 of lesson 1. There are a few things you should take note of now that will impact your future Web-page-building activities.

- Did you notice that you saved the HTML file as my_first_page.html, with **.html** at the end instead of **.txt** or some other extension? It's very important that you always save your Web pages with an .html extension so that Web browsers know that your documents are Web pages and should be treated as such.
- When you access Web pages saved on your hard drive -- known as local files -- in a Web browser, you don't type in a URL as you do when you're accessing files over the Internet. Instead, you have to browse through your local file system, find the file, and then open it with the browser.
- The document title -- My First Web Page -- described with the **<title> . . . </title>** element doesn't show up in the browser window, but instead is displayed as the browser window title bar. This is the way that all browsers deal with the document title.
- You should keep your text editor and Web browser open at the same time so you can easily switch back and forth between the two of them as you edit and review your work.

Now that you've built your first page and seen a little HTML at work, it's time to dig into the nuts and bolts of how HTML works.

One document, two views

Your single Web page is actually open in two different applications -- the text editor and the Web browser -- when you're building Web pages. You make changes to the page in the text editor and view them in the Web browser.

Say hello to HTML Page 2 of 8

HTML is referred to as a markup language because you use a specific mechanism (language) to describe (mark up) the different pieces of content you want to display on a Web page. You use different markup elements to describe different pieces of content, and markup elements are made up of tags. Nothing drives a point home like an example, so check this one out:

```
<h1>Odi et amo</h1>
<p>Odi et amo: quare id faciam,
   fortasse requires.<br>
   Nescio, sed fieri sentio et excrucior.
</p>
```

This short example, a Latin poem by Catullus, is described with HTML tags. You may not be able to read the Latin, but if you know a bit about HTML, you don't really have to read the Latin to know that the first line, contained within the `<h1>` and `</h1>` element is a first level heading. The second, third, and fourth lines, contained within the `<p> . . . </p>` element, are a paragraph.

Work on your home page while away from home

Don't feel like you have to be tied to a desk as you practice your HTML skills and build your web page. Because you create all of your Web pages on your local system, you can take your HTML work with you on a HP notebook PC.

» HP Compaq nc4010 notebook PC

Web browsers can't read Latin either -- or English, or any other language, for that matter -- so they rely on markup elements to tell them how to display content. Figure 2-1 shows how a Web browser interprets this bit of HTML and content and displays it.

Figure 2-2: HTML markup and Latin content displayed in a Web browser.

So all you do when you build Web pages is use HTML tags to describe your content. Your unique combination of HTML, text, and graphics determines what users will see in their Web browsers. The secret to HTML is simply learning what kind of content each tag describes and how you can combine tags to build Web pages.

Of course, you can't just randomly assign elements to content: there are some basic rules for how Web pages need to be structured. In addition, elements have these nifty companions known as attributes that make them even more effective. This lesson includes all you need to know to get started using elements. Lessons 3 - 7 introduce you to the most basic markup elements and show you how to use them to describe different kinds of content.

Content is king

If there's one thing you should remember when building Web pages, it's that content is king. Without content for users to read and interact with, there would be no reason for HTML or the Web. In the end, Web pages are all about publishing content online. If you don't have something to say, your Web pages will be meaningless.

Of course, the design and layout of your Web pages are important. If you use lime-green italicized text on a bright yellow background, you can't really expect your users to read through your content, no matter how good it is. The look and feel of your pages should revolve around your content, making it easier for people to read and understand. As you build more and more pages you'll want to link them together -- using HTML, of course -- and your page design will help users work their way through your different Web pages.

In the end, everything that you do when creating a Web page should contribute to the effective sharing of information with visitors. You

have something to say, and you use HTML and page design to say it in the best possible way.

HTML includes several different controls for describing content, creating hyperlinks on Web pages, and adding media such as graphics, audio, and animations to Web pages. As you begin building your first few Web pages you will learn all about these conventions. The good news is that they're really easy to learn and you'll be an HTML master before you know it. The different HTML controls you'll be dealing with as you build Web pages include:

- Elements
- Attributes and values
- Entities

Of course, you use these different controls to describe content, so in a real sense every Web page is made up of two components:

- Content
- Controls

Elements are the foundation of HTML and have myriad uses, including:

- Describing the different chunks of content you want to include in your pages
- Embedding graphics, audio clips, animations, and other non-text multimedia elements in a Web page
- Creating hyperlinks by attaching a URL to a bit of text or a graphic in a Web page

The trick to learning HTML is learning which elements do what. You've already seen that the `<title> . . . </title>` element describe a Web page title, while the `<p> . . .</p>` element describes a paragraph. You'll find as you read through the remaining course lessons that each HTML element has a particular and proper use. You'll also find as you write HTML and view your Web pages that each browser interprets each HTML element just a little bit differently.

HTML element syntax

The syntax for building HTML elements is very straightforward and easy to work with, but it's important that you take the time now to learn how to build elements correctly. Web browsers rely on elements to determine how to display your Web pages, and if you make a mistake while building your tags you may get some unexpected results when you look at your page in a browser.

Elements are made up of tags -- like the open `<title>` tag and close `</title>` tag -- that surround the content that the element is describing. The tags are like switches that provide a set of instructions to a Web browser to tell it how to handle your content. For example, when a browser encounters an open bold tag (`<b>`) it knows to display all of the text following the tag in boldface until it reaches the close

The right tool makes elements easier

Over time you'll get sick and tired of typing tags and hunting down errors in the tags you've typed. You'll forget to close tag pairs, mistype tag names, leave out slashes in closing tags, and more. You can save yourself a ton of trouble right from the start by working with a good text editor that is HTML-aware.

The two text editors suggested for use with this

bold tag ().

There are two kinds of HTML elements:

- Elements made up of tag pairs that include open and close tags, such as the paragraph tag:

```
<p> . . . </p>
```

- Elements made up of singleton tags that only include one open tag, such as the horizontal rule tag:

```
<hr>
```

The majority of HTML tags come in pairs; the open tag switches a set of instructions to the browser on, and the close tag switches those instructions off. There are a few singleton tags in HTML that don't act as switches but instead include something in the Web page, such as an image or a horizontal line. Because these tags don't turn a set of instructions on and off, but instead tell the browser to insert something in the page, they only need one tag instead of two.

You build each and every HTML tag according to the same rules:

- Every HTML tag begins with a less-than sign (**<**) and ends with a greater-than sign (**>**).
- Closing tags in tag pairs include a slash (**/**) immediately after the less-than sign (**<**).

Always use a forward slash (the one below the question mark on your keyboard) when building your tags. A few browsers will still display your HTML when you use the wrong kind of slash -- a back slash (\) -- but most will not, and it's never a good idea to take chances.

- The text in an opening tag of a tag pair (such as **<title>**) must match the text in the closing tag of the pair (**</title>**).
- You can build HTML tags in lowercase or in uppercase, or in a combination of both, if you like. However, you'll find that over time you'll want to stick to one particular case or another for building your tags. All of the examples in this course use lowercase tags because future versions of HTML will most likely be case-sensitive and use lowercase tags.

The HTML rulebook -- formally known as the HTML specification -- defines exactly which tags you can use to build your HTML pages. Much as you might like to, you can't go making up your own tags.

As you're starting to see, there's not a whole lot of room for mistakes when you're working with HTML. Your tags are instructions to the browser, and the browser only knows how to follow instructions phrased in certain ways. More often than not, when you are trying to figure out why your Web page display is strange or broken, you can trace the error to mistyped tags or other simple errors in your HTML.

class -- BBEdit and HomeSite -- are both designed specifically to make your Web building activities easier. Both have mechanisms for automatically inserting tags into pages, making sure you have closed open tags, and debugging your HTML.

Although elements are the foundation for HTML, they can't do the job of describing data on their own. Even though you may have several different paragraphs in the same

Keeping track of elements

document, all described with the paragraph tag pair (`<p> . . . </p>`), you may want one or two paragraphs aligned to the right or center instead of to the standard left side of the page. Or you may want to include five different images in your page using the image element (`<img>`). Because each image is different, it has a different file name so each is referenced just a little differently in your Web page.

Attributes are HTML's mechanism for providing additional information specific to one instance of an element. If you have three paragraphs, and you want each to be aligned differently, you use the paragraph element and the `align` attribute, to describe each chunk of text as a paragraph and then set its alignment, as in this code:

```
<p align="left">This is a paragraph aligned to the
  left.</p>

<p align="right">This is a paragraph aligned to the
 right.</p>

<p align="center">This is a paragraph aligned to the
 center.</p>
```

The Web browser uses the combination of the paragraph element and the `align` attribute to determine how the text should be displayed. A browser displays a paragraph in a particular way, usually in a 12- or 14-point font and aligned to the left. Adding the align attribute changes the standard way the paragraph is aligned. Figure 2-2 shows how a Web browser interprets these three paragraph element, each with a different value for the align attribute.

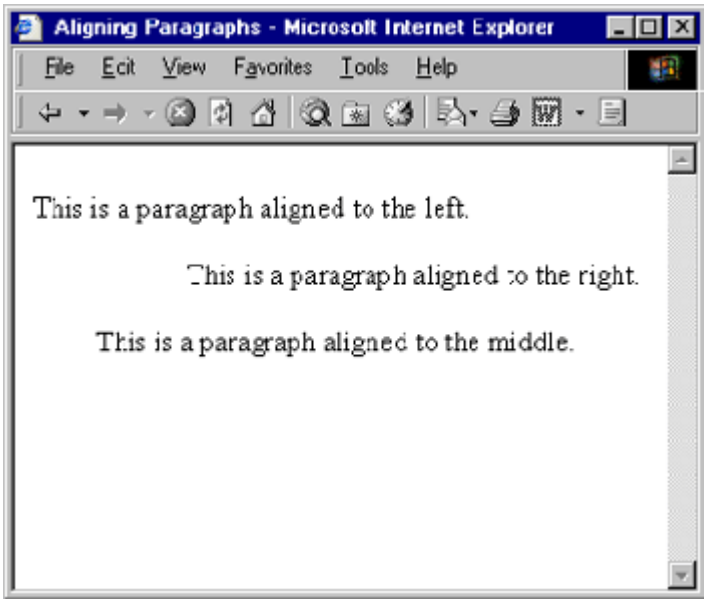

Figure 2-2: The align attribute tells a Web browser how to display text described with the paragraph tag.

Attribute syntax

Attributes are easy to use and are simply an extension of an element. You add attributes to elements according to this basic syntax:

```
<tag attribute="value">
```

and attributes

It can be very difficult to remember all of the HTML tags and their attributes. A good HTML element reference is an invaluable tool. The Web Design Group maintains a great online tag reference that organizes elements alphabetically and in groups by function. The information for each element includes a complete rundown of the attributes you can use with it, the other elements it works with, an explanation of how to use the element, and examples of the element at work. Take a few minutes now to visit the reference and bookmark it.

Notice that the attribute and its value both reside inside of the opening tag's less-than and greater-than signs (< >). You must include the attributes within the tag itself so that the browser will know that the attribute goes with the tag. An HTML-aware text editor is an invaluable tool for helping you work with attributes as well as elements. A good editor has some mechanism for showing you those attributes you can use with an element and will let you know as part of the debugging process if you've incorrectly used an attribute.

Attributes have a variety of uses, including:

- Specifying file locations for graphics
- Identifying URLs for hyperlinks
- Setting the font face, color, and size in which text is displayed

The use of an attribute helps define what kind of value it takes. For example, attributes that point to graphics and identify hyperlinks usually take URLs as their values. Attributes that specify color use a color name or a special code to identify the color. There are also a few attributes that don't take any value at all, but simply work because they are added to an element. As you learn about the different HTML elements in the other lessons in the course, you'll find out which attributes you can use with each element and what kind of value the attribute can take.

Important things to remember about attributes

Attributes add functionality to elements but to work effectively you need to use them correctly. Keep these things in mind as you work with attributes:

- Always place attributes within the opening tag of the element that they modify. If the attribute is not inside of the less-than and greater-than signs, the browser won't know what to do with it.
- The attributes you can use with any given element are predefined. While some elements may share attributes (the paragraph and heading elements both take the align attribute) you cannot mix and match attributes across elements at will. Part of the trick to learning HTML tags is learning which attributes you can use with which elements.
- While a browser won't throw a fit if you don't put quotation marks around all attribute values -- you'll find exceptions to this rule in later lessons -- it's good practice to do so anyway. Future versions of HTML will require that you quote all attribute values, so it's a good habit to get into now.
- Attributes only apply to a single instance of an element. If you want all of your paragraphs to be aligned to the right, you will have to include **align=right** with each and every paragraph element. Yes, there is a better way to control element display, but that's a subject for another class.
- Some elements have required attributes that must be present each and every time you use the element. A good example is the image tag (**<img>**), which requires a source attribute (**src=**) each time the tag appears.

You learn the most about attributes and the effect they have on the way browsers display Web pages by using attributes with your elements. In the upcoming lessons you'll have ample opportunity to work with attributes and see how you can use them to guide the display of your Web page.

You know that the ultimate display of your HTML in a Web browser is driven by the way you combine elements (and their attributes, of course) and content. To kick it up a notch, you can combine multiple HTML elements to more accurately describe your content, as in this bit of code:

```
<p>
  <font face="Arial">
This paragraph will be displayed in Arial.
  </font>
</p>
```

Combining two elements, as the paragraph and font elements are combined in the example, is called nesting elements. One element (the font element) sits inside of, or is nested within, another (the paragraph element). Not only is the text in the example described as a paragraph, but the font tag also tells the browser what font face the paragraph's text should be displayed in. Several of the more complex HTML elements, such as tables, frames, and forms, all take advantage of nesting to describe often complex information using the simple HTML element system.

Whenever you nest elements, there is one important thing that you must always remember: elements must be completely and correctly nested within each other. You should always close the element you opened last, first. Looking back at the example, notice that the paragraph element opens first, followed by the font element. Then the font element closes followed by the paragraph element. The font element is completely nested within the paragraph element.

A handy nesting tip

Think of nesting your elements in the same way you nest suitcases. You can't close the outermost suitcase until the innermost is first closed. While many Web browsers won't care if you don't always nest your elements correctly, some do, and if they do your page will not display correctly.

As you learned in Lesson 1, Web pages are built with HTML, and you create HTML elements and the text that they describe in plain ol' ASCII text. This is a good thing because:

- Any kind of computer, running any kind of operating system, can read plain text files.
- You can use almost any software tool, from Notepad to your favorite souped-up text editor, to write HTML and build Web pages.

If Web pages weren't built with text, they wouldn't be so easily shared across the Internet and among thousands of different computers. But while it is a very good thing that Web pages are written in plain text, there are a few drawbacks:

- You must mix the HTML tags that describe your content in with the content itself. This is a bit alien in our modern world of graphical user interfaces (GUIs) for building documents, such as WordPerfect or Microsoft Word. It can also make it difficult to read and work with Web page content when you're new to HTML and plain text pages.
- ASCII text is limited to a small set of characters. As a result, HTML has to find a way to express all kinds of information -- from copyright symbols to boldface text -- using a limited set of characters. This accounts for the different way that HTML describes content using tags. If you're worried about using non-ASCII characters like copyright symbols, accented letters, and Greek symbols, never fear. The creators of HTML were pretty crafty and they came up with a way to describe non-ASCII characters using nifty little widgets called entities. You'll learn more about these a bit later in the lesson.

In a nutshell, the fact that HTML is written using only ASCII text is a good thing, but you do have to get

used to a new and different set of conventions if you want to build Web pages.

More about ASCII than you ever wanted to know

You may have heard the phrase **ASCII text** bandied about in the world of geeks without really knowing, or caring to know, what it is exactly -- until now. Because HTML is written in ASCII text, it's a good thing to know a bit more about ASCII than you ever really wanted to know before.

To start with, ASCII stands for American Standard Code for Information Interchange. It includes 128 characters from our standard Roman alphabet, both lowercase and uppercase, as well as numbers and a few other control characters like commas, slashes, and less-than and greater-than signs.

The International Organization for Standards (ISO) maintains the ASCII character set, and its official ISO designation is ISO 646. The problem with ASCII (or ISO 646) is that it only supports a limited number of characters, so it doesn't support many languages other than English. To deal with this problem, the ISO created the Latin-1 character set, known as ISO 8859-1, which includes the original 128 characters from ASCII and an additional 128 characters, including accented letters, symbols, and other non-ASCII characters. Of course, you can still only use ASCII -- the first 128 characters of the Latin-1 character set -- to write Web pages, but HTML's entities (discussed later in the lesson) allow you to include non-ASCII Latin 1 and other special characters in your Web pages.

You can only build HTML pages using the 128-character ASCII character set and that's fine and dandy if you're dealing with plain standard English. However, the second you try to add an accented character or a trademark symbol, you're out of luck.

The developers of HTML were well aware that while ASCII text is fine for building HTML pages, it doesn't support the many different characters we use in text communications, and it has absolutely no support for letters outside of the Roman alphabet. Not a good thing, since the Web is worldwide.

As with all other things HTML-related, the solution to the problem is fairly simple. HTML supports these nifty little elements called entities. An entity is simply a collection of ASCII characters that stand for a non-ASCII character. The entity `&copy;` represents the copyright symbol and the entity `&aacute;` represents a lowercase letter "a" with an acute accent (a). Notice that entities begin with an ampersand (&) and end with a semi-colon (`;`). When a browser encounters an ampersand, it starts trying to match the characters that follow with an entity of some sort. If the match is successful, the browser displays the character in place of the entity.

Technically, entities that contain letters are called character entities. You can also represent various non-ASCII characters using a numbered entity as well. Numeric entities begin with an ampersand and a pound sign (&#) and end with a semicolon. For example, © is the numeric entity for the copyright sign. Every character in the Latin-1 character set (discussed on the first page of the lesson) has a numeric entity. Most also have a character entity.

Moving on

And that's it for this lesson. We've covered a lot of ground. By now you should have found your text editor of choice, even if it's just NotePad or SimpleText, and you've even built your first Web page. The assignment for this lesson gives you another chance to get nice and comfortable with working in a text editor and browser, so be sure to complete it before moving on. Don't forget to take a trip to the message board to catch up on important information from the instructor.