Lesson 2: HTML5 Coding

Objectives

By the end of this lesson, you will be able to:

- 2.1.3: Use HTML elements and tags to format paragraphs and text.
- 2.1.7: Add comments to HTML code and document page/site creation.
- 2.1.8: Explain the importance of consistently developing to a single W3C standard.
- 2.7.4: Validate Web page design according to technical and audience standards adopted by employers.
- 2.9.2: Create a Web page using the HTML5 standard.
- 2.12.1: Test and validate Web documents.

Pre-Assessment Questions

1. Which choice represents recommended HTML tag use?

   a. <p>Web page text</p>
   b. <p>Web page text<p>
   c. </p>Web page text</p>
   d. Web page text</p>

2. What HTML element is recommended to create boldface text?

   a. <bold>
   b. <em>
   c. <strong>
   d. <val>

3. How many heading styles (such as H1, H2, etc.) are available when using HTML?

   ____

OBJECTIVE
2.1.8: Using a single W3C standard

Introduction to Using HTML

In this lesson, you will learn how to use standard HTML elements and tags to create functional documents. These documents are commonly referred to as pages when written for the Web or for any other application. The default or starting page of a Web site is called the home page for that site.

NOTE:
Not only can you create a Web document in any text editor, but you can also view the document in any Web browser. You will be viewing the Web pages you create on different Web browsers throughout the course.

This course demonstrates the HTML5 standard. Remember that when you begin creating a Web page or site, you should first choose the W3C HTML standard that you will use. Then stick to your chosen standard, applying it consistently throughout your document, pages and site. This practice ensures that your elements and markup are universal and your pages will render properly in any user agent that your site visitors use.

CIW Online Resources – Movie Clips

Visit CIW Online at http://education.Certification-Partners.com/CIW to watch a movie clip about this topic.

Lesson 2: HTML5 Coding

Elements and Markup Tags

HTML elements are specific components of an HTML document that can provide content and attributes to a Web page. Each element provides meaning to the page, such as identifying the title of the document or specifying where a video or audio file should be placed. These elements are interpreted by Web browsers and other user agents so that the Web page renders properly.

In a nutshell, HTML documents can be described as a collection of elements. Without elements, a Web page would have no structure or formatting.

The majority of HTML elements include a start tag and end tag, also called markup tags. Markup tags enclose elements in angle brackets, or wickets. Please note that the terms "elements" and "tags" are used interchangeably by many Web developers.

Tags embed the element information in the document so that a user agent will render text or other content as instructed by the associated element. For example, the <strong> element is rendered using a start tag, <strong>, and an end tag, </strong>. The text surrounded by the start and end tag is rendered as bold font when viewed in a browser.

For example, bold text identified in an HTML document is written as:

```
<strong> The text between the start and end tag of the strong element appears in
bold when viewed in a Web browser. </strong>
```

The Web browser renders the HTML document text in bold, but the element and its tags do not appear:

Text between the start and end tag of the strong element appears in bold when viewed in a Web browser.

The combination of elements, markup tags and standard text is loosely referred to as either code or markup. Although markup languages are not programming languages, the elements and tags instruct the browser to perform certain actions, and so the use of the term "code" is appropriate in this context.

Container tags and empty tags

NOTE:
It is important to use container tags when writing HTML code.

There are two types of HTML tags:

- **Container tags** — tags that come in pairs. Container tags use starting and ending tags. For example, when you want emphasis (italic) text, you will contain the text between starting and ending <em> </em> tags. These tags are also called the opening and closing tags.
- **Empty tags** — tags that stand alone. Empty tags are those that do not directly format a specified block of text, and therefore one tag can execute the instruction. For example, if you want to create a line break, you insert the <br> tag at the point you want the break to occur.

HTML allows you to use some empty tags, but good coding practice requires you to use container tags. HTML encourages you to use only container tags or empty tags that were meant to be stand-alone tags.

Figure 2-1 demonstrates the proper use of a container tag. The <title> tag contains text between an opening and a closing tag. Note that the closing tag includes the slash (/) character.

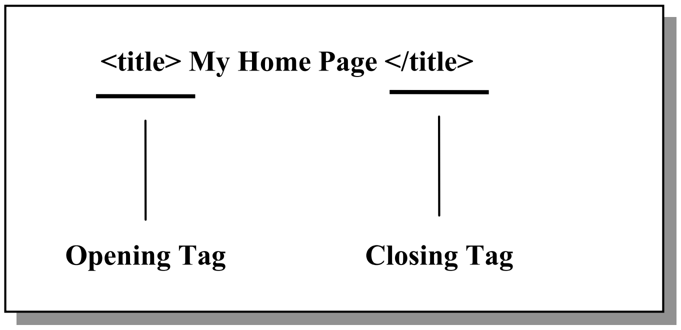


Figure 2-1: <title> tag pair as container for page title text

NOTE:
Be sure you understand where to place the slash (/) character in your tags. The slash appears before the element name for the closing container tags in an HTML tag pair.

Container tags are also known as non-empty tags. You will learn more about creating the appropriate tags throughout the rest of this course.

Tech Note *Empty tags can be written with a slash after the element to become a stand-alone non-empty tag. A good example is the <br> element. It is an empty tag used to create a line break in an HTML document. When coding the <br> tag, it is written as <br/>. Other empty tags are written this way, including the <meta/> and <link/> tags. You will experiment with the <br/>, <meta/>, and <link/> tags later in this lesson.*

What constitutes a tag?

NOTE:
You must be familiar with the functions of HTML5 elements, attributes and values, and be able to identify them in a page of HTML code.

A tag can consist of the following three items inside the angle brackets:

- **An element** — provides the main instruction of the tag. An element is required in every tag. Elements include <body>, <p>, <h1>, <title>, <table> and many others.
- **An attribute** — specifies a quality or describes a certain aspect of the element. For example, a hyperlink is added to a Web page by using the <a>, or anchor, element. The *href* attribute is added, which identifies the hyperlink reference. Many elements require specified attributes, but some do not. An attribute is required in a tag only if the element requires it.

NOTE:
All values should be in quotation marks for HTML.

- **A value** — gives value to the element and its attribute. For example, <a href="http://www.ciwcertified.com"> has a value that instructs the hyperlink to access the CIW Certified Web site. Like attributes, values are optional in a tag unless required by a specified attribute to the element. Values are used only with attributes; elements do not take values directly. Values should be surrounded by quotation marks; they are not required, but placing values in quotation marks is considered good coding practice.

WARNING! *Be sure to close any quotation marks that you open. If you do not close them, the page content up to the next occurrence of the quotation mark character may disappear.*

Some elements, such as <br>, do not use attributes or values. Others, such as the <a> element, are almost always used with attributes and values. It is important that you understand this terminology as you continue throughout the course.

CIW Online Resources – Movie Clips

Visit CIW Online at http://education.Certification-Partners.com/CIW to watch a movie clip about this topic.

Lesson 2: Building a Basic Web Page

OBJECTIVE
2.1.3: HTML text formatting

Document Structure Tags

HTML5 documents usually contain most of the following document structure components:

- **<!DOCTYPE> declaration** — The <!DOCTYPE declaration is the first tag in an HTML document. It informs the interpreter (usually a Web browser) what version of HTML the Web page is written in. Previous to HTML5, the <!DOCTYPE> declaration was an SGML statement and required a fairly complex declaration. In HTML5, however, the tag is written as only <!DOCTYPE html>. The declaration is not case-sensitive, but it is almost always written in uppercase letters by Web developers (it will be uppercase in this course).
- **<html> tag** — The <html> tag is used as a container for the entire HTML document. It nests all code except for the <!DOCTYPE> declaration.
- **<head> tag** — The head section allows you to insert <meta> tags (which describe the nature of the document), links to style sheets, and the <title> tag.
- **<meta> tags** — The <meta> tag can specify various information about the document, known as metadata. This metadata can include a document description, revision dates, and keywords to help search engines index the page. It also specifies the HTML5 character set used, which is usually UTF-8. The <meta> tag is placed within the <head> container tags.
- **<link> tag**— The <link> tag references a style sheet and is recommended for HTML5. A style sheet usually has a .css file name extension and a file name similar to the page to which it is linked (e.g., syb.css for the HTML page named syb.html). Style sheets are often placed in a subdirectory for the Web page. This subdirectory contains all images and associated files for the page. The <link> tag is placed within the <head> container tags.

NOTE:
All of the tags discussed here must be closed.

- **<title> tag** — This tag identifies the document title. Most browsers will display the title in the browse's title bar. The <title> tag is placed within the <head> container tags.
- **<body> tag** — This tag begins the body of the document and includes all the content of the Web page, such as the text, video, hyperlinks and images. The <body> tag is placed after the <head> tag.

OBJECTIVE
2.7.4: Technical and audience standards

If your document fails to include these basic structure elements, it may still validate, given the flexible nature of HTML5. However, the document may or may not render in older browsers that do not support HTML5, such as Internet Explorer versions prior to IE9. You will learn more about backward-compatibility issues later in this course.

NOTE:
Review the basic structural elements carefully. Begin with the <!DOCTYPE> declaration and review each element, including the use of the <meta/> and<link/> tags as stand-alone non-empty tags, similar to the <br/> tag. The difference is that <meta/> and <link/> contain attributes and values. The <html>, <head>, <title> and <body> tags show the standard way to create a container tag.

These basic structural elements are greatly simplified from previous HTML and XHTML versions. The following code displays the basic HTML5 document structure tags:

```
<!DOCTYPE html>
 <html>
 <head>
<meta name="keywords" content="HTML5, basics, elements, tags"/>
 <meta name="author" content="Sampson Avilla"/>
 <meta charset="utf-8"/>
<link rel="stylesheet" type="text/css" href="stylesheet.css"/>
<title>HTML5 Structural Elements</title>
 </head>
 <body>
```

As you learn HTML5, you will start with the structural elements common to most HTML documents. These include:

```
 </body>
 </html>
```

Previous HTML and XHTML documents were different because the <!DOCTYPE> declaration included a Document Type Definition (DTD) file which referenced a specific HTML or XHTML standard. The DTD file is no longer required in HTML5. For example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Are HTML tags case-sensitive?

NOTE:
A key recommendation for HTML is that all elements, attributes and values be written in lowercase letters.

HTML tags are not case-sensitive, but older XHTML tags are case-sensitive. Because XML is case-sensitive in that it requires strict conformance to letter case specified in a given DTD, it was decided that all XHTML document elements and attributes should be developed in lowercase letters to ensure consistency, compatibility and conformance.

The benefit to writing code in lowercase is that the code is now compliant with both HTML and XHTML, and will render in all user agents that follow W3C standards. This practice has been widely adopted for all HTML coding.

Document type declaration (<!DOCTYPE>)

document type declaration (<!DOCTYPE>)
A declaration of document or code type embedded within an HTML, XHTML, XML or SGML document; identifies the version and markup language used. Denoted by the <!DOCTYPE> declaration at the beginning of the document.

The **document type declaration**, or <!DOCTYPE> declaration, describes the markup language and version of your code. It is placed at the very top of your document.

Prior to HTML5, the <!DOCTYPE> declaration was technically not XHTML or HTML; it was actually SGML. It included a reference to the Document Type Definition (DTD) for the markup version used. For instance, the XHTML 1.0 Transitional <!DOCTYPE> declaration contained a reference to *www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd*. If you followed the link, you could read the XHTML 1.0 DTD.

Tech Note

Be careful not to confuse the document type declaration (<!DOCTYPE> declaration) with the Document Type Definition (DTD). The <!DOCTYPE> declaration is a statement that identifies code versions in a document. The DTD is a separate, older document containing a set of rules for structure, syntax and vocabulary, used commonly with XHTML and XML. Previous versions of HTML also included a <!DOCTYPE> declaration that contained a DTD document reference. The DTD applied the rules of the specified language version.

If you do not specify a <!DOCTYPE> declaration, then two problems may arise:

- You may not be able to control how your code renders in the future.
- You will not be able to use a markup validator, because the validator cannot determine the type of markup you are using (e.g., HTML5, XHTML Transitional or HTML 4.01).

Some examples of <!DOCTYPE> declaration statements follow.

HTML 2.0

The following <!DOCTYPE> declaration is used for HTML 2.0 files:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 2.0//EN">
```

HTML 3.2

The following <!DOCTYPE> declaration is used for HTML 3.2 files:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

HTML 4.01

The following <!DOCTYPE> declarations are used for files written in the specified flavors of HTML 4.01 (the Web addresses are optional):

- **HTML 4.01 Transitional**

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
  ```

- **HTML 4.01 Strict**

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
  ```

- **HTML 4.01 Frameset**

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
  ```

XHTML 1.0

XHTML 1.0 approximates the HTML 4.01 <!DOCTYPE> declarations. If you are using the XHTML Transitional flavor and you are not including XML in your document, there will be little difference between an HTML 4.01 and an XHTML 1.0 document. The following

<!DOCTYPE> declarations are used for the specified flavors of XHTML 1.0 (the Web addresses are optional).

- **XHTML Transitional**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

- **XHTML Strict**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- **XHTML Frameset**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

HTML5

HTML5 is not based on SGML, so the traditional <!DOCTYPE> declaration is not required and there is no DTD required. However, it is important to include the <!DOCTYPE> declaration to ensure the browser can identify the version of HTML used in the Web page. The following <!DOCTYPE> declaration s used for HTML5 files:

<!DOCTYPE html>

By using the <!DOCTYPE> declaration, you will improve your page's ability to work with various browsers. The <!DOCTYPE> declaration can help you create a more efficient interface. However, most of its functionality occurs within process.

The <html> tag

The opening <html> and closing </html> tags must encompass all markup for the entire page. The <html> tag can include several attributes, including:

- **manifest** — an attribute used for offline browsing. It lists the address of the HTML document's cache manifest. The manifest attribute requires each page you want cached to include the attribute. This technique is considered more reliable than a traditional browser cache.
- **lang** — configures the page to use a particular language. For instance, a Web document written in English would use <html lang="en"> and a document written in French would use <html lang="fr">. This attribute is helpful for search engines and speech synthesizers. It is a universal attribute that can be used with many different elements besides <html>.
- **xmlns** — If your content needs to conform to XHTML, then specify the XML namespace attribute The default entry is *xmlns="http://www.w3.org/1999/xhtml"*.

The <head> tag

NOTE:
Visit *www.w3schools .com.*

The <head> and </head> tags encompass several page elements, including:

- The <meta> tag.
- The <link> tag that references a CSS file, if present.
- The <title> tag.

The <meta> tag

character set
The group of symbols used to render text on a page.

Unicode
A universal character set designed to support all written languages, as well as scholarly disciplines (e.g., mathematics).

The <meta> tag can specify various information, or metadata, about the document. Attributes of the <meta> tag include the following:

- **charset** — specifies the **character set** used in HTML documents (which is often set by the Web server for HTML documents, rather than by the document itself). It usually specifies the **Unicode** character set, which is standard in today's Web pages:

  ```
  <meta charset="utf-8"/>
  ```

- **name** — values include "keywords," "description" and "author. " This attribute must be accompanied by the *content* attribute. The "keywords" value of the *name* attribute allows you to specify individual words as the value in the accompanying *content* attribute; these words are used by search engines to match pages to search keywords, and to describe the meaning of the document. The "description" value of the *name* attribute allows you to specify entire sentences as the value in the accompanying *content* attribute; these sentences display in search engines to describe the purpose of the document.
- **content** —When paired with the *name* attribute, the *content* attribute values can supply keywords, author name, page descriptions and so forth, as previously described. Following are some examples.

To provide keywords for search engines, you can use the <meta> tag as follows:

```
<meta name="keywords" content="TCP/IP, networking, Java, CIW, certification"/>
```

If you want to use the <meta> tag to provide a detailed description of your page, use syntax similar to the following:

```
<meta name="description" content="You can enter a useful description of the
page here. You can use sentences, as you would in an e-mail or letter, but
keep it concise."/>
```

To specify the author of the Web page, use the <meta> tag with the "author" value:

```
<meta name="author" content="Rosa Estelle Rodriguez"/>
```

An extended discussion of the <meta> tag is beyond the scope of this course. However, the <meta> tag is a very effective back-end tool for ensuring that your pages work well across networks. The <meta> tag was discussed in this section because it is placed within the document structure tags.

The <link> tag

Style sheet references are specified with the <link> tag in the <head> section, usually before the <title> tag. The <link> tag and CSS are recommended for HTML5. The link must point to a CSS file that is simple ASCII text.

The <title> tag

The <title> tag is the first tag that allows you to specify content that will appear on the page. Any text you enclose with this tag appears in the page title box at the top of a browser. This text also appears in the history list and on the page when printed. Title text becomes the Bookmark name if the page is bookmarked or added to a browser Favorites folder.

The <body> tag

All content viewed by a Web browser or other user agent needs to be placed between the <body> and </body> tags. This content includes text, images, video, tables, lists and hyperlinks. The <body> tag no longer has attributes with HTML5. The previous formatting attributes have been replaced by CSS and the inline CSS style attribute.

CIW Online Resources – Online Exercise

Visit CIW Online at *http://education.Certification-Partners.com/CIW* to complete an interactive exercise that will reinforce what you have learned about this topic.

Exercise 2-1: Document Structure Tags

Web Site File Structure

When creating a Web page, you must consider the site's structure. Your HTML and images will be uploaded to a server eventually, so it is always good practice to organize your files as you create them. Figure 2-2 illustrates a typical Web site file structure.

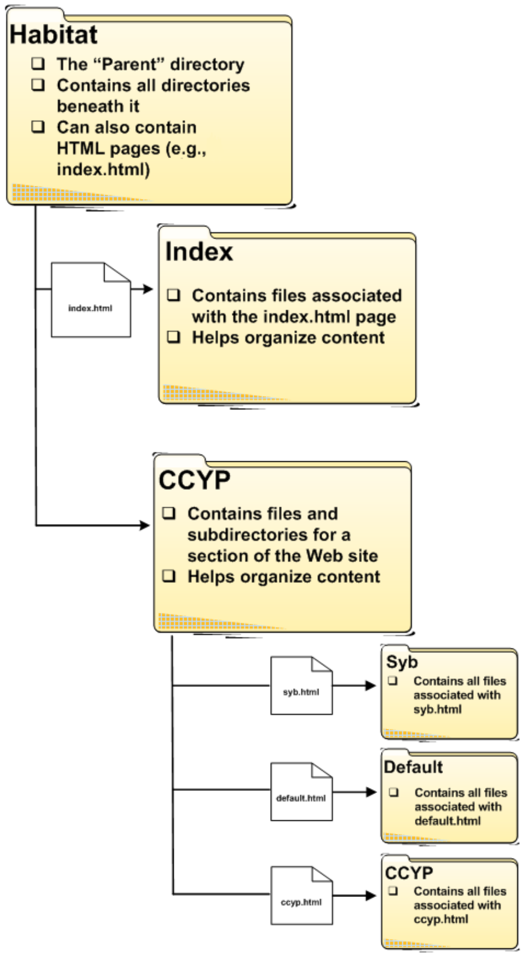

Habitat
The "Parent" directory
Contains all directories beneath it
Can also contain HTML pages (e.g., index.html)
index.html
Index
Contains files associated with the index.html page
Helps organize content
CCYP
Contains files and subdirectories for a section of the Web site
Helps organize content
syb.html
Syb
Contains all files associated with syb.html
default.html
Default
Contains all files associated with default.html
ccyp.html
CCYP
Contains all files associated with ccyp.html

Figure 2-2: Typical Web site file structure

As shown in this figure, the HTML pages are usually placed in a directory, and all images and files used in that page are stored in subfolders with the same name.

In this course, you will work on selected pages from a previously live version of the Habitat for Humanity Web site. All necessary files are provided. Specifically, you will work on a Summer Youth Blitz page, which resides in the Habitat\CCYP\ directory.

CIW Online Resources – Course Mastery

Visit CIW Online at *http://education.Certification-Partners.com/CIW* to take the Course Mastery review of this lesson or lesson segment.

SDA Lesson 2 - Part A

Preparing Your Development Environment

NOTE:
If you do not want to use a simple text editor such as Notepad, you can use an editor such as Bluefish (*http://bluefish.openoffice.nl*).

Before you begin creating Web page code, you should:

- **Obtain a text editor** — Most operating systems have their own editors, so you do not need to download and install one. However, you may prefer to obtain a text editor that automatically numbers lines so you can easily reference your code. It is best to use a text editor that automatically saves standard ASCII text. Applications such as Microsoft Word can save to standard ASCII text only if you explicitly command them to do so. Common text editors include Notepad, WordPad, Vi, Pico and Emacs.
- **Install multiple browsers** — You will need to test your code in multiple environments.
- **Set file preferences** — The Windows operating systems do not show file name extensions by default. You will be working with files with various extensions (e.g., .html, .css, .txt), so you will need to be able to see them. You can set preferences in Windows 7 by selecting Start | Control Panel | Appearance and Personalization | Folder Options, then selecting the View tab. Deselect the Hide Extensions For Known File Types check box so that you can view all file name extensions.

OBJECTIVE
2.1.3: HTML text formatting

In the following lab, you will create a simple HTML page. Suppose you have been assigned to create a basic markup page as a placeholder for a page that will describe a summer youth program for Habitat for Humanity volunteers. This simple file should be named syb.html and should validate as HTML5. What steps would you take to create this simple page?

Lab 2-1: Creating a simple HTML5 page

OBJECTIVE
2.9.2: Creating pages with HTML5

OBJECTIVE
2.12.1: Validating Web documents

In this lab, you will create a basic file directory structure, and you will create an HTML page then validate it as HTML5.

1. If necessary, configure your operating system so that you can read the full extensions of all file names. This will allow you to find your HTML files more easily. In Windows 7, select **Start | Control Panel | Appearance and Personalization | Folder Options**, then select the **View** tab. Deselect the **Hide Extensions For Known File Types** check box. Select the **Show Hidden Files, Folders, And Drives** radio button. Click **OK** to close the Folder Options dialog box. Close the Control Panel.
2. Right-click the **Desktop** and select **New | Folder**.

3. Name the new folder ***Habitat***.

4. Double-click the **Habitat** folder to open it. Inside, create a subfolder and name it ***CCYP***.

5. Double-click the **CCYP** folder to open it.

6. When you are finished, review the directory structure you have created. In Windows Explorer, it should appear off of your Desktop as Habitat\CCYP\.

7. The Habitat\CCYP\ folder will eventually contain the syb.html file, among others. You have now created a standard directory structure for a Web site in which you can organize your images and CSS pages.

8. Right-click inside the **Habitat\CCYP\** folder, and select **New | Text Document**. Name the new text document ***syb.html***. A warning dialog box will appear stating "If you change a file name extension, the file might become unusable. Are you sure you want to change it?" Click **Yes**. You are changing a text file into an HTML document.

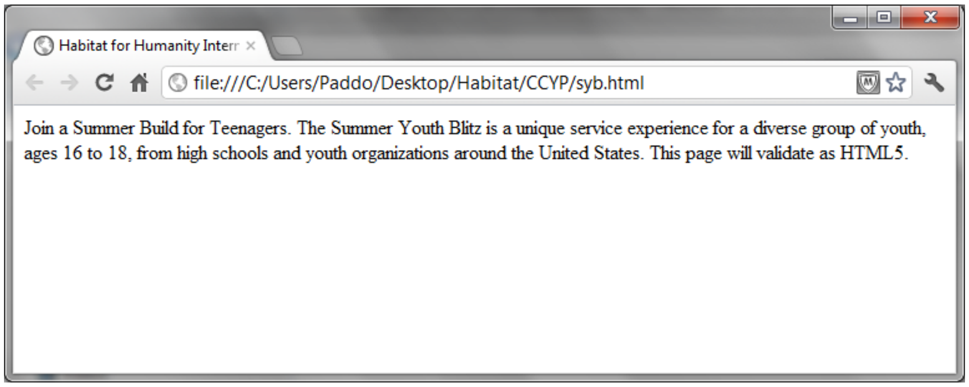
9. Right-click the **syb.html** file and open it in **Notepad**. If Notepad does not appear when you right-click, select **Choose Default Program**. Click the down arrow next to **Other Programs** and double-click **Notepad**. You will see that the syb.html file is currently empty.

10. In the blank syb.html file, enter the following code exactly as written:

```
<!DOCTYPE html>

<html>

<head>
<meta name="keywords" content="CIW, HTML5, Habitat for Humanity"/>
<meta name="description" content="Simple XHTML page for Habitat site"/>
<meta charset="UTF-8"/>
<title>Habitat for Humanity International Summer Youth Blitz Program</title>
</head>

<body style="background-color:white">

Join a Summer Build for Teenagers.
The Summer Youth Blitz is a unique service experience for a diverse group of
youth, ages 16 to 18, from high schools and youth organizations around the
United States. This page will validate as HTML5.

</body>
</html>
```

11. Once you have inserted this code, save your changes. Make sure that your file is named syb.html, not syb.txt or anything else. You may have to close Notepad to rename the file.

12. Now, open **syb.html** in a Web browser by right-clicking the file, selecting **Open With**, and choosing an HTML5-compliant browser. It should resemble Figure 2-3.


Habitat for Humanity Interr
file:///C:/Users/Paddo/Desktop/Habitat/CCYP/syb.html

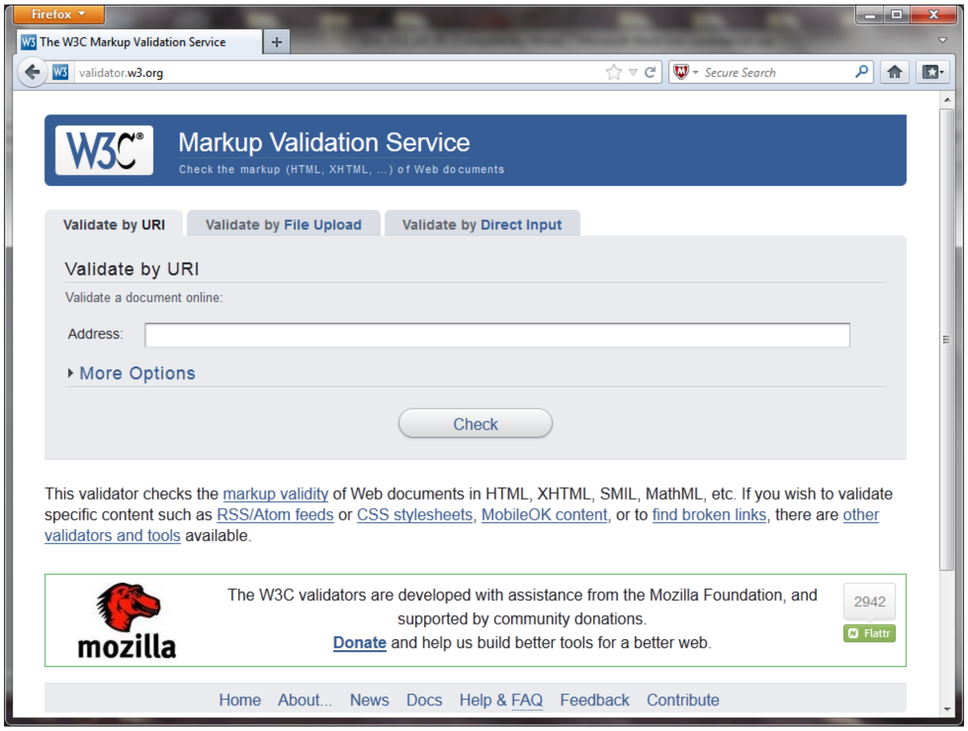
Join a Summer Build for Teenagers. The Summer Youth Blitz is a unique service experience for a diverse group of youth, ages 16 to 18, from high schools and youth organizations around the United States. This page will validate as HTML5.

Figure 2-3: File syb.html in Chrome

Note: You can use any Web browser. In fact, you are encouraged to view code in multiple browsers to ensure that you are creating pages that render well in various environments.

NOTE:
The look and feel of the W3C Markup Validation Service Web site may change over time. However, the functionality will remain similar. If changes occur, make sure that you take the necessary steps to validate your pages to a consistent standard.

13. As you can see, you have created a rudimentary Web page that will validate as HTML5, as long as you have entered the code correctly. To verify this, visit ***http://validator.w3.org***. You will see the W3C Markup Validation Service Web page, as shown in Figure 2-4.



This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available.

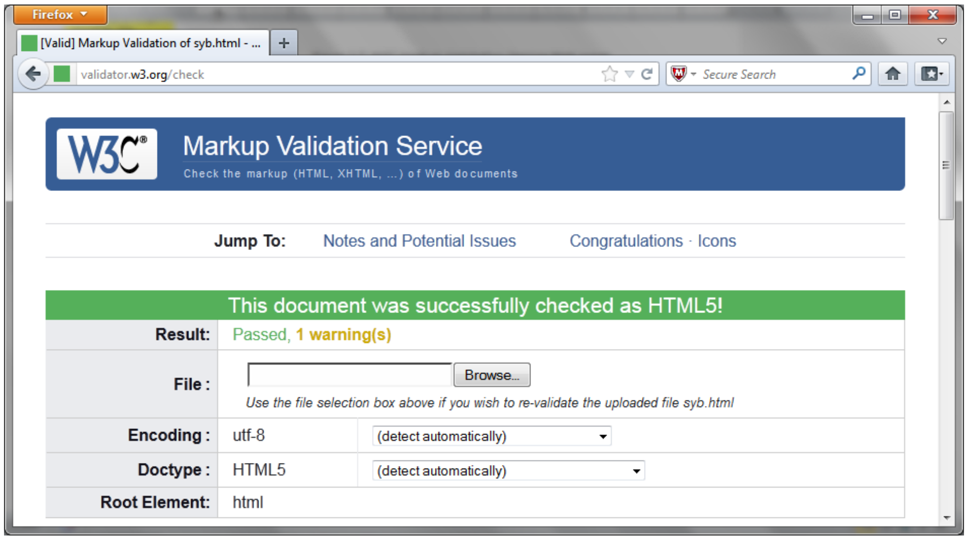
The W3C validators are developed with assistance from the Mozilla Foundation, and supported by community donations. Donate and help us build better tools for a better web.

Figure 2-4: W3C Markup Validation Service Web page

14. Click the **Validate By File Upload** link. In the File text box, click the **Browse** button. Navigate to the **syb.html** file you have created and select it by double-clicking.

15. Click the **Check** button.

16. If your code does not validate, make appropriate changes. Warnings are OK and do not mean you made an error. In Figure 2-5, the warning states that the W3C validator checked the document with an experimental HTML5 conformance checker. The checker may not be up to date with cutting-edge HTML5 code.


Firefox
[Valid] Markup Validation of syb.html - ...
validator.w3.org/check
Secure Search
W3C
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Jump To:
Notes and Potential Issues
Congratulations · Icons
Browse...

This document was successfully checked as HTML5!

Result:	Passed, 1 warning(s)
File :	Use the file selection box above if you wish to re-validate the uploaded file syb.html
Encoding :	utf-8 (detect automatically)
Doctype :	HTML5 (detect automatically)
Root Element:	html

Figure 2-5: Successful HTML5 validation with one warning

17. Once your code validates, change the background color of your page to light blue. To do this, find the <body style="background-color:white">tag, then change the *style* attribute value to read as follows:

```
<body style="background-color:lightblue">
```

18. Save **syb.html**, then open it in your Web browser again.

19. Notice the change in color. Experiment with changing to other background colors. Then validate your code again.

NOTE:
Make sure that you return the background color to white and validate your markup again.

20. Return your Web page to its original state. Change the *style* attribute back to the value "background-color:white" then validate your HTML again.

Tech Note: The background color is typically identified in an external CSS file to which the HTML document is linked. For this lab, you used the inline CSS style attribute, which provides style to an individual file. It overrides any styles identified in the external style sheet. You will learn more about CSS in the next lesson.

21. Close your Web browser.

In this lab, you created and validated an HTML5 document.

Paragraph Formatting and Block-Level Elements

block-level element
A markup element that affects at least an entire paragraph.

text-level element
A markup element that affects single characters or words.

Markup elements that affect an entire paragraph or multiple paragraphs are referred to as **block-level elements**. Elements that can affect something as small as a character or a word are referred to as **text-level elements**. Block-level elements are automatically preceded and followed by paragraph breaks. Text-level elements are not followed by breaks unless the breaks are manually added.

OBJECTIVE
2.1.3: HTML text formatting

Paragraph breaks and line breaks

The most basic block-level element is the paragraph element. The line break element is technically a text-level element, but it is included here in the context of formatting paragraphs. The <p> tag defines the start of a new paragraph, and a closing </p> tag specifies the end of the paragraph.

The <br> tag inserts a simple line break into the document. Because the <br> tag usually breaks a line of text, it never spans words or multiple lines of text, as does the <p> tag.

The <br> tag does not use a separate closing tag, so it follows a unique tag syntax. A closing slash is appended to every line break tag to make it a stand-alone non-empty tag, as follows: <br/>

In the following lab, you will use HTML tags to insert paragraph and line breaks. Suppose the marketing and legal departments have created several paragraphs of text for the syb.html page, and you are assigned to insert this text. You can use the <p> and <br> tags to format information on the page and make it easy to read.

Lab 2-2: Creating paragraph breaks and line breaks with HTML

NOTE:
Switch browsers after completing several labs. Notice that you can view your files on any browser you choose.

In this lab, you will use the <p> and <br> tags to add paragraph breaks and line breaks to a Web page.

1. **Editor:** Open the file **syb.html**, which you edited in the previous lab.

2. **Editor:** Delete all text located between the <body></body> tags except for the "Join a Summer Build for Teenagers" line. Place your cursor on a new line just below "Join a Summer Build for Teenagers."

3. **Editor:** Minimize **syb.html**.

4. Navigate to the **C:\CIW\Site_Dev\Lab Files\Lesson02\Lab_2-2\** directory. Copy the file **syb.txt** to your Desktop, then open it in the text editor. Notice that the text is organized into five separate paragraphs.

5. Copy the contents of the syb.txt file and paste it into **syb.html** (between the <body></body> tags except for the "Join a Summer Build for Teenagers" line). Save the **syb.html** file.

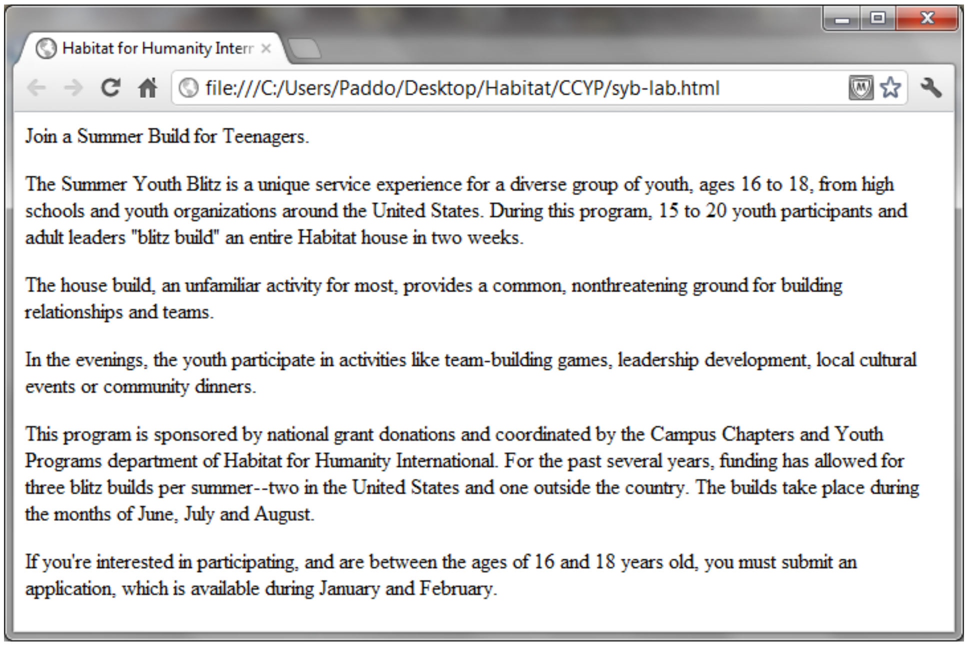
6. Load **syb.html** into a browser. You will now see more text on the Web page, but it is not organized into paragraphs when you view it in a browser.

7. Edit the text in syb.html to add paragraphs breaks that will be recognized by a browser. Add the following code shown in bold:

```
Join a Summer Build for Teenagers.
<p>
The Summer Youth Blitz is a unique service experience for a diverse group of
youth, ages 16 to 18, from high schools and youth organizations around the
United States. During this program, 15 to 20 youth participants and adult
leaders "blitz build" an entire Habitat house in two weeks.
</p>
<p>
The house build, an unfamiliar activity for most, provides a common,
nonthreatening ground for building relationships and teams.
</p>
```

```
<p>
In the evenings, the youth participate in activities like team-building games,
leadership development, local cultural events or community dinners.
</p>
<p>
This program is sponsored by national grant donations and coordinated by the
Campus Chapters and Youth Programs department of Habitat for Humanity
International. For the past several years, funding has allowed for three blitz
builds per summer--two in the United States and one outside the country. The
builds take place during the months of June, July and August.
</p>
<p>
If you're interested in participating, and are between the ages of 16 and 18
years old, you must submit an application, which is available during January
and February.
</p>
```

8. Load **syb.html** into your browser. You will see that the text is now separated into paragraphs, as shown in Figure 2-6.


Habitat for Humanity Interr
file:///C:/Users/Paddo/Desktop/Habitat/CCYP/syb-lab.html

Join a Summer Build for Teenagers.

The Summer Youth Blitz is a unique service experience for a diverse group of youth, ages 16 to 18, from high schools and youth organizations around the United States. During this program, 15 to 20 youth participants and adult leaders "blitz build" an entire Habitat house in two weeks.

The house build, an unfamiliar activity for most, provides a common, nonthreatening ground for building relationships and teams.

In the evenings, the youth participate in activities like team-building games, leadership development, local cultural events or community dinners.

This program is sponsored by national grant donations and coordinated by the Campus Chapters and Youth Programs department of Habitat for Humanity International. For the past several years, funding has allowed for three blitz builds per summer--two in the United States and one outside the country. The builds take place during the months of June, July and August.

If you're interested in participating, and are between the ages of 16 and 18 years old, you must submit an application, which is available during January and February.

Figure 2-6: File syb.html after adding <p> tags

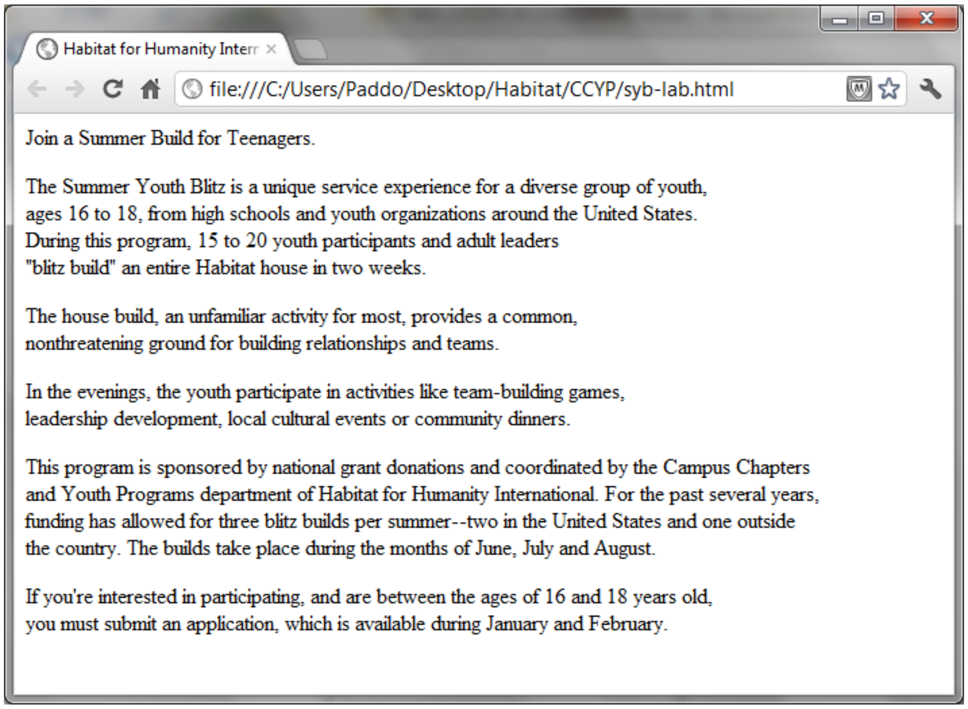
9. Notice that using the <p> tags has created paragraphs. Now, add some <br/> tags to see the difference between a line break and a paragraph break in HTML. Open the **syb.html** file again and enter the following <br/> tags as indicated in bold:

```
Join a Summer Build for Teenagers.
<p>
The Summer Youth Blitz is a unique service experience for a diverse group of
youth,<br/>ages 16 to 18,
from high schools and youth organizations around the United States.
<br/>During this program, 15 to 20
youth participants and adult leaders <br/>"blitz build" an entire Habitat
house in two weeks.
</p>
<p>
The house build, an unfamiliar activity for most, provides a common,
<br/>nonthreatening ground for building
relationships and teams.
</p>
```

```
<p>
In the evenings, the youth participate in activities like team-building games,
<br/>leadership development,
local cultural events or community dinners.
</p>
<p>
This program is sponsored by national grant donations and coordinated by the
Campus Chapters<br/> and Youth
Programs department of Habitat for Humanity International. For the past
several years, <br/>funding has allowed
for three blitz builds per summer--two in the United States and one outside
<br/>the country. The builds take
place during the months of June, July and August.
</p>
<p>
If you're interested in participating, and are between the ages of 16 and 18
years old, <br/>you must submit
an application, which is available during January and February.
</p>
```

10. Notice that you entered the stand-alone non-empty <br/> tag, rather than encompassing text between <br> and </br>.

11. Reload your file in the browser. You should now see the lines break across the page, as shown in Figure 2-7. Notice that although the lines break, no extra returns are added after the line breaks, as they are with the paragraph breaks.


Habitat for Humanity Interr
file:///C:/Users/Paddo/Desktop/Habitat/CCYP/syb-lab.html

Join a Summer Build for Teenagers.

The Summer Youth Blitz is a unique service experience for a diverse group of youth,
ages 16 to 18, from high schools and youth organizations around the United States.
During this program, 15 to 20 youth participants and adult leaders
"blitz build" an entire Habitat house in two weeks.

The house build, an unfamiliar activity for most, provides a common,
nonthreatening ground for building relationships and teams.

In the evenings, the youth participate in activities like team-building games,
leadership development, local cultural events or community dinners.

This program is sponsored by national grant donations and coordinated by the Campus Chapters
and Youth Programs department of Habitat for Humanity International. For the past several years,
funding has allowed for three blitz builds per summer--two in the United States and one outside
the country. The builds take place during the months of June, July and August.

If you're interested in participating, and are between the ages of 16 and 18 years old,
you must submit an application, which is available during January and February.

Figure 2-7: File syb.html after adding <br/> tags to create line breaks

12. Close all browser and editor windows.

In this lab, you used the <p> and <br> tags.

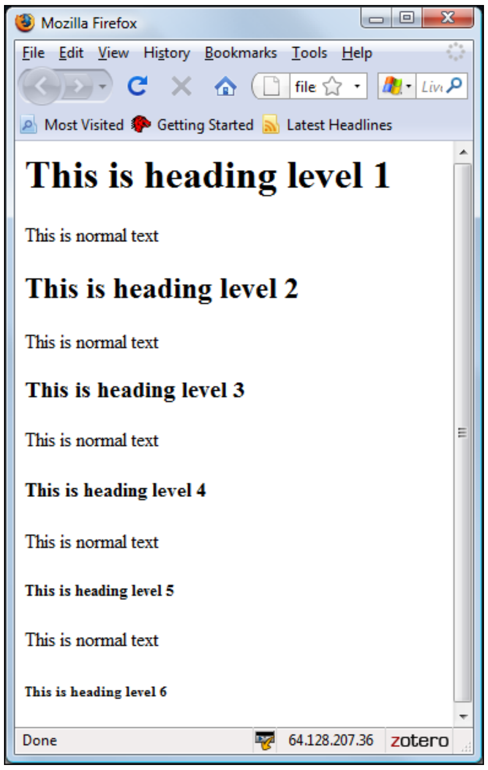
The preceding lab demonstrates that the appearance of text in the editor will not necessarily match the appearance of text in the browser. Do not become frustrated when the text in your browser does not appear as you intended. Determine what needs to be done to achieve the desired appearance, and add the appropriate code to your file.

OBJECTIVE
2.9.2: Creating pages with HTML5

Heading levels

Even the most basic documents will usually include at least one heading, and more likely several. Denoting text as heading elements emphasizes the start of different sections on your page and draws attention to that text. Heading tags have built-in styles associated with them. For example, text formatted as a heading level 1 element is rendered by default in a large, bold, serif font.

HTML uses six heading styles. Heading tags are container tags that encompass the affected text. The <h1> and </h1> tags cause enclosed text to be rendered in the heading level 1 style; the <h4> and </h4> tags cause enclosed text to be rendered in the heading level 4 style, and so forth. The largest heading is level 1. Heading level 4 text is rendered the same size as normal text. Heading levels 5 and 6 are smaller than normal text and should be used sparingly, if at all. Figure 2-8 shows heading appearances relative to normal text.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file
Most Visited Getting Started Latest Headlines
Done 64.128.207.36 zotero

This is heading level 1

This is normal text

This is heading level 2

This is normal text

This is heading level 3

This is normal text

This is heading level 4

This is normal text

This is heading level 5

This is normal text

This is heading level 6

Figure 2-8: Heading-level text and normal text

Because headings are block-level elements, they are automatically preceded and followed by paragraph breaks, regardless of the relative position of the element to other text in the source code. It is important to note that you cannot place any header elements within a set of <p> </p> tags. If you do, your code will not validate as HTML5 and it may not render properly.

Tag nesting in markup

NOTE:
Make sure that you understand the concept of tag nesting.

You will often use multiple sets of tags to format some text. Placing a pair of tags within another pair of tags is called tag nesting. You must ensure that your code is properly nested. Proper nesting means that you must open and close a pair of tags within another pair. The following two examples show both proper and improper tag nesting techniques.

Proper: `<h1> <em> ... </em> </h1>`

Improper: `<h1> <em> ... </h1> </em>`

Notice that in the improper example, the <em> tag (for italic text) is opened within the <h1> tag, but then closed outside the </h1> tag. If you fail to properly nest code, your pages may still render in some user agents, but they will not validate and may fail to render in the future.

Similarly, heading tags (e.g., <h1>, <h2>) should not be used within formatting tags (e.g., <em>, <p>). This tag combination constitutes improper nesting because the elements are incompatible, and will therefore prevent your code from validating.

In the following lab, you will use HTML to create headings. Suppose your project manager has assigned you to add headings to a Web page so that readers immediately understand the page's topics. The headings must help organize the content. The page content includes an introduction, the program's sponsors and an application section.

OBJECTIVE
2.1.3: HTML text formatting

Lab 2-3: Using headings in HTML

NOTE:
Example code for this lab is provided in the supplemental files.

In this lab, you will add heading tags to Web page code to help organize the content.

1. **Editor:** Open **syb.html**.

2. Add and edit the code as shown in bold so that it has <h1> tags but no period at the end:

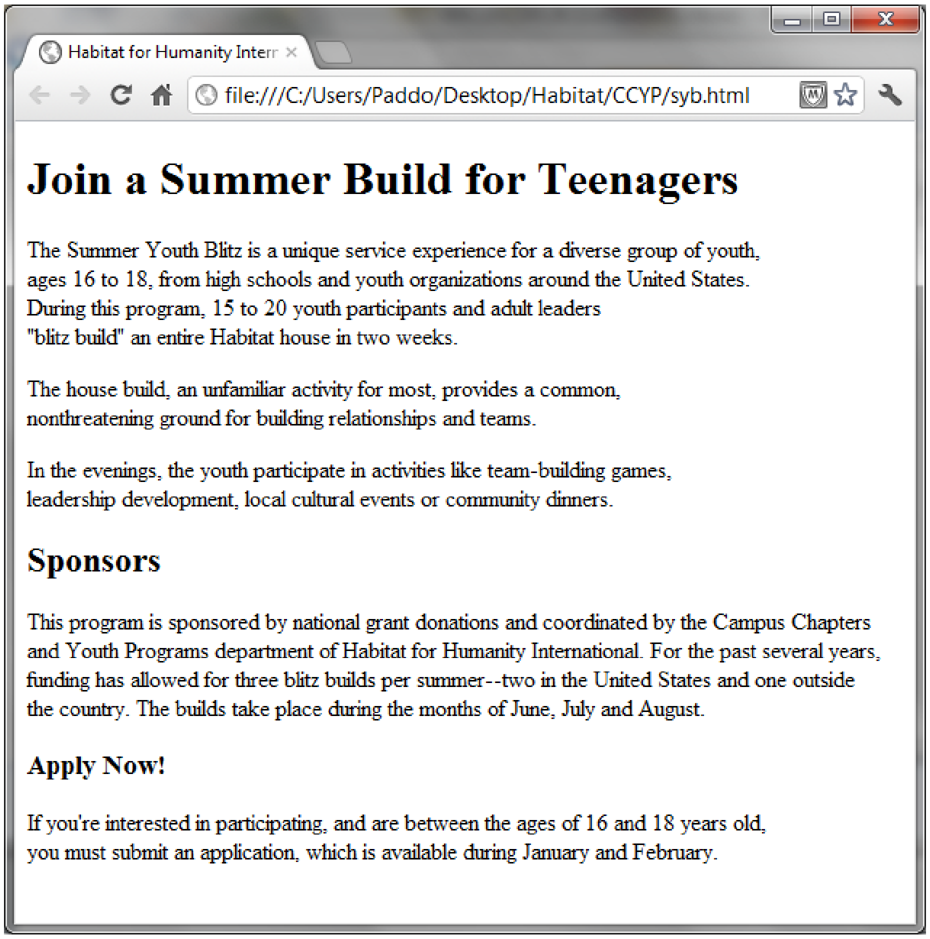
```
<h1>Join a Summer Build for Teenagers</h1>
```

Note: Make sure that the <h1> line is not placed between a set of <p>...</p> tags.

3. **Editor:** Add a line that says ***Sponsors*** just above the fourth paragraph, and make it an h2 heading. Again, make sure that your heading is not between a set of <p>...</p> tags.

4. **Editor:** Add a line that says ***Apply Now!*** just above the last (fifth) paragraph, and make it an h3 heading. Again, make sure that your heading is not between a set of <p>...</p> tags.

5. **Editor:** Save your changes and load the file into a browser. Your screen should resemble Figure 2-9. When you are finished, validate your code.


Habitat for Humanity Interr
file:///C:/Users/Paddo/Desktop/Habitat/CCYP/syb.html

Join a Summer Build for Teenagers

The Summer Youth Blitz is a unique service experience for a diverse group of youth,
ages 16 to 18, from high schools and youth organizations around the United States.
During this program, 15 to 20 youth participants and adult leaders
"blitz build" an entire Habitat house in two weeks.

The house build, an unfamiliar activity for most, provides a common,
nonthreatening ground for building relationships and teams.

In the evenings, the youth participate in activities like team-building games,
leadership development, local cultural events or community dinners.

Sponsors

This program is sponsored by national grant donations and coordinated by the Campus Chapters and Youth Programs department of Habitat for Humanity International. For the past several years, funding has allowed for three blitz builds per summer--two in the United States and one outside the country. The builds take place during the months of June, July and August.

Apply Now!

If you're interested in participating, and are between the ages of 16 and 18 years old,
you must submit an application, which is available during January and February.

Figure 2-9: File syb.html after adding heading, paragraph and line break tags

6. Close all browser and editor windows.

In this lab, you used markup headings.

OBJECTIVE
2.9.2: Creating pages with HTML5

fixed-width font
A font in which every character, including the space character, has equal width. In proportional-width fonts, letters such as I and J have less width than M or B.

Primitive formatting with the <pre> tag

Sometimes, you may want to use text that has already been formatted in a table or with a **fixed-width font**, such as Courier or Lucida Sans Typewriter. With the preformatted text tag (<pre>), all line breaks and spacing will be displayed in the browser exactly as they appear in the original text. The text will display in a fixed-width font, usually Courier.

The <pre> tag allows you to display plaintext files in their original format. It is commonly used to display tabular data. The <pre> tag is a container tag, requiring a closing </pre> tag.

Tech Note *To learn more about the <pre> tag, visit www.w3schools.com/html5/tag_pre.asp*

As you will learn in a later lesson, HTML tables are more attractive and functional for presenting tabular data. However, if you have preformatted data and little time, the <pre> tag is quick and simple to use.

OBJECTIVE
2.9.2: Creating pages with HTML5

NOTE:
You should use the style attribute with the "text-align:center" value to center text, as opposed to the deprecated <center> tag.

Indenting and centering text

When you want to center a paragraph of text, you must use an inline CSS *style* attribute in the paragraph tag. The syntax is as follows:

```
<p style="text-align:center"> This text is centered. </p>
```

In this example, the <p> tag encompasses the text you want to format. The *style* attribute tells the browser that the paragraph text should be aligned to the specified value, "center".

You can use the *style* attribute for many formatting functions. Most often, you will use it to format content in an individual HTML file when you are not using CSS or when you want to override the external CSS file. For example, you can center text, tables and images. You can also use the *style* attribute to justify items to the right or left on a page. For example, consider the following code:

```
<p style="text-align:right"> This text is aligned to the right. </p>
```

In this example, the text would render on the right side of the page.

The <blockquote> tag indents a block of text. As its name suggests, it is often used to format quotes within a Web page. Do not use <blockquote> tags within <p> tags, and do not use <h1> tags within <blockquote> tags. Doing so will prevent your code from validating.

In the following lab, you will use HTML to center and indent text on a Web page. Suppose your project manager has seen your HTML work thus far and asked you to continue formatting the document. She has suggested indenting and centering some text to enhance the page's appearance. She has also asked you to add some contact information at the bottom of the page.

Lab 2-4: Indenting and centering text with HTML

OBJECTIVE
2.1.3: HTML text formatting

In this lab, you will use HTML to indent and center text in the file you created in previous labs.

1. **Editor:** Open **syb.html** and scroll to the bottom of the file.

2. **Editor:** Add the following line just above the </body> tag:

```
<p style="text-align:center"> For more information, contact us at (800) 422-
4828, ext. 2220. </p>
```

NOTE:
You must properly nest code, or else it may fail validation, render incorrectly, or both.

3. **Browser:** Load the **syb.html** file. You will see that your new line is centered. Validate your code at ***http://validator.w3.org***.

   Note: Properly nest code, or else it may fail validation, render incorrectly, or both.

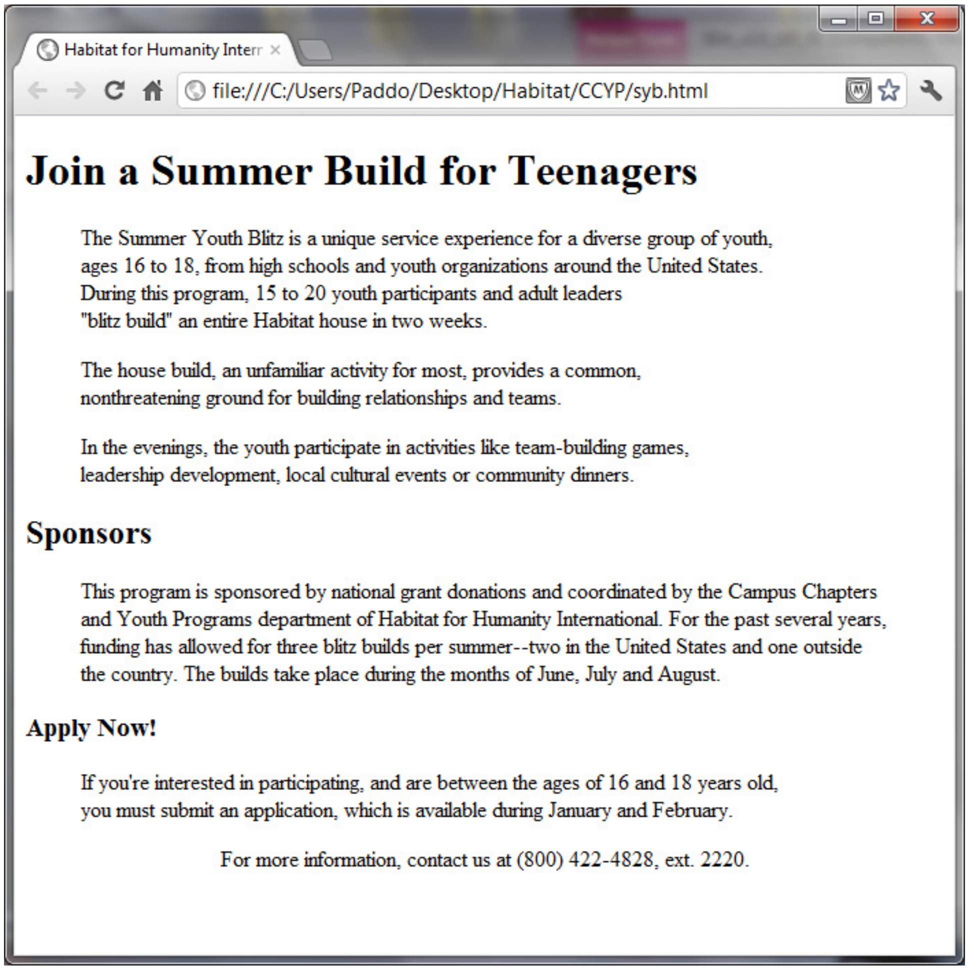
4. Use the **<blockquote>** tag to indent all of the text beneath each header, as follows:

```
<h1>Join a Summer Build for Teenagers</h1>
<blockquote>
<p>
The Summer Youth Blitz is a unique service experience for a diverse group of
youth, <br/>ages 16 to 18,
from high schools and youth organizations around the United States.
```

```
<br/>During this program, 15 to 20
youth participants and adult leaders <br/>"blitz build" an entire Habitat
house in two weeks.
</p>
</blockquote>
```

NOTE:
Resizing the browser window repositions the last line of text to appear centered relative to the rest of the document.

5. Repeat this formatting for the entire page by enclosing each text paragraph in <blockquote> tags, except the last "For more information" line. Do not format headings as blockquotes. Notice that the tags are properly nested: The <blockquote> tags are not placed inside of the <p> tags, nor are the <h1> tags inside of a <blockquote> tag. When you are finished, resize your browser window so your page resembles Figure 2-10.


Habitat for Humanity Interr
file:///C:/Users/Paddo/Desktop/Habitat/CCYP/syb.html

Join a Summer Build for Teenagers

The Summer Youth Blitz is a unique service experience for a diverse group of youth, ages 16 to 18, from high schools and youth organizations around the United States. During this program, 15 to 20 youth participants and adult leaders "blitz build" an entire Habitat house in two weeks.

The house build, an unfamiliar activity for most, provides a common, nonthreatening ground for building relationships and teams.

In the evenings, the youth participate in activities like team-building games, leadership development, local cultural events or community dinners.

Sponsors

This program is sponsored by national grant donations and coordinated by the Campus Chapters and Youth Programs department of Habitat for Humanity International. For the past several years, funding has allowed for three blitz builds per summer--two in the United States and one outside the country. The builds take place during the months of June, July and August.

Apply Now!

If you're interested in participating, and are between the ages of 16 and 18 years old, you must submit an application, which is available during January and February.

For more information, contact us at (800) 422-4828, ext. 2220.

Figure 2-10: File syb.html after indenting with <blockquote> and centering text

6. Close all browser and editor windows.

In this lab, you used HTML to center and indent text in a Web page.

Additional block-level elements

You can incorporate additional block-level elements into your pages. These include forms, horizontal ruling lines, and lists.

By this point, you should understand how to use the most common block-level tags:

- <p>
- <h1> through <h6>
- <blockquote>

<p style="text-align:center"> Other elements are discussed later in this lesson and in the lessons that follow.

Additional paragraph tags are not added to your code when you use block-level elements. The block-level elements are interpreted by the client browser, which automatically includes the additional spacing.

Text-Level Elements

Text-level elements can affect a section of text as small as a single character or as large as an entire page. In the discussion that follows, you will learn how to use several text-formatting elements to emphasize text and embellish your pages.

OBJECTIVE
2.1.3: HTML text formatting

Bold and italic text

Simple text-level elements include the following:

- <strong> for **bold** text
- <em> for *italic* text

Underlined text will not be covered in this course. As a general rule, you should never use underline because this convention designates hyperlinks in Web page text. It will confuse your Web page viewers. If you still want to use underlined text, visit the CSS page at *www.w3schools.com.*

Text-formatting tags are simple to use. Open the tag before the text to be affected, and close the tag where you want that effect to end. Ensure that tags are nested properly: The tag that opens first closes last, and the tag that opens last closes first..

OBJECTIVE
2.1.3: HTML text formatting

Phrase elements and font style elements

The <strong> element and the older <b> element both create bold text. However, each element accomplishes this effect differently. The difference is that <b> specifically means apply the bold font style, whereas <strong> indicates that the text is to be given a strong appearance. In short, <b> represents a font appearance instruction, whereas <strong> represents the weighting of the phrase relative to surrounding text. The <b> element is called a font style element; <strong> is called a phrase element. The same is true of the older <i> element and the current <em> element, respectively, which both create italic or emphasized text.

Outside of ancient rhetoric, there is no such thing as "bold" speech, but the term "strong" can be used both to denote bold text when printed and strongly spoken text when output through an audio device.

For printed output, you can use phrase and font elements interchangeably. However, if you are coding for the future (as you should be), you should consider how the markup might be used in a different context, then apply the most appropriate tag.

Tech Note: *As a general rule, use CSS to achieve a richer effect. Phrase elements are used more often for individual pages instead of entire Web sites.*

Many text-level phrase elements output the same appearance. For example, the <code>, <kbd> and <samp> tags all make text appear in a fixed-space font. Phrase tags distinguish normal italic text from word definitions or variable program code within the HTML document. Using phrase tags is also beneficial because it is much easier to search for a specific tag within an HTML file instead of searching all italic, fixed-space or bold text.

Table 2-1 lists text-level phrase elements, their usage and their appearances. Font-style elements are slowly disappearing from the Web and are not emphasized in this course.

NOTE:
You do not need to memorize the text-level phrase elements in this table. However, you should be familiar with them.

Table 2-1: Text-level HTML phrase elements

Phrase Element	Usage	Appearance
<em>	For emphasis	*italic text*
<strong>	For stronger, bolder text	**bold text**
<dfn>	For word definitions	*italic* text
<code>	For program code examples	`fixed-space font`
<kbd>	For user keyboard text to be typed	`fixed-space font`
<samp>	For program sample output	`fixed-space font`
<var>	For variable text in program code	*italic text*

NOTE:
Make sure that you understand the use of the <code>, <kbd> and <samp> tags.

In the following lab, you will use text-level phrase elements to format text on a Web page. Suppose your project manager has asked you to add emphasis to certain phrases with formatting such as italic or bold type. You could perform these steps to add the proper elements to your Web page code.

Lab 2-5: Using text-level phrasing elements in HTML

OBJECTIVE
2.1.3: HTML text formatting

NOTE:
An example of the finished code for this lab is available in the supplemental files.

In this lab, you will add text-level phrasing elements to the file you worked with in previous labs.

1. **Editor:** Open **syb.html**.

2. **Editor:** In the first full paragraph, find the phrase "*Summer Youth Blitz*" and add the <em> tag as shown:

```
<em>Summer Youth Blitz</em>
```

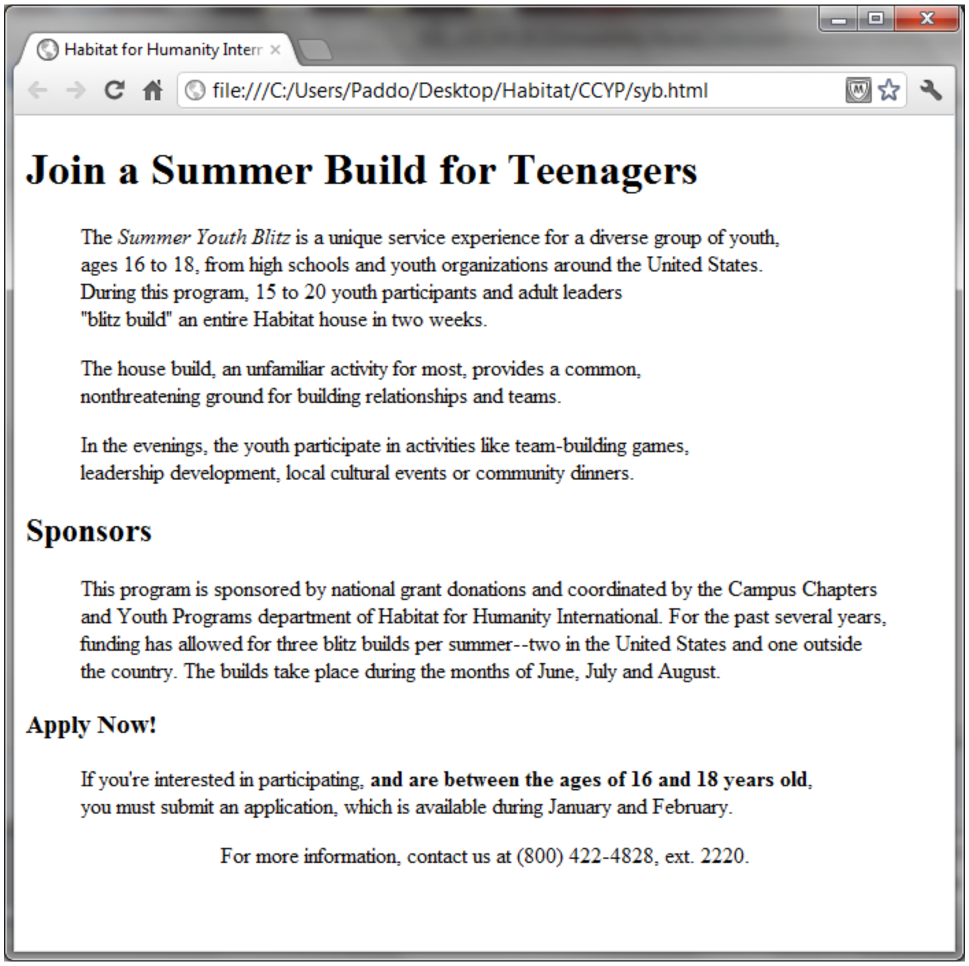
3. **Editor:** Save the file.

4. Review your work in a browser, and then validate it.

5. **Editor:** Find the phrase "*and are between the ages of 16 and 18 years old*," and add the <strong> tags as shown:

```
<strong>and are between the ages of 16 and 18 years old</strong>,
```

6. Save your code and load it into a browser. Your page should resemble Figure 2-11.


Habitat for Humanity Interr
file:///C:/Users/Paddo/Desktop/Habitat/CCYP/syb.html

Join a Summer Build for Teenagers

The *Summer Youth Blitz* is a unique service experience for a diverse group of youth, ages 16 to 18, from high schools and youth organizations around the United States. During this program, 15 to 20 youth participants and adult leaders "blitz build" an entire Habitat house in two weeks.

The house build, an unfamiliar activity for most, provides a common, nonthreatening ground for building relationships and teams.

In the evenings, the youth participate in activities like team-building games, leadership development, local cultural events or community dinners.

Sponsors

This program is sponsored by national grant donations and coordinated by the Campus Chapters and Youth Programs department of Habitat for Humanity International. For the past several years, funding has allowed for three blitz builds per summer--two in the United States and one outside the country. The builds take place during the months of June, July and August.

Apply Now!

If you're interested in participating, **and are between the ages of 16 and 18 years old**, you must submit an application, which is available during January and February.

For more information, contact us at (800) 422-4828, ext. 2220.

Figure 2-11: File syb.html after adding text-level phrase elements

7. **When time permits:** Experiment with using the <code>, <kbd> and <samp> tags. Be sure to delete these tags when you are finished experimenting.

8. Close all browser and editor windows.

In this lab, you added text-level phrasing elements to a Web page.

Now that you know how to use the text-level phrase elements, you will work with lists.

NOTE:
Add a closing tag to each list item. For instance:
<li> This is the first numbered item.
</li>
Even though these closing tags are optional in HTML (i.e., the code will still validate), it is good coding practice to establish the habit of using closing tags.

Lists

A common markup function is to create bulleted and numbered lists. Lists are compound, block-level elements. Encompassed within list definition tags are individual list item tags. A paragraph break automatically precedes and follows the entire list. Individual list items are separated automatically by single line breaks.

There are two types of lists:

- **Ordered list** — a numbered list. Uses the <ol> element and requires a closing tag.
- **Unordered list** — a bulleted list. Uses the <ul> element and also requires a closing tag.

Both list types use identical syntax. Each list item is specified using the list item element:

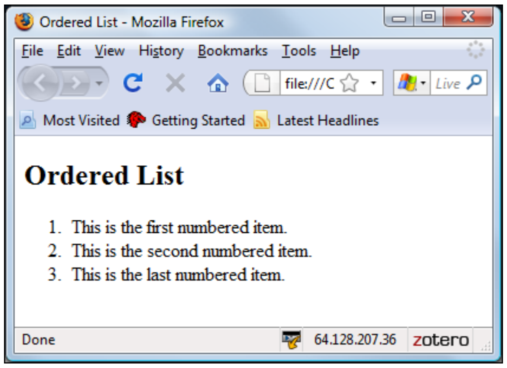
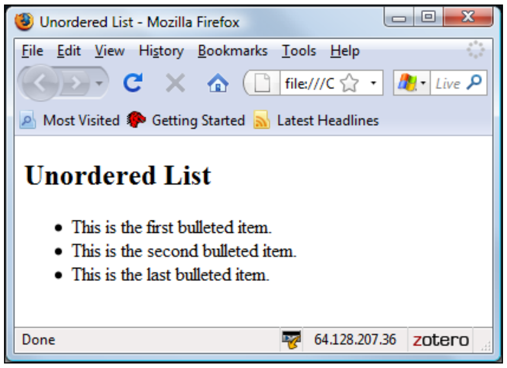
- **List item** — specifies list items in an ordered or unordered list. Uses the <li> element and requires a closing tag.

Contrast the code in the left column of Table 2-2 and its resulting display in Figure 2-12, with the code in the right column of Table 2-2 and its resulting display in Figure 2-13.

Table 2-2: Ordered and unordered list syntax and display

Ordered List	Unordered List

`<h2>Ordered List</h2>`
`<ol>`
`<li>This is the first numbered item.</li>`
`<li>This is the second numbered item.</li>`
`<li>This is the last numbered item.</li>`
`</ol>`
	`<h2>Unordered List</h2>`
	`<ul>`
	`<li>This is the first bulleted item.</li>`
	`<li>This is the second bulleted item.</li>`
	`<li>This is the last bulleted item.</li>`
	`</ul>`

 Ordered List - Mozilla Firefox Ordered List 1. This is the first numbered item. 2. This is the second numbered item. 3. This is the last numbered item. *Figure 2-12: Ordered list*	 Unordered List - Mozilla Firefox Unordered List • This is the first bulleted item. • This is the second bulleted item. • This is the last bulleted item. *Figure 2-13: Unordered list*

In the following lab, you will use HTML to create bulleted and numbered lists. Suppose your supervisor has given you the following text to add to the Web page:

> "During the house build, you will: help lay the foundation, assist in framing the home, and do simple carpentry under supervision."

You can see that a list format would work well for this multi-point information. You can experiment with both an ordered and an unordered list. Which is most appropriate for the information?

Lab 2-6: Creating lists with HTML

OBJECTIVE
2.1.3: HTML text formatting

In this lab, you will create a bulleted list and a numbered list on a Web page.

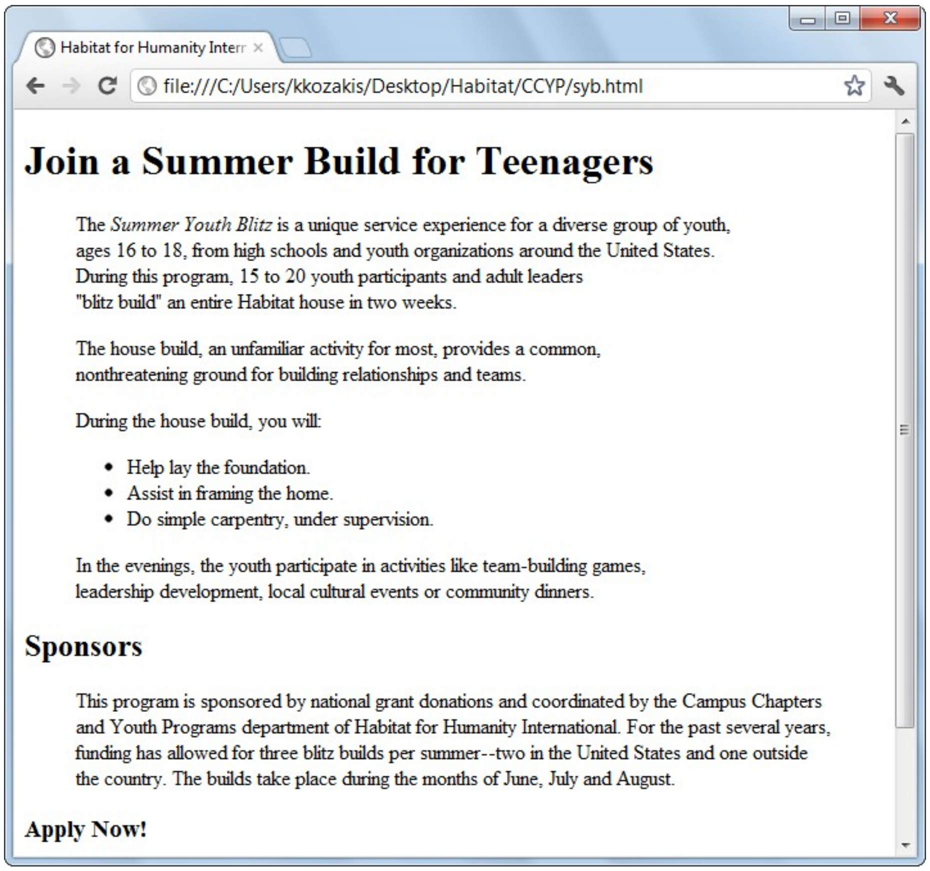
1. **Editor:** Open **syb.html**.

2. Create empty space by adding a return immediately beneath the </p> tag located after the text that reads "*for building relationships and teams.*"

3. Add the text shown and format it as an unordered (bulleted) list. Make sure that the list (beginning with the <ul> tag) is not placed within a set of <p> tags:

```
<blockquote>
<p>
The house build, an unfamiliar activity for most, provides a common,
<br/>nonthreatening ground for building relationships and teams.
</p>
<p>During the house build, you will:</p>
<ul>
<li>Help lay the foundation. </li>
<li>Assist in framing the home. </li>
<li>Do simple carpentry, under supervision. </li>
</ul>
</blockquote>
```

4. **Editor:** Save your changes and view the page in a browser. Your page should resemble Figure 2-14.



Habitat for Humanity Interr
file:///C:/Users/kkozakis/Desktop/Habitat/CCYP/syb.html

Join a Summer Build for Teenagers

The *Summer Youth Blitz* is a unique service experience for a diverse group of youth,
ages 16 to 18, from high schools and youth organizations around the United States.
During this program, 15 to 20 youth participants and adult leaders
"blitz build" an entire Habitat house in two weeks.

The house build, an unfamiliar activity for most, provides a common,
nonthreatening ground for building relationships and teams.

During the house build, you will:

- Help lay the foundation.
- Assist in framing the home.
- Do simple carpentry, under supervision.

In the evenings, the youth participate in activities like team-building games,
leadership development, local cultural events or community dinners.

Sponsors

This program is sponsored by national grant donations and coordinated by the Campus Chapters and Youth Programs department of Habitat for Humanity International. For the past several years, funding has allowed for three blitz builds per summer--two in the United States and one outside the country. The builds take place during the months of June, July and August.

Apply Now!

Figure 2-14: File syb.html after adding bulleted list

5. Validate your code at ***http://validator.w3.org***.

6. Change your unordered list to an ordered list, then validate your code again. View the page with the ordered list in the browser. Do you think a numbered list format is more appropriate for this information than a bulleted list?

7. Change your code back to an unordered list, then close all editors and browsers.

In this lab, you created ordered and unordered lists.

NOTE:
To review the terms introduced in this lesson, see **Activity 2-1: Identifying HTML elements and attributes**.

Good Coding Practice

Now that you have learned the basics of working with HTML5 elements and tags, you should consider not simply which elements to use, but how to best use them in conjunction with your text.

Forward-compatibility

Remember that good coding practice involves writing code for forward-compatibility. That means you should always make your code cleaner to allow an easier transition to updated standards in the future. Such practices include:

- Closing all tags.
- Using lowercase letters within tags.
- Surrounding attribute values with quotation marks.

OBJECTIVE
2.1.8: Using a single W3C standard

Universal markup and consistency

As previously discussed, another facet of good coding practice is creating universal markup that applies W3C standards consistently and thus renders consistently across most or all browsers. Remember to choose one HTML standard for your Web document, pages or site, then apply that standard carefully and consistently throughout. Applying the syntax rules of multiple standards in the same document or site not only prevents your code from validating but also produces unexpected rendering results in your users' browsers.

Note that this point does not contradict the previous point about applying the stricter syntax rules of XHTML in your HTML documents for the purpose of forward-compatibility. Using stricter syntax than required will rarely produce output problems. Inconsistency is mostly an issue with the older HTML standards because their looser rules allowed some sloppier coding practices to render without penalty. Keep in mind that you should understand the requirements of whichever standard you are using, and be particularly aware of syntax and tag usage.

Readability

NOTE:
You can practice converting HTML 4.01 code to HTML5 in **Optional Lab 2-1: Converting HTML 4.01 Transitional to HTML5.**

If you are coding an HTML page and you are the only one who will ever look at the code, you may think the appearance of your code does not matter. This statement is basically true. But suppose you must share your work with others. Some coding techniques provide better readability, and make finding and changing code a simple operation. Other coding techniques produce a busy, confusing format that makes it difficult to decipher and edit the code.

Examine the two boxes of HTML code in Figure 2-15. Both sets of code will render the same in a browser, but clearly one set of markup is easier to read than the other. Suppose you were hiring someone to write HTML code for you. Would you be more inclined to hire the developer of the code on the left, or the developer of the code on the right? The code on the right is much more readable.

```
<html><head><title>Overview
</title></head><body><h1>
Looking Back</h1></body>
</html>
```

```
<html>
<head>
<title>Overview</title>
</head>
<body>
<h1>Looking Back</h1>
</body>
</html>
```

Figure 2-15: Same HTML code with different line breaks

Exceptions

In some cases, you may find it impossible to make the code more readable without affecting the way it renders in the browser. With older XHTML code involving images or the <div> tag, you may find that entering random white space can affect rendering in certain browsers. Always try to make your code readable, but verify that it renders properly.

OBJECTIVE
2.1.7: HTML code comments

Adding hidden comments

You can hide comments within your HTML source code that will not appear on the page. The syntax for including a comment within your HTML document is as follows:

```
<!-- comment text here -->
```

When to use comments

When creating markup pages, you can use comments to:

- "Comment out" code to see how a page will appear without a particular markup element.
- Inform others about important elements in the code you are creating.
- Remind yourself why you inserted a particular piece of code.
- Insert programming code, such as JavaScript.

CIW Online Resources – Online Exercise

Visit CIW Online at *http://education.Certification-Partners.com/CIW* to complete an interactive exercise that will reinforce what you have learned about this topic.

Exercise 2-2: Using comments

In the following lab, you will insert hidden comments into your HTML code. Suppose your project has incurred a change and you have been asked to temporarily remove the bulleted list. You can use hidden comments to add notes to the file that document this change, and also make it easy to reverse it if necessary.

Lab 2-7: Documenting and commenting HTML code

In this lab, you will document your HTML code and "comment out" certain portions.

1. **Editor:** Open the **syb.html** file.
2. **Editor:** Comment out your bulleted list and the introduction to it using the <!-- and --> tags.

```
</p>

<--
<p>During the house build, you will:</p>
<ul>
<li>Help lay the foundation. </li>
<li>Assist in framing the home.</li>
<li>Do simple carpentry, under supervision. </li>
</ul>
-->

</blockquote>
```

3. Save your changes and view your edited file in a browser. You will no longer see the bulleted list and its introduction.
4. Validate your code to ensure that you have used the comments properly. Sometimes adding comments can cause you to mistakenly omit closing tags and/or interrupt a nesting sequence.
5. Document the reason that you removed the bulleted list by creating another comment immediately after the bulleted list you just commented out:

```
<!-- Bulleted list removed at the request of supervisor, Jane Doe.-->
```

6. Near the bottom of the file, just above the </body> tag, document your code so that another developer can identify who wrote it. Insert your name, the date, and a statement that this code validates to HTML5:

```
<!-- Your Name, Today's Date. This code validates to HTML5-->
```

7. Validate your code again.
8. Review your code in the text editor. Make sure that it is easy to read in terms of good coding practice.
9. **If time permits:** Create additional comments explaining the code.
10. Close all editors and browsers.

In this lab, you documented and commented out portions of your markup.

CIW Online Resources – Course Mastery

Visit CIW Online at *http://education.Certification-Partners.com/CIW* to take the Course Mastery review of this lesson or lesson segment.

SDA Lesson 2 - Part B

Case Study

HTML Convert

Vlad works as a developer on his company's Web team. He has been assigned to convert the Web site's code from HTML 4.01 to HTML5. He developed a conversion plan that included the following steps:

- Use a valid <!DOCTYPE> declaration for all pages.
- Ideally, use style sheets. Developers do not necessarily need to use CSS, but it is recommended.
- Identify deprecated and forbidden tags already in use. Determine replacement tags and/or other methods of achieving the same effects using proper HTML5 markup.
- Estimate the necessary time and resources to make these changes.

Vlad presented to his project manager an example of a converted page, a new style sheet, and an estimate for the time it would take to complete the conversion process. His project manager was able to obtain funding for the conversion.

* * *

As a class, discuss the following questions.

- Suppose Vlad uses a GUI HTML editor. What changes and/or updates must he make?
- What <!DOCTYPE> declaration should Vlad choose if he wants the site to be HTML5-compliant as quickly as possible?
- Considering that Vlad must identify and change deprecated tags, should he add time to the project?

Lesson Summary

Application project

You have already learned many of the basic tags that HTML provides for formatting text and paragraphs. In the lessons that follow, you will learn how to incorporate graphics, create links, use external CSS, work with tables and create HTML forms. Now, consider the following:

- What is the difference between a container element and an empty element? Consider that a container element must contain the text that it formats between opening and closing tags.
- Review all the HTML code you have created. Verify that it adheres to the good coding practices discussed in this lesson. If not, modify your code as necessary and save the files. As your Web site becomes more complicated throughout this course, you will be able to quickly locate code for modification within your files.

Skills review

In this lesson, you learned to use container and stand-alone HTML elements and their tags. You learned the basic structure elements that must be present in any HTML document. You learned how to format both text and paragraphs, and how to create a bulleted list and a numbered list. Finally, you were introduced to the concept of good coding practice, and the importance of correct application and sequence of your code.

Now that you have completed this lesson, you should be able to:

✓ 2.1.3: Use HTML elements and tags to format paragraphs and text.

✓ 2.1.7: Add comments to HTML code and document page/site creation.

✓ 2.1.8: Explain the importance of consistently developing to a single W3C standard (e.g., HTML5).

✓ 2.7.4: Validate Web page design according to technical and audience standards adopted by employers.

✓ 2.9.2: Create a Web page using the HTML5 standard.

✓ 2.12.1: Test and validate Web documents.

CIW Practice Exams

Visit CIW Online at *http://education.Certification-Partners.com/CIW* to take the Practice Exams assessment covering the objectives in this lesson.

SDA Objective 2.01 Review

SDA Objective 2.07 Review

SDA Objective 2.09 Review

SDA Objective 2.12 Review

Note that some objectives may be only partially covered in this lesson.

Lesson 2 Review

1. Markup tags include container, empty and stand-alone non-empty tags. Which tag types are valid in HTML?

2. Are HTML tags case-sensitive?

3. What three items can be contained inside the angle brackets (wickets) of an HTML tag?

4. What is the function of the <!DOCTYPE> declaration?

5. Define text-level element.

6. What notation can you use to write a note to yourself or others in the HTML code that will not appear in the page when rendered in a browser?

7. Should all values be placed in quotation marks for HTML?

CIW®

Certified Internet Web Professional