Web Module 2:

HTML, Part I

Introduction

In this module, the first of a two-module sequence on HTML (HTML Part I and Part II), we'll introduce you to HTML, "the language of the World Wide Web." HTML is the coding language you use to create web pages and sites.

When you're done working through this module, you'll know:

- How HTML works
- How to create an HTML5 document
- How to code HTML elements and attributes
- The four pillars of web design
- The difference between structural and presentational markup
- How to create links
- How to create bulleted and numbered lists

Caveat Lector

These eight Web modules build on each other. For example, in Module 4 it's assumed you understand the concepts and techniques from Modules 1-3. So, if you run into anything you're unfamiliar with in this module, browse through the earlier modules or turn to Google for help.

HTML Basics

What is a web page?

An HTML document that resides on a web server and is displayed in a browser.

What is an HTML document?

A plain-text document with a file extension of .htm or .html:

index.htm, index.html, schedule.htm, glossary.html, etc.

HTML stands for HyperText Markup Language, which is a way of "marking up" text (and other content) so that it can be viewed in a web page.

What is a plain-text document?

A document whose content consists solely of the 95 ASCII "printable" characters – letters, numbers, and symbols – with no support for text formatting.

Here's a chart of the 95 ASCII printable characters:
en.wikipedia.org/wiki/ASCII#ASCII_printable_characters

You create a plain-text document by using a text editor.

Popular Macintosh text editors include TextWrangler (free) and BBedit. Popular Windows text editors include Notepad++ (free) and Sublime Text. But there are plenty more good text editors out there; turn to Google for help.

Warning! Do NOT use Microsoft Word or any other word processor whose files have a proprietary extension. If you do, your HTML files won't work.

What about Unicode?

HTML documents can also be written in Unicode, which contains all the ASCII characters plus an additional ~100,000 characters for world languages.

Can HTML documents display anything other than text?

O ja! They can display images, video, sound, vector graphics, animations, etc.

But, no matter what kind of media an HTML document displays, the document itself consists only of plain text.

What is HTML?

HTML is the primary language used to create pages and sites on the World Wide Web.

The World Wide Web – also known as the WWW, W3, or simply the web – is a system of interlinked hypertext documents (a web of docs) accessed via the Internet.

The Internet is a global network of computers.

HTML stands for HyperText Markup Language.

HyperText is text in one digital document that is connected, via a hyperlink (link), to text in a different document or a different part of the same document.

HyperText is what makes the WWW the WWW: the ability to link from one document to another, thus creating a web of information.

Markup language adds codes to plain text to structure it and make it more readable.

Creating an HTML Document

To create an HTML document (web page):

Either use a text editor to enter the HTML code and page content by hand.

Popular Macintosh text editors include TextWrangler (free) and BBedit. Popular Windows editors include Notepad++ (free) and Sublime Text.

Or use an HTML authoring tool.

Dreamweaver is the most popular HTML authoring tool for Macs and PCs. KompoZer is a free alternative for both platforms. For more, turn to Google.

HTML5

HTML5 is the latest and greatest flavor of HTML. It's powerful, efficient, and almost all recent browser versions support it.

For these reasons, you should code your web pages to the HTML5 standard, not to an earlier HTML (or XHTML) standard.

Note: The one exception to this is if you are creating web pages for a specific audience using older browsers that don't support HTML5.

Here's a beautiful interactive chart (html5readiness.com) depicting HTML5 and CSS3 support for major browsers.

Anatomy of a Web Page

A web page (HTML document) is composed of HTML elements.

For example: headers, paragraphs, links, lists, tables, images, etc.

You use tags to create HTML elements. Tags are strings of HTML code enclosed in <...> angular brackets. For example:

`<h1>Heading Text</h1>` – a level-1 heading

`<p>paragraph text ...</p>` – a <p> (paragraph) element

`<img src="image.jpg">` – an <img> (image) element

`<a href="http://www.w3.org/">W3C</a>` – an <a> (anchor, or link) element

Creating a Web Page by Hand

Here's the general process for creating a web page by hand (rather than in an HTML authoring tool like Dreamweaver):

1. In a text editor open an HTML5 document template. (See next section.)
2. Enter the page title (where indicated in the template).
3. Enter the page content (where indicated in the template).
4. Use HTML elements to mark up the content.
5. Save the file as *name*.html (where *name* is your desired file name).
6. View the page in a browser.
7. Modify the page's HTML code as desired. ← Iterate!
8. Resave the page and reload it in the browser.
9. Loop steps 6-8 (modify, resave/reload) until the page is finished. This is called iterative design, and it's a practice all web developers know very very (very!) well.

HTML File Naming Rules

An HTML document filename should not have any spaces or special characters (%$#@&^) in it.

Yes: my-portfolio.htm, file1.htm, index.html

No: my portfolio.htm, file@ 1.htm, index$.html

Use an underscore (_) instead of a space.

Yes: file_1.htm, index_page.html, my_portfolio.htm

No: file 1.htm, index page.html, my portfolio.htm

The filename should only have lowercase letters.

Yes: file1.htm, index.html, my – portfolio.htm

No: File1.htm, iNdex.html, My – Portfolio.htm

The filename must end in .html or .htm

Generally the .html and .htm extensions work equally well. But .html is slightly more universally compatible with servers.

The moral: Use .html, not .htm.

HTML5 Document Template Primer

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>pagetitle</title>
</head>
<body>
  pagecontent
</body>
</html>
```

Here's the bare-bones code for a valid* HTML5 document. Think of it as a skeleton for a HTML5 web page that you, the developer, will flesh out with content and code.

* A valid HTML5 document is one that adheres strictly to the HTML5 standard. For more on HTML validation, see the HTML, Part II module.

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>pagetitle</title>
</head>
<body>
   pagecontent
</body>
</html>
```

What is this first line of code?

A doctype declaration.

What does the doctype declaration do?

It tells the browser what version of HTML is used in the document, HTML5 in this case.

Is it required?

Yes, if your goal is to write valid HTML5 code.

Where should it appear?

Right the top of the HTML document, before the opening <html> tag.

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>pagetitle</title>
</head>
<body>
   pagecontent
</body>
</html>
```

What is this code?

The opening and closing tags of the document's html element.

Where should it appear?

Enclosing everything after the doctype element.

What is lang="en" ?

lang is a required attribute (for valid HTML5 documents) that specifies the language of the html element, everything between <html> and </html>.

Here lang is assigned the value "en" (English).

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>pagetitle</title>
</head>
<body>
   pagecontent
</body>
</html>
```

What is this code?

The required head element of the HTML document.

Where should it appear?

Right after the opening <html> tag.

What is meta?

A required standalone element that provides the metadata attributes for an HTML5 doc.

What is charset="utf-8" ?

A required attribute (for valid HTML5 documents) that specifies the character set of the document, "utf-8" (Unicode) in this case.

What is title?

A required container element that specifies the title of the web page.

What is pagetitle?

A placeholder for the title of the page.

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>pagetitle</title>
</head>
<body>
   pagecontent
</body>
</html>
```

What is this code?

The required body element of the HTML document.

What goes in the body of an HTML5 document?

All of the page's content (text, links, pointers to images, audio, video files, etc.) and HTML code. Only content that resides in the body element (between <body> and </body>) will be displayed when the page is loaded in a browser.

Where should it appear?

Between the head element and closing </html> tag.

What is page content?

A placeholder for the content of the page.

Practice: Creating an HTML5 Template

To write valid HTML5 code, you need to begin each page with a valid HTML5 template. In this hands-on activity, you'll create this template.

1. In your text editor of choice, create a new plain-text file. (Do NOT use MS Word!)
2. Copy/paste this HTML5 code to the file:

```
<!doctype html>

<html lang="en">

<head>
   <meta charset="utf-8">
   <title>pagetitle</title>
</head>

<body>
   pagecontent
</body>

</html>
```

3. Save the file as html5_template.html to your desktop or an appropriate folder.

   Make sure to save the file as a plain-text file! If you save it as a formatted file, it won't work as an HTML5 template.

You can use this HTML5 template every time you create a new HTML5 page by hand. Simply open the template, save it with a new name, replace pagetitle with the actual page title, and pagecontent with the page content and HTML code. Let's try it out:

1. In your text editor, open html5_template.html.
2. Save the file with a new name. You don't want to overwrite your template!
3. Replace pagetitle with the actual title of your page.
4. Replace content with this Lorem Ipsum text:

   The Standard Lorem Ipsum Passage

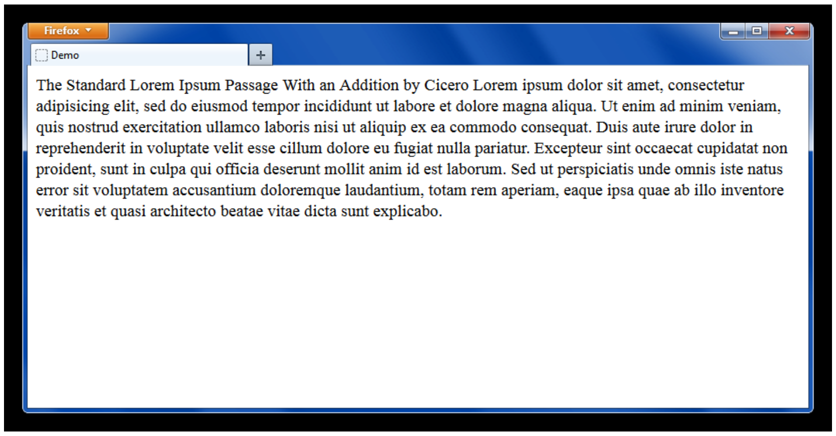
   With an Addition by Cicero

   Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

   Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

   Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

5. Resave the file (with the same name you chose in step 2) and open it in a browser. Congratulations: You've created a valid HTML5 web page! But it looks terrible! This is what happens when you enter text in an HTML document with no HTML markup tags: The browser displays it as an unbroken sprawl of identically formatted text.



The Standard Lorem Ipsum Passage With an Addition by Cicero Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

6. Let's mark up the text and see what happens. Add the following h1 (level 1 heading), h2 (level 2 heading), and p (paragraph) tags to the Lorem Ipsum passage:

<h1>The Standard Lorem Ipsum Passage</h1>

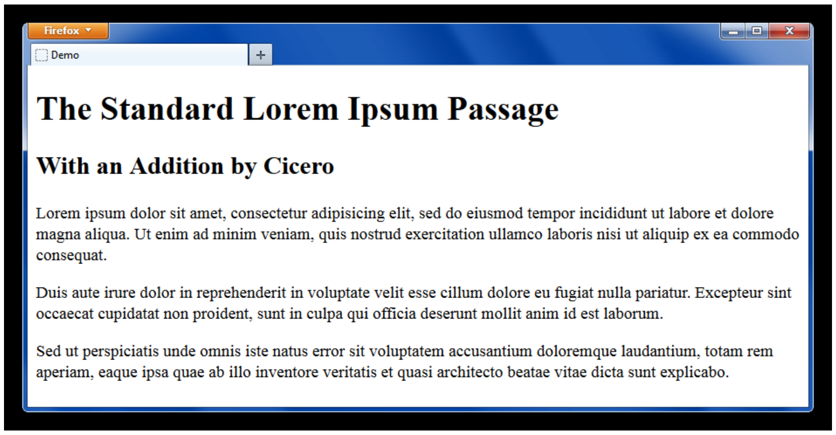
<h2>With an Addition by Cicero</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>

<p>Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

<p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.</p>

7. Resave the file and reopen it in your browser. Much better, yes?



The Standard Lorem Ipsum Passage

With an Addition by Cicero

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Container and Empty HTML Elements

In terms of how they're coded, there are two types of HTML elements: container and empty.

Container elements have two tags, an opening tag and a closing tag, and contain (enclose) text to be modified by the element:

Container element syntax:

```
<elementName>text</elementName>
```

An example of a container element:

```
<p>Text contained in p tags is formatted as a paragraph.</p>
```

Empty elements have one tag:

Empty element syntax:

```
<elementName>
```

An example of an empty element:

```
<hr>
```

The hr (horizontal rule) element draws a line across the entire width of the page.

Empty elements are sometimes called standalone elements.

They stand alone <*elementName*> – not in pairs <*elementName*>...</*elementName*>.

Commonly Used Container Elements

Container tags modify the text they contain (enclose). For example:

```
<p>I am a paragraph, and as such should really have at least 2-3
sentences. I am a paragraph, and as such should really have at
least 2-3 sentences. I am a paragraph, and as such should really
have at least 2-3 sentences.</p>

<h1>I am a level 1 (main) heading.</h1>

<h2>I am a level 2 (section) heading.</h2>

<h3>I am a level 3 (subtopic) heading.</h3>

<div>I create a div (box) on the page.</div>

<strong>I make text strong by bolding it.</strong>

<em>I em(phasize) text by italicizing text.</em>

I create a <a href="http://google.com">link</a> to another page.
```

Commonly Used Empty Elements

Empty tags don't contain (enclose) text.

`<br>` – moves down to the beginning of the next line on the page

`<hr>` – draws a line across the entire width of the page

Nesting Elements

You can nest HTML elements: Enclose them within other elements.

In this example, nesting an em element within a strong element causes Hello! to be displayed both strong (bolded) and em-phasized (italicized):

```
<strong><em>Hello!</em></strong>
```

Nesting Rule

The first element opened must be the last element closed.

```
<elem1><elem2><elem3>content</elem3></elem2></elem1>
```

Think of the tags as being symmetrically mirrored around the content.

Examples:

`<strong><em>Hello!</em></strong>` – correct nesting
`<strong><em>Hello!</strong></em>` – incorrect nesting

Practice: Container and Empty Elements

1. Use your trusty HTML5 template (the one you built above in Creating an HTML5 Template) to create a web page named html_mod1.html. Give it an appropriate *pagetitle*. You'll be using this page to store all of your coding efforts in this module.
2. Experiment with adding different HTML container and empty elements to the page body (between the <body> and </body> tags). Add all the examples in this section, then grab 5-10 more from the W3Schools HTML Reference (w3schools.com/tags/default.asp).
3. Try nesting container and empty elements within container elements. Pay special attention to what properties of the outermost nested elements are inherited by the inner elements, and which are not inherited. (You'll learn about inheritance in the CSS, Part II module.)

Block and Inline Elements

In terms of where they're displayed on a web page when viewed in a browser, there are two types of HTML elements: block elements and inline elements.

- Block elements – are displayed in their own line/area on the page

  Block elements can contain other block or inline elements.

  Examples: <p>, <h1>, <ol>, <ul>, <blockquote>, <div>.

- Inline elements – are displayed in the same line as the surrounding content

  Inline elements contain other inline elements, but NOT other block elements.

  Examples: <a>, <big>, <strong>, <em>, <img>.

Try it out: Add a few block elements to a page until you get a feel for how they display in their own lines/areas. Add some inline elements to see how they share lines with other content.

HTML Attributes

An HTML attribute specifies a property or behavior of the HTML element whose tag it resides in. For example:

```
<img src="mugshot.gif">
```

The src attribute specifies the URL of the image that the <img> element displays.

Here's the syntax of an HTML attribute:

```
<elementName attributeName="attributeValue">
```

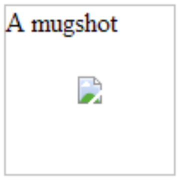
elementName is the name of the HTML element, img in the previous example.

attributeName and "*attributeValue*" are the name and value (always in straight "..." quotes) of the attribute, src in the example.

An HTML element can have two or more attributes. The opening img example had one, src. This img example has two, src and alt:

```
<img src="mugshot.gif" alt="A mugshot">
```

The alt attribute specifies the text that will display in place of the image in image-disabled browsers, like this:



Global HTML Attributes

In addition to the attributes that are specific to certain HTML elements – for example, the <img> element's src and alt attributes – there is a set of a dozen or so attributes that can be used with any HTML element. These are called HTML global attributes.

You'll be using a few of these global attributes during the course of the four HTML and CSS modules, including: class, id, and style. For now, all you need to know is they exist.

Attribute Coding Rules

An HTML element can have 0, 1, or 2+ attributes.

Attributes always go in the opening tag of a container element.

Or in the only tag of an empty element, since they only have one tag.

Multiple attributes can go in any order. These are rendered identically by a browser:

```
<img src="cup.gif" alt="Cup">
<img alt="Cup" src="cup.gif">
```

An attribute must always be assigned assigned a value (with the = operator):

```
attributeName="value"
```

Always put the value in straight quotes: "..."

Using curved quotes: “...” can cause the server to misinterpret the value in the quotes, which can in turn mess up the page display.

Element and attribute names should be all lowercase.

Yes: `<p align="right">`

No: `<P ALIGN="right">`, `<P align="right">`, etc.

Don't use attributes that HTML doesn't support.

Older versions of HTML (4.01, for example) support elements and attributes that HTML5 doesn't. Though it can be tempting to use some of these unsupported items in your HTML5 code – many are quite handy! – it's poor web development to do so.

Note: Most of the features of the unsupported items are now handled by CSS.

To find out if an element is supported by HTML5, turn to the W3Schools HTML Reference (w3schools.com/tags/default.asp). Unsupported elements are in red and marked "Not supported by HTML5." To find out if an attribute is supported, click on the link to its element and view the Attributes table.

Practice: HTML Attributes

1. Go to the W3Schools HTML Reference (w3schools.com/tags/default.asp) and find out:

   What do these elements do and which are supported by HTML5: <abbr>, <basefont>, <button>, <dir>, <div>, <frame>, <frameset>, <section>, <tt>, <u>.

   Which attributes (if any) are supported by each of these elements: <blockquote>, <body>, <form>, <img>, <p>, <textarea>, <ul>.
2. In your html_mod1.html file (the one you built above in Container and Empty Elements) , try out the HTML5-supported attributes of these elements: <a>, <blockquote>, <img>, <menu>, <source>, and <style>. In addition, try adding the global attribute title to an <img> and <p> element. Cool, eh?

The Four Pillars of Web Design

Web design rests on these four pillars: content, structure, presentation, and behavior.

- Content

  Meaningful text, images, audio, and other media.
- Structure

  The division of page content into functional units: headings, paragraphs, images, links, etc. The structure of a web page is defined by the HTML tags you add to the content.
- Presentation

  The appearance of the content = how the page looks in the browser: layout, formatting, line spacing, etc. The presentation of a web page is defined by the CSS styles you use.
- Behavior

  The interaction of the user and the page content – rollovers, slide shows, pop-ups, etc. Web page behavior is implemented with JavaScript or plug-ins like Flash.

In the two HTML modules we'll focus on structure, and in the two CSS modules on presentation. We'll get to behavior in the Javascript and Forms modules.

Structural and Presentational HTML

HTML supports structural and presentational elements.

Structural (or semantic) HTML elements give meaning and structure to page content.

They're used to define headers, footers, navbars, paragraphs, etc.

This was the original purpose of HTML and the current purpose of XML/XHTML.

Presentational HTML elements determine the appearance of page content.

They're used to set fonts (face, size, style, alignment), colors, line spacing, etc.

This is now primarily the task of CSS, not HTML.

Some commonly used structural and presentational HTML elements:

Structural HTML elements	Presentational HTML elements
<html> – html element	<b> – bold
<head> – head element	<i> – italics
<body> – body element	<big> – large font
<h1> – level 1 header	<small> – small font
<p> – paragraph	<font> – multiple font attributes
<div> – box	<center> – center align

An Example of What NOT to Do

This body element uses lots of presentational markup tags (red):

```
<body>
  <font size="6"><b>My Favorite Joke</b></font><br><br>
  What did the fish say when he <i>bumped</i> his head?<br><br>
  <big><b>Dam!</b></big><br><br>
  <b>(I'm here all week.)</b>
</body>
```

The presentational markup elements:

font – font size

i – italics

b – bold

big – big text

br – line break

Why is this an example of what not to do? Because you should use HTML for structural markup, not presentational markup. Leave page presentation to CSS.

An Example of What TO Do

Here's the same body element using structural markup:

```
<body>
<h1>My Favorite Joke</h1>
<p>What did the fish say when he <em>bumped</em> his head?</p>
<h3>Dam!</h3>
<strong>(I'm here all week.)</strong>
</body>
```

The structural markup elements:

h1 – level 1 heading (very large and bolded)

p – paragraph

em – emphasis, usually displayed by the browser as italics

strong – usually displayed as bold

h3 – level 3 heading (somewhat large and bolded)

This is good coding, because it uses HTML tags for structural – not presentational! – markup.

Practice: Structural HTML

1. Replace all of the presentational HTML tags in this passage with structural tags:

```
<font size="6"><b>Lorem Ipsum</b></font><br><br>

<big><b>With an Addition by Cicero</big></b><br><br>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.<br><br>

Duis aute irure dolor in <b><i>reprehenderit in voluptate
velit</i></b> esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.<br><br>

Sed ut perspiciatis unde omnis iste natus error sit voluptatem
accusantium doloremque laudantium, totam rem aperiam, eaque ipsa
quae ab illo inventore veritatis et quasi architecto beatae vitae
dicta sunt explicabo.<br><br>
```

The <style> Element

The <style> element is used to define CSS (Cascading Style Sheets) style rules in a web page.

CSS style rules determine the appearance of HTML elements.

They're presentational, not structural.

Here's the syntax of the <style> element:

```
<style type="text/css">
style-rules
</style>
```

type="text/css" assigns the value "text/css" to the property type. This tells the browser that the tags <style type="text/css">...</style> enclose CSS styles.

For example, this <style> element contains a style rule that will display all <p> (paragraph) elements on the page in a 24-point red, Arial font.

```
<style type="text/css">
p { color: red; font-family: Arial; font-size: 24pt; }
</style>
```

Try it out: Add the above <style> element to the Lorem Ipsum web page you created earlier on. If your code is correct, all paragraph text (not headings) will be displayed in 24-point red Arial.

You'll learn all about CSS styling in the two CSS modules. Patience!

Creating Links

You use the <a> (anchor) element to create a link on a page. Here's the syntax:

```
<a href="URL">link-name</a>
```

href (hypertext reference) is an attribute of the <a> element.

URL is the relative or absolute URL of the page (image, audio file, etc.) being linked to.

link-name is the text (or image) that, when clicked, will jump to the href URL.

For example:

```
<a href="recipe.html">My Recipe</a>
```

<a starts the opening tag of the <a> element.

"recipe.html" is the value assigned (via =) to the href attribute.

It's the relative URL of the linked page. (More on relative/absolute URLs soon.)

> (after "recipe.html") closes the opening <a> tag.

"My Recipe" is the link text that appears in the browser.

When a user clicks on the My Recipe link, the browser jumps to recipe.html

Links are typically underlined, but you can change this with CSS. (See CSS modules.)

</a> closes the <a> element.

What is a URL?

URL = Uniform Resource Locator.

A string of letters, numbers, and symbols that identifies a unique web resource: page, image, mp3 song, mp4 movie, etc. Think of a URL as the "address" of an item on the web.

URL syntax:

`http://`*`server`*`/`*`file`*

http:// (or https:// for secure servers)

Stands for Hypertext Transfer Protocol (or Hypertext Transfer Protocol Secure).

server – 2+ part domain name

For example: www.cnn.com, www.rit.edu, etc.

You can usually leave the www out; the server adds it in: nytimes.com, rit.edu.

file – folder (path) and file name

index.html, music/music.htm, etc.

URL examples:

http://cnn.com, https://mycourses.rit.edu, http://rachmiel.org/music/music.htm

If no file name is included, the browser opens the index.* file in that folder:

http://cnn.com opens http://cnn.com/index.html

Relative and Absolute URLs

Relative URL

Specifies a file relative to the folder in which the HTML file containing the URL resides.

If index.html resides in the folder www and image.gif resides in the same folder, you could display image.gif in index.html with this:

```
<img src="image.gif">
```

If index.html resides in the folder www and image.gif resides in the folder www/images, you could display image.gif in index.html with:

```
<img src="images/image.gif">
```

Absolute URL

Specifies a file with its full (absolute) URL – http://*server*/*file*

If the full URL of image.gif is http://www.site.com/images/image.gif, no matter which folder index.html resides in, you could display image.gif in index.html with:

```
<img src="http://www.site.com/images/image.gif">
```

Rule of Thumb

Use relative URLs rather than absolute URLs whenever possible.

Relative URLs are generally shorter and easier to manage.

Practice: Links (<a> Elements)

1. Create a set of 5-10 links (<a> elements) to your favorite web sites.

   Tip: You'll need to assign absolute URLs to your href attributes.

2. Locate some HTML files that are stored on your computer. (If you don't have any, grab a few from the web and save them to a work folder you create on your desktop.)

   Create links to these HTML files.

   Tip: You'll need to assign relative URLs to your href attributes.

Fun with Lists

We'll end this module by letting you have some fun making bulleted and numbered lists.

You use the <ol>, <ul>, and <li> elements to make lists of items.

Two flavors of lists:

Ordered (numbered) lists

The <ol> element defines the entire list. The <li> element defines individual items in the list. HTML re-orders the numbers automatically when you add/delete list lines.

For example:

```
<h3>My favorite foods</h3>
<ol>
  <li>Thai</li>
  <li>Vietnamese</li>
  <li>Authentic Mexican</li>
  <li>Indian</li>
</ol>
```

Unordered (bulleted) lists

The <ul> element defines the entire list. The <li> element defines individual items in the list.

For example:

```
<h3>My favorite foods</h3>
<ul>
  <li>Thai</li>
  <li>Vietnamese</li>
  <li>Authentic Mexican</li>
  <li>Indian</li>
</ul>
```

Practice: Ordered and Unordered Lists

1. Add the ordered and unordered list examples above to a web page to see what they look like in a browser.
2. Edit the links you created in the previous Practice task to format them as an ordered list, then as an unordered list.