HyperText Markup Language (HTML) [*Draft 2009-01*]

Introduction to the Hypertext Markup Language (HTML)

This discussion points out major features of the HTML system at a very high level and can be used in conjunction with parallel discussions, descriptions, and examples by seminar participants, instructors, or guides. (A companion document, CSS, will discuss how an HTML document can be *styled* to your typographic specifications).

HTML is a structured *markup* language that allows web publishers and developers to insert structuring and formatting codes (that is, HTML markup "tags") into text documents. Beyond this immediate use, HTML is also a core component of other languages used on the net, such as XHTML.

Let me start with the basics and gradually introduce more detail as I go along. I'll jump right in and show an easy HTML document (showing its underlying 'markup') that I have stored in my local file system. The intent here is just to give you an initial look at an actual HTML file, more detail will follow throughout this manual. Note: by 'markup' I mean the characters starting with a "<"and ending with an ">". So, the characters <html> is an example of 'markup'. These markup characters are usually called "tags". So, the <html> markup is called the starting "html" tag, and its closing tag is denoted by </html>. All of the characters enclosed by an opening and closing tag (and including the tags) is called an element. This is a more abstract description of the character set than you actually see.

An HTML File

Next is shown an HTML file called `garden.html`

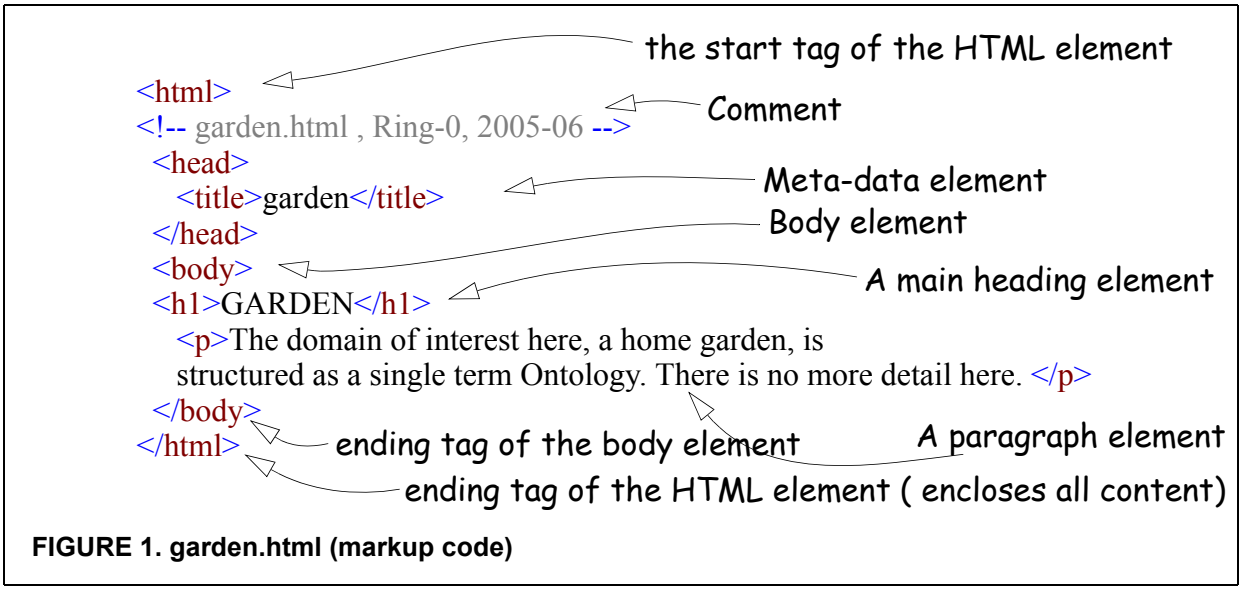


FIGURE 1. garden.html (markup code)

A Discussion of the garden.html markup code

In the Figure 1 on page 1 the markup breaks down like this:

- html - this is the outer element for an HTML file. Everything in the file is enclosed by this starting and ending tag.
- head - this tag encloses the meta-data (data about data). This content doesn't appear in the browser window. Here is where you could tell the browser what kind of characters to expect or to give key-words to web-crawling search engines.
- title - this is a meta-data tag whose content appears as the browser window title. It is a required element in the latest specs for HTML.
- body - this is the element that contains the visible output in the browser window
- h1 - a tag that represents the most important heading. There are 5 other built in headings, h2 through h6.
- p - usually used to introduce paragraph text.

The Rendered Document

In Figure 2 on page 2, the browser rendered output is shown (actually a screen copy from an XML editor program)

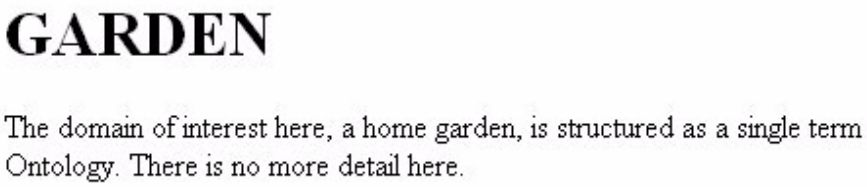


FIGURE 2. garden.html (browser rendering)

More Discussion of the Markup Code for garden.xml

If you want a bit more detail about the code shown in Figure 1 on page 1. read-on. Otherwise, you might want to skip this section and go toward a bigger picture of the role of HTML.

Looking at the first line of the example, the document outer element consists of the initial and concluding markup tags <html> and </html>. You can think of these tags as encircling or enclosing all of the document's content. These tags indicate(to the browser) that the whole document is to be treated as an *HTML* category of document.

The next tag, the <head> tag, begins the meta-data section of the document. Within this tag you can place <title> tags, as is shown here, as well as tags denoting the type of content that the document can hold such as keywords or links to a further styling document like a 'CSS' file. I will have more to say about this 'styling' language in another document. Keep in mind though, that the browser already has a default way to show all of the content of HTML tags and it is the task of this other language CSS, to allow you to override selected defaults.

After the <head> content comes the <body> tag which starts the material that will actually show

up in the final browser window. You will place all of the text you want to have displayed within these start and end tags.

Then comes an <h1> tag. This indicates to the browser that the characters "GARDEN" are to be rendered according to the first level heading style that is browser specific. The browser will usually interpret these “h1, h2, . . . h6” tags to mean that you would like the enclosed text to be set in a bold font, in a bigger size that the surrounding text, and perhaps in a different font family. For example, document designers often set the headings in one font family such as *Arial* while the body of the text is set in another family such as *Times New Roman*. The contrast between these two families is that, *Arial*, is a "sans-serif" font while the *Times New Roman* is a "serif" font. Other heading tags such as the series <h2>, <h3>, down to <h6>, are rendered in successively diminished displays indicating lesser importance. It is the case that, for every HTML tag, all of the modern browsers have a style they will apply to that tag's content, unless you override that style with your own rules. Similarly, the starting tag and ending tags, and direct the browser to set the enclosed text according to the browser's default *paragraph* style. This default style begins a new line on the screen, selects a browser determined basic font and point size, and then renders the text between the tags.

We will see later, in our section on Cascading Style Sheets (CSS), that these default built-in browser specific styles may be overridden by invoking an author constructed file consisting of style rules called a *CSS* stylesheet. In this case, the author could override the usual browser defaults of the color black for the text color and instead, specify another color, say, red. (Making your pages look good is no easy task and here is where the services of a graphics designer are often employed.

How Do These HTML Documents Get to Me?

The marked up documents, called "HTML documents", are usually placed in a standard directory that is known to the user. If the documents are stored on the local file system (that is, your machine), then typing the file address into the browser location field will cause the browser to access that document (see below for subsequent processing). Usually though, you will use the browser’s optional navigation procedure to get to the local file. In other words, if you have some HTML documents on your machine, then you can view them using a browser such as FireFox, or Internet Explorer. This simply requires you to navigate to those documents and open them up by double clicking. Next is shown an example file path that I could type into a browser location field to display the `garden.html` file.

```
file:///C:/aasexxml2005/ring0/garden.html
```

If the document you want is stored remotely, then it will be stored on a "web server" in a well known directory and be available over the internet using the http protocol.

```
http://java.sun.com/reference/techart/index.html
```

When an HTML document, stored in the file system of a web server, is requested by a browser, (usually by typing the address of the server, followed by the name of the desired document, into the browser's location field), the document is delivered to the browser over the net. The data formatting protocol used to deliver this document over the internet is called *HTTP* (Hypertext Trans-

port Protocol).

This protocol, HTTP, is layered on top of the most famous protocol set of all, *TCP/IP*. These are the Transport (also called 'Transmission) Control Protocol and the Internet Protocol respectively. Conforming to the specifications defining these protocols allows packets of data to be sent reliably from your browser to the targeted server and back again. These four technologies may fairly be called the basis of the success of the web. HTML is the simple language, HTTP is a very simple protocol for data formatting, and the other two protocols guarantee end to end reliable transmission over many intermediate routes (“hops”).

Now that the Document is At My Browser, Now What? - The Document Object Model (DOM

Once the browser receives the document, it "parses" it, that is, it reads in the document and separates out text and tags. The result of this separation is an internal model of the document (a tree like structure called the Document Object Model (DOM)) that the browser will use to "render" or display that document. Referring to this internal DOM model, the browser examines each "tag", interprets it's structure, event, or presentation component, and acts accordingly. For example if the browser recognizes a tag that represents the start of a paragraph, that is, a <p> tag, then the browser will end the current flow of text and start a new line on the screen, and start displaying the paragraph content text in the correct font family, size, and color. The browser's 'rendering' engine continues the traversal through the document model tree, rendering text as it goes until it comes to the end, logically that would correspond to the ending <html> tag that you placed at the end of your document. (If, when the browser begins to read your HTML file and it encounters references to the CSS language, it incorporates those over-rides as it is rendering the text.

HTML, Its Glory and its Limits

HyperText Markup Language (HTML) was first presented in 1992, and was used initially to prepare scientific documents for web display. In these early days, the content of the technical papers was most important and the format or presentation was accorded less weight. As time went on though, tens of thousands of people started marking up documents for presentation on the web and since then has steadily expanded its capabilities, its authorship, and its audience.

The Browser Wars

Many of these people now creating HTML documents had come from the desktop publishing world where control over every aspect of a document was assumed, as a matter of course. Now these publishing oriented people were confronted with very limited control over their page layout. This dissatisfaction caused the browser vendors, principally, Netscape and Microsoft, to engage in the "browser wars" during the era, 1995 - 1998. In this period, each browser manufacturer competed with the others to offer new or enhanced markup tags to allow more control over document presentation or manipulation. Unfortunately, these new additions were usually incompatible with other browser's rendering engines. This situation caused untold frustration among developers who would have to write tags at the "lowest common denominator" so that their pages would look similar on different browsers. I was coding web pages during that period and it was a major pain to

write HTML files that would display properly on each browser.

Evolution of HTML

To give a bit of context to the evolution of HTML, we need to look back into the 'printing' past, or actually from the advent of writing. Surely, the first people learning to write, would make mistakes, and so the rise of *proofreaders* and *editors!* Later, when some people began to direct the presentation of people's writing, they would need to indicate their instructions either verbally or on the medium itself. *Mark-up!*. But more seriously, paper-based "mark-up" languages have a history stretching back at least to the 14^{th} century in the West, and well before that in South East Asia and Africa. Mark-up languages were, and are, used to give directions, usually by making additional notes on the manuscript itself, for the processing of that manuscript or other textual material, through additional phases of the printing process. (Note that today, markup is also used to direct additional processing of text that may not be targeted to print, but rather to all manner of multi-media representations, aural, visual, transport protocols, or even executable code.)

The Rise of Typography

To spend a little more time on the "old days", an author would submit a typed or handwritten manuscript to an editor, who would mark on the manuscript his corrections plus headings, paragraphs, margins and other typographic events, in a process called *copyediting*. This marked-up manuscript would then be given to the printer so that he could select the proper typographic elements, such as the actual ink, paper, and lead characters to be used as well as other spacing materials that would render the document in accordance with the mark-up. That typography markup vocabulary survived, evolved, and is still used today.

Hand Markup by Encircling Pieces of Text

The basic idea, which is still relevant today, is that the hand markings indicate which text is to be rendered, in what particular manner. The manuscript editor would sometimes literally encircle sections of text to show the target of associated instructions. This idea of "encircling" the text and linking associated instructions for what is to be done with it, carries over to the HTML markup approach. When you look at the document as marked up by the HTML tags, you will see that these "tags" perform the same function as encircling the text by hand and the names of the tags, together with their associated rules, indicate the instructions to be performed. The tags I am talking about are the pieces of text between angle brackets such as <h1> and </h1> and the text enclosed is called the *content*.

Today's Desktop Publishing Systems

Today's desktop publishing programs do the same thing by inserting formatting (mark-up) code in your text, but in a transparent manner, rendering the result in real time in a mode called: "what you see is what you get" (WSYWIG). In modern publishing software, the user has a number of prespecified formats to choose from such as a *Book* pattern or a *Report* template. Within these "templates" there are numerous preconfigured formatting 'tags' one can choose just by clicking. The following text is then rendered according to the typographic definitions associated with these

tags. Usually, you don't need to see these tags since their effect is shown immediately, allowing immediate revision as necessary.

The mark-up tags of HTML are a little different since they are visible in the text when you enter them and second, you don't see their effect until you view that document within a browser. In a way, this is a step backwards since physically inserting a start tag <h1> or a paragraph start tag <p> directly into the text is actually closer to the old time printers mark-up approach since they too were *explicitly* inserted within a manuscript (document)! The plus side of all this is that the HTML tags are very lean and efficient and don't carry the incredible bloated code baggage that a page of RFT (rich text format), or a Word document requires. It is the simplicity and efficiency of HTML that guaranteed its success in the early days of slow dial-up modems and very limited bandwidth.

Recent Technical Origins of HTML - SGML

Technically, HTML is an *application* of a more abstract language, one that is designed to guide the *production* of markup languages. This type of language is called a *meta-language*. You could think of this meta-language as providing the rules or grammar, for a chosen set of vocabulary elements. The abstract language chosen to produce HTML is known as Standard Generalized Markup Language (SGML) and HTML is an application of SGML in the sense that HTML consists of about 100 standardized tags that follow the SGML syntax directives. SGML is an international standard as I discuss below.

SGML is an International Standard

SGML became a world wide standard in 1986 and has a very long and distinguished history as a meta-language. SGML has been used to produce many other languages, called *applications*. Since its introduction in the late mid 1980's, application vocabularies have been constructed for such fields as law, medicine, and chemistry. Originally, SGML was designed to cope with the publishing needs of extremely large scale document sets, consisting perhaps of millions of pages. This sort of requirement happens in documenting a nuclear regulatory facility, a modern weapons system such as an aircraft carrier, or perhaps just a few million pages for the maintenance manuals for a fleet of Boeing 767's.

This SGML language is a *meta language*, the grammar that must be followed when a domain specific language is to be constructed. From this perspective then, SGML is a language factory used to construct languages, with HTML being one of these constructed languages. This construction consists of a set of *tags*, the vocabulary of the language, and rules (grammar rules) for how these tags can appear in a document and how they are to be interpreted.

Additional Notes and Insights into HTML

HTML is simple, simple, simple. That simplicity fueled the explosive creativity of the web and is one of the most appealing features of HTML. The simplicity is due to the few number of HTML tags, around 100, and their fixed, non-extensible nature (but there is a work around using two tags called “div” and “span”). Originally designed to allow the interchange of scientific documents

where presentation was secondary to content, HTML was designed to be the simplest possible structuring language, with the original version not even supporting graphics. This simplicity of HTML allowed web page production by people with no need for computer science training or even software experience. A literal explosion of web pages resulted and is still ongoing.

HTML Has Mixed Content with Presentation (Big Time in the Past)

Ar limiting feature of HTML that begins to show up when you deal with hundreds of pages, was that the structure of the information, denoted by such elements as *headings* and *paragraphs* were mixed in with presentation features such as specifying *font families and styles* within the same document. Similarly, other elements that allowed *italic* or *bold* text to be specified are part of the HTML vocabulary. This mixing of element types unfortunately violates a fundamental design principle, namely, keep the *content* separate from the *presentation*. Some help was definitely needed and the World Wide Web Consortium (W3C) stepped in, took over the maintenance of HTML, and published unifying standards for HTML and its successors (most notably XHTML).

The Way Forward!

In 1999, the W3C published a new specification, called XHTML 1.0, that was intended to recast the HTML language in terms of another meta-language called Extensible Markup Language (XML). This specification is the *next generation* of HTML and is now the official direction going forward. There will be no new HTML specifications, all that content has been rolled into XHTML. So instead of HTML continuing to evolve under the framework of SGML, a new formulation was specified with the HTML version 4.01 providing a core set of tags but within an XML framework that will allow additional tags and elements to be introduced as needed by designers, in a controlled modular fashion.However, having said that, since there are millions of pages of HTML out there on the web, it’s not going away for many years.

Structure of an HTML Document

We can learn a little more about HTML's " layout" by looking at the diagram again. All HTML document have this same structure, namely:

1. !DOCTYPE validator (optional). You can specify to the browser that you want the browser to perform some validity checks on your document so the reader won’t get some unpleasant surprises.
2. Comments (optional)
3. Document Root Element (html tag)
4. Head Section
5. Body Section

Another Structural Perspective on HTML Documents

The diagram below shows two more ways to look at the structure of HTML (and XHTML for that matter) documents. The intent here is to show that one can think of an HTML document as a set of nested boxes, or as a tree.

Nested Box Viewpoint

On the left diagram you see an overall box labeled "document". This represents the overall document (box) that can contain other boxes representing nodes and elements. Within the outermost box we can have non-content entities such as Comments, Document Type Definitions (DTDs), and Processing Instructions. These may occur prior to the actual content of the document. The actual content of the document begins with the *html* element. This particular element is called the *Root Element* since all the content is encapsulated by it. Within that root element we see various other elements enclosed. Each such element can, in turn, enclose "child" elements. It is one of the roles of a DTD to allow or disallow different types of element enclosures. That is, a paragraph element *p* is allowed to contain an *em* (emphasis) element but an *em* element is not allowed to enclose a paragraph element. This nested box view is very helpful both in thinking about how an (X)HTML document is laid out as well as how CSS applies to such elements. That is, we will see later that CSS uses this " box " viewpoint to apply style rules.

garden.html as a Box Layout

Recall the garden.html file in Figure 1 on page 1. Let me draw this out in a different way that emphasizes the *nesting* idea of elements.

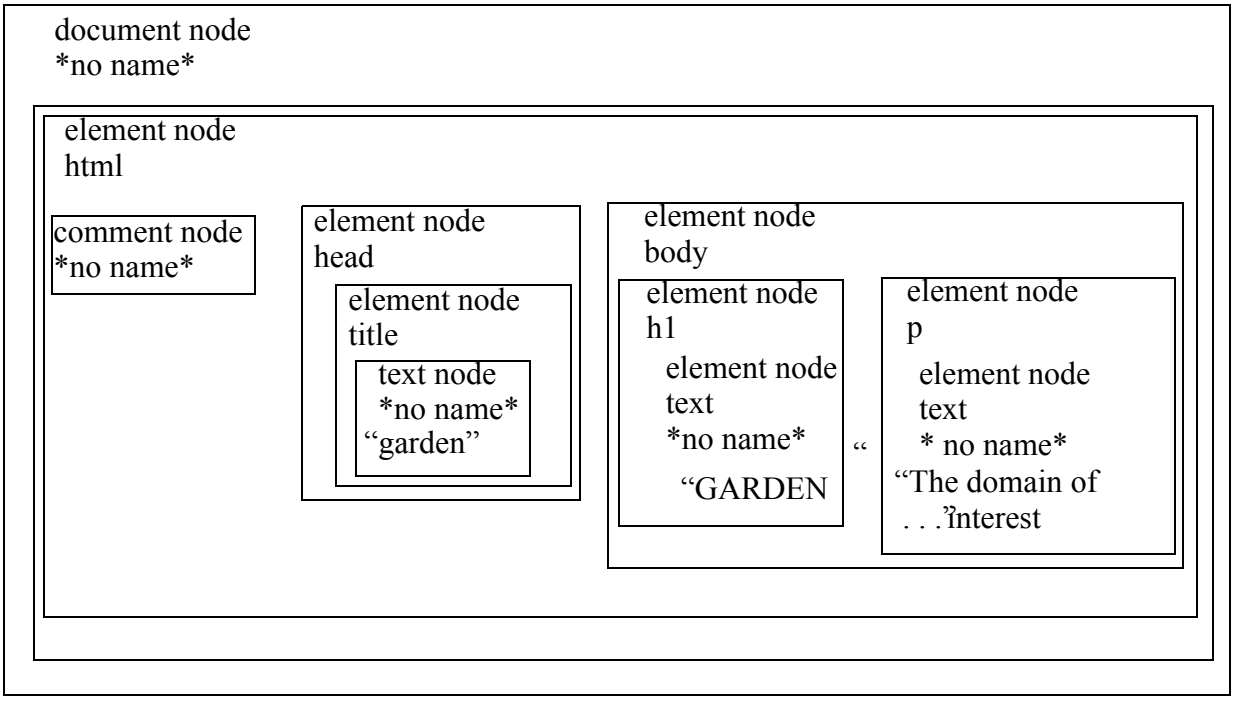


FIGURE 3. Box Model of HTML

The Tree View

An alternative view is to think the document as a (upside down!) tree, with the *overall document root* as its root. Descending from this root, we see the Comment node and then the *root element* which is the *html* element. This view of an (X)HTML document as a tree with branches at various levels is useful conceptually as well as guiding search algorithms and processing implementations. Here is the markup code again:

```
<html>
<!-- garden.html , Ring-0, 2005-06 -->
 <head>
   <title>garden </title>
 </head>
 <body>
 <h1>GARDEN</h1>
   <p>The domain of interest here, a home garden, is
   structured as a single term Ontology. There is no more detail here. </p>
 </body>
</html>
```

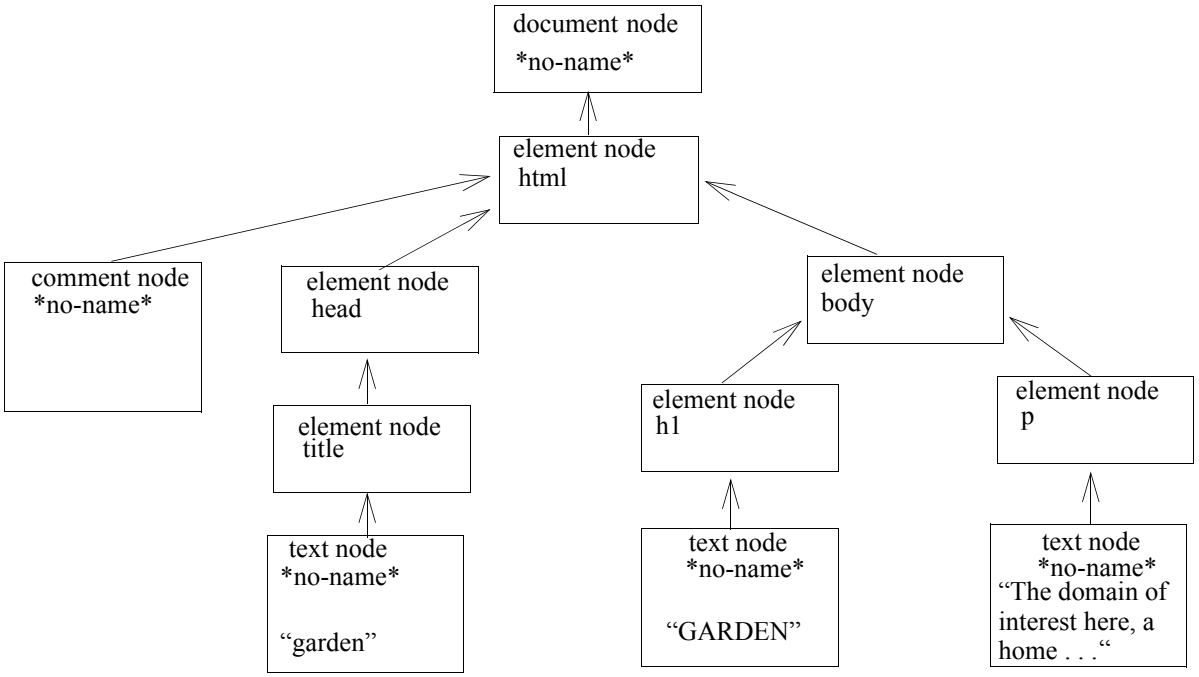


FIGURE 4. Tree View of garden.html

The Basic HTML Tag Set

The actual vocabulary of HTML itself, consists of a domain tag set originally developed by Tim Brenners-Lee. This tag set is the one you are familiar with (or becoming familiar with) that has the well known heading tags such as `<h1>` or `<h2>` or the paragraph tag `<p>`. (If you check out the source code of this document, you will see these tags in action!). In fact you can see the (basic) tag set for this language at the end of this document.

Comparing Tags and Elements

Up to this point I have been pretty loose in talking about *tags* and *elements*. This next diagram show the connection between these two ideas. Think of the *element* as the conceptual entity that is

realized by marking off its realization within an actual document by using markers, that is, *tags*.

The above diagram shows that elements are made up of alphanumeric tokens within angle brackets. The above element is the *anchor* element that is represented by the *a* tag. Note that an element can have *attributes* as well. These serve to modify the element. In the example, the attribute represents the location from where an HTML file may be fetched. There are two parts to an attribute, a *name*, in this case "href", and a *value* which in this case is "htmlreference.html".

Elements Can Have Modifiers Called *Attributes*

There is much more to the HTML story than we have time for here, but I must discuss the role of *attributes* within *elements*. The purpose of attributes is to modify or add to, the base information carried by the element. For example, one of the tags/elements we will study is called an *anchor* tag and denoted by *a*. (You can see the structure of this element and its attribute(s) in the "(X)HTML Elements and Tags " diagram above). This tag is used to specify where to find resources to bring back or to jump to a named location. An example use of this element and its attributes is to reference the HTML tag set used in this course.Here is what the source code looks like:

```
<a href="../referencematerial/htmlbasicreference.html" > HTML Tags </a>
```

Core HTML Attributes

The HTML 4.01 spec, has four main attributes that are available to be inserted within almost every tag:

1. class - This attribute is used to categorize sets of elements so that a rule in a style sheet, for example, could apply to otherwise different elements whose only common feature was an attribute with the same *class* value. (This will become clear when we study CSS stylesheet rules.)

   ```
   <p class="warning" > A paragraph element categorized as a "warning "
   paragraph</p>
   <p class="warning" > A second paragraph element categorized as a "warning
   " paragraph</p>
   <h6 class="warning" > A sixth level head is also categorized as a "warn-
   ing " element</p>
   ```

   Using this "warning" categorization, a designer could specify, in a style sheet, that all "warning > elements should be rendered in,say, *red* colored text.

2. id - This attribute is intended to specify a unique alphanumeric identifier for an element. This identifier picks out a particular element to be cross-referenced from a style sheet, a link, or a scripting routine (such as JavaScript). Note: in the next generation XHTML documents, this *id* attribute will replace the *name* identifier currently used in HTML 4.01.

   ```
   <a href="quigley">Jump to the (unique) Quigley paragraph </a>.
   • • •
   <p id="quigley" > A uniquely identified paragraph element that can be
   jumped to as in this example, or targeted from a style rule within a CSS
   file. </p>
   ```

   This paragraph element could be the target of a style rule, or a destination for an anchor

3. style - This allows the designer to "inline" styling tasks associated with an element. This use is discouraged however, as it defeats the desired goal of keeping content separate from presentation. Use external style sheets instead. (This is what we routinely do in this course.)

```
<p style="color:red; font-size:16pt "> Here is a direct way to set this
paragraph's text color to red and its text point size to 16. (Note: there
are 72 points per inch )</p>
```

4. title - This attribute allows the designer to supply explanatory text that might be rendered in the browser as a *tool tip* when the pointing device is over it or touches it. This attribute is commonly used within the *abbreviation* and *acronym* tags.

```
<acronym title="Extensible Markup Language">XML </acronym>.
```

   In this example, the browser will display the value of the *title* attribute, when the mouse is over the "XML" text.

5. lang - This attribute describes the language to be used for the element content.(See

```
<p lang="fr">Le plum et sur le table </p>.
```

   This markup asks the browser to display the enclosed text in its French mode.

6. dir - This is currently little used, but as the need for internationalization increases, account must be taken of languages that are read from right to left rather than from left to right. The allowable values are *ltr* (left to right) and *rtl* (right to left). This attribute is used with the *bdo* (bi-directional override) element.

```
<bdo dir="rtl">Able was I ere I saw Elba </bdo>.
```

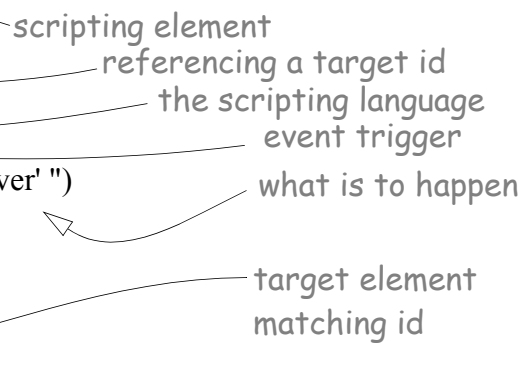
HTML Events

There are a core set of events associated with most of the HTML elements. These are the kinds of events that are triggered by a mouse click, drag, hover, or a key press / release. They can also be the consequence of a browser about to load a page, or unload one. To see these and find out what they do, check your HTML site at the `www.w3c.org`. The code below shows an example of event handling. This code is followed by an analysis of what happened here.

When your mouse hovers over this <p> text, an alert dialog box pops up. If you look up in the "head" section of the source code of this document, you will see a script with an attribute named for="palert". That is the script that is invoked when you place your mouse over any of the <p> text. Place your mouse over this <p> text to see an alert box pop up.

A Scripting Example

```
<html>
<!-- gardenEvent.html , Ring-0, 2005-06 -->
 <head>
   <title>garden</title>
   <script for="palert"
   language="JavaScript"
   event="onmouseover">
   alert(" This is an HTML Event,'onmouseover' ")
   </script>
 </head>
 <body>
 <h1>GARDEN</h1>
   <p id="palert">
        The domain of interest here, a home garden, is
</body> structured as a single term Ontology. There is no more detail here.
</html>
```



So What Happened with the Event Test Above?

You just saw an example use of the 'onmouseover' event above, where the browser software informs your HTML program that a *mouseover* event has occurred over an <p> HTML element. That is, a mouse has been positioned over some part of the text enclosed by that specified tag. Note that this course will not cover scripting in any detail since it is being supplanted by other technologies such as Java Server Pages and server side capabilities, but to give you an idea of what you can currently do lets examine the actual script. The script and its target, an HTML <p> element that is the target of that script are shown below. Note that the event code is in two places. First, up in the <head> section, I place the scripting code. Second, down in the body of the document, I am going to target that script code from an <p> tag by placing an "id" attribute that the script will reference. The result is that when the user positions the mouse over any part of this particular "p" text, the script code is invoked, which in this case causes an "alert" dialog box to appear.

Linking CSS to HTML, XHTML, and XML Documents

The linkage between an HTML, XHTML, or XML document and a CSS stylesheet is effected by a *link* element that is inserted within the *head* section of an (X)HTML document. The structure of this *link* is shown in the following example:

```
<link href="mystyle.css" rel="stylesheet" type="text/css" >
```

The actual vocabulary of HTML itself, consists of a domain tag set originally developed by Tim Brenners-Lee. This tag set is the one you are familiar with (or becoming familiar with) that has

the well known heading tags such as <h1> or <h2> or the paragraph tag <p>. (If you check out the source code of this document, you will see these tags in action!).

Basic (X)HTML Sorted by Element

Element Type Name	Module Category	Description
a	Hypertext	This element provides a linkage to other parts of the same document or to a different document.
abbr	Text	This element lets authors indicate a sequence of characters that comprise an abbreviation.
acronym	Text	This element lets authors indicate a sequence of characters that comprise an acronym.
address	Text	This element indicates ownership or authorship of a document or document fragment. It is usually placed at the beginning or at the end of the document or document fragment.
base	Base	This element specifies the base URL to use for all relative URL appearing within the document. It must be contained within the head element.
blockquote	Text	This element shows an extended quotation. Usually this is rendered by indentation.
body	Structure	This is the content section of the HTML document
br	Text	This element forces a line break.
caption	Basic Tables	This element defines a caption within a table (can also be used with figures).
cite	Text	This element indicates a citation from other material. It is usually rendered in italic text.
code	Text	This element indicates that the text is to be interpreted as source code in some programming language. It is usually rendered in a monospaced font.
dd	List	Definition list entry - the definition of the term
dfn	Text	This element indicates the defining instance of a term. It is usually rendered as bold or italic text.
div	Text	This element is designed to allow authors to designate blocks of text to be treated as a logical unit. See "span".
dl	List	Definition list element (contains dt and dl entries)

dt	List	Definition list entry - the term to be defined
em	Text	This element indicates emphasized text. It is usually rendered as italic text.
form	Basic Forms	This element defines a form that is presented to the user. The form's structure contains sub-elements such as "labels" and "form controls". See the other "Basic Forms" elements.
h1, h2, h3, h4, h5, h6	Text	These elements, h1 through h6, are headings. These tags implement 6 graded levels of headings, with h1 being the most prominent and h6 being the least prominent. These are usually rendered as bold text in a point size larger than the body text point size.
head	Structure	This is the "head" or meta-data section of the HTML document
html	Structure	This is the overall container for an HTML document
img	Image	This element indicates a media object to include in an (X)HTML document. The object is usually a graphics image like a "tif" or "jpeg" file, but may be video or other types of animations, depending on the browser implementation.
input	Basic Forms	This element is a "form control". This element is an input control and can be a variety of types such as: single line text field, multiline text field, password style, check box, radio button, or push button.
kbd	Text	This element indicates text that would have been input from a keyboard. It is usually rendered as monospaced text.
label	Basic Forms	This element relates text descriptions to form controls.
li	List	List item (contained by ol or ul element)
link	Link	This element specifies relationships between the current document and other documents. It is often used to specify an external stylesheet or to define a navigational frame to move from one related document to another.
meta	Metainformation	This element specifies information about the document, such as content type, or author and version, or keywords for use by search engines.
object	Object	This element allows an arbitrary element to be included within an (X)HTML document. The object can be an image, applet, ActiveX control, media object, or even another document.
ol	List	Ordered list element (contains li entries)
option	Basic Forms	This element specifies an item in a selection list. This element is used in conjunction with the "select" element.

p	Text	This element indicates a paragraph of text. It is usually rendered with a blank line before and after.
param	Object	This element specifies a parameter to pass to an embedded object that uses the "object" tag.
pre	Text	This element indicates "preformatted" text. This means that formatting characters are preserved such as spaces, returns, and tabs. Often used to render poetry where spacing and other formatting should be preserved.
q	Text	This element indicates a short inline quote. It is usually rendered within quote marks. Compare with blockquote
samp	Text	This element is used to indicate sample text. It is usually rendered in a monospaced text.
select	Basic Forms	This element is a selection list within a form. This control allows the user to select one or more list options, depending on the format of the selection list.
span	Text	This element is designed to allow authors to designate groupings of inline text to be treated as a logical unit. See "div".
strong	Text	This element indicates strongly emphasized text. It is usually rendered by bold text in a visual context but can be used to indicate emphasis in other modalities such as aural.
table	Basic Tables	This element defines a table
td	Basic Tables	This element specifies a data cell within a table. This element is contained in a "tr" table row.
textarea	Basic Forms	This element specifies a multiline text input field specified within a form.
th	Basic Tables	This element specifies a header cell in a table. This element should be contained in a "tr" table row tag.
title	Structure	This is a document identifier that appears on the Browser title bar. This element is contained within the "head" section.
tr	Basic Tables	This element specifies a row in a table. The cells of the row are specified by the "tr" or "th" elements.
ul	List	Unordered list element (contains li entries)

var	Text	This element indicates a variable. Variables are usually associated with identifiers that occur in programming or mathematical languages. It us often rendered in italic text.

Element Type Name	Module Category	Description

Basic (X)HTML Sorted by Module

Element Type Name	Module Category	Description
base	Base	This element specifies the base URL to use for all relative URL appearing within the document. It must be contained within the head element.
form	Basic Forms	This element defines a form that is presented to the user. The form's structure contains sub-elements such as "labels" and "form controls". See the other "Basic Forms" elements.
input	Basic Forms	This element is a "form control". This element is an input control and can be a variety of types such as: single line text field, multiline text field, password style, check box, radio button, or push button.
label	Basic Forms	This element relates text descriptions to form controls.
option	Basic Forms	This element specifies an item in a selection list. This element is used in conjunction with the "select" element.
select	Basic Forms	This element is a selection list within a form. This control allows the user to select one or more list options, depending on the format of the selection list.
textarea	Basic Forms	This element specifies a multiline text input field specified within a form.
caption	Basic Tables	This element defines a caption within a table (can also be used with figures).
table	Basic Tables	This element defines a table

td	Basic Tables	This element specifies a data cell within a table. This element is contained in a "tr" table row.
th	Basic Tables	This element specifies a header cell in a table. This element should be contained in a "tr" table row tag.
tr	Basic Tables	This element specifies a row in a table. The cells of the row are specified by the "tr" or "th" elements.
a	Hypertext	This element provides a linkage to other parts of the same document or to a different document.
img	Image	This element indicates a media object to include in an (X)HTML document. The object is usually a graphics image like a "tif" or "jpeg" file, but may be video or other types of animations, depending on the browser implementation.
link	Link	This element specifies relationships between the current document and other documents. It is often used to specify an external stylesheet or to define a navigational frame to move from one related document to another.
dd	List	Definition list entry - the definition of the term
dl	List	Definition list element (contains dt and dl entries)
dt	List	Definition list entry - the term to be defined
li	List	List item (contained by ol or ul element)
ul	List	Unordered list element (contains li entries)
ol	List	Ordered list element (contains li entries)
meta	Metainformation	This element specifies information about the document, such as content type, or author and version, or keywords for use by search engines.
object	Object	This element allows an arbitrary element to be included within an (X)HTML document. The object can be an image, applet, ActiveX control, media object, or even another document.
param	Object	This element specifies a parameter to pass to an embedded object that uses the "object" tag.
body	Structure	This is the content section of the HTML document
head	Structure	This is the "head" or meta-data section of the HTML document
html	Structure	This is the overall container for an HTML document

title	Structure	This is a document identifier that appears on the Browser title bar. This element is contained within the "head" section.
abbr	Text	This element lets authors indicate a sequence of characters that comprise an abbreviation.
acronym	Text	This element lets authors indicate a sequence of characters that comprise an acronym.
address	Text	This element indicates ownership or authorship of a document or document fragment. It is usually placed at the beginning or at the end of the document or document fragment.
blockquote	Text	This element shows an extended quotation. Usually this is rendered by indentation.
br	Text	This element forces a line break.
cite	Text	This element indicates a citation from other material. It is usually rendered in italic text.
code	Text	This element indicates that the text is to be interpreted as source code in some programming language. It is usually rendered in a monospaced font.
dfn	Text	This element indicates the defining instance of a term. It is usually rendered as bold or italic text.
div	Text	This element is designed to allow authors to designate blocks of text to be treated as a logical unit. See "span".
em	Text	This element indicates emphasized text. It is usually rendered as italic text.
h1, h2, h3, h4, h5, h6	Text	These elements, h1 through h6, are headings. These tags implement 6 graded levels of headings, with h1 being the most prominent and h6 being the least prominent. These are usually rendered as bold text in a point size larger than the body text point size.
kbd	Text	This element indicates text that would have been input from a keyboard. It is usually rendered as monospaced text.
p	Text	This element indicates a paragraph of text. It is usually rendered with a blank line before and after.
pre	Text	This element indicates "preformatted" text. This means that formatting characters are preserved such as spaces, returns, and tabs. Often used to render poetry where spacing and other formatting should be preserved.
q	Text	This element indicates a short inline quote. It is usually rendered within quote marks. Compare with blockquote
samp	Text	This element is used to indicate sample text. It is usually rendered in a monospaced text.
span	Text	This element is designed to allow authors to designate groupings of inline text to be treated as a logical unit. See "div".

strong	Text	This element indicates strongly emphasized text. It is usually rendered by bold text in a visual context but can be used to indicate emphasis in other modalities such as aural.
var	Text	This element indicates a variable. Variables are usually associated with identifiers that occur in programming or mathematical languages. It us often rendered in italic text.

Element Type Name	**Module Category**	**Description**

Summary

This basic tag set will be expanded as we proceed in learning more about HTML and CSS.