

The LINGUIST List Tutorial on Basic HTML

1. HTML and Tags: The beginning

HTML (Hypertext Markup Language) is an authoring language that is added to basic text documents to control the layout and format of the document content as viewed on the web. The language is made up of **tags**—markers, enclosed in brackets (<>), which indicate how text or graphics should appear.

Tags are generally used in pairs, and the closing tag is always preceded by a forward slash (/). Here is an example of a line of text which includes a tag:

```
<TAG>This is the content/text that the tag is acting on.</TAG>
```

There are some tags which are “unpaired.” The
 tag is an example of this. Other tags do not **require** a “closing” tag, and many browsers will interpret the code correctly even if this optional closing tag is missing. However, best practices in programming dictate that you close all appropriate tags which are paired.

HTML is not case-sensitive. However, some prefer to write HTML tags in all-caps, so as to see them better when reading code.

2. Tag Attributes: What can Tags do for you?

Many tags can be further defined by including attributes. **Attributes** contain information about the characteristics the tag should take on. For example, you can define a section of text as a paragraph using the <P> tag. Then, you can assign certain attributes (alignment is a good example) within the tag. The syntax of this is as follows:

HTML code: <P>This is a normal paragraph.</P>
 <P ALIGN="center">This is a centered paragraph.</P>

Display:

This is a normal paragraph.

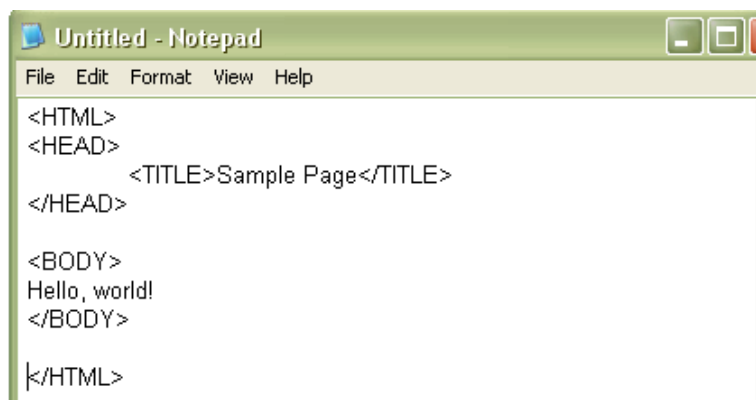
This is a centered paragraph.

Thus, the command ALIGN="center" is included within the brackets of the opening <P> tag, and defines certain characteristics of that paragraph.

It is of interest to note that **not all attributes work with all tags**, and that **all attributes have “default” values**. Therefore, if you do not specify where to align a paragraph, most web browsers will automatically align it at the left.

3. Building the Page: The Bare Necessities

All web pages must have three tags to help define the basic structure of the page: <HTML>, <HEAD>, and <BODY>. They must be “nested” correctly as shown below (we’ll talk more about nesting later):



```
Untitled - Notepad
File Edit Format View Help
<HTML>
<HEAD>
  <TITLE>Sample Page</TITLE>
</HEAD>
<BODY>
Hello, world!
</BODY>
</HTML>
```

The <HTML> tag signals to the browser that this is an HTML page. The <HEAD> tags surround information, often **metainformation**, or information about the page, and the <BODY> tags surround the page content.

In addition to these three tags, there are a great many other tags (with attendant attributes) which can be used to create links, display images, and format text.

Formatting Text: Everybody Get in Line

Let's begin with a typical page, with the three required tags, a title, and a few paragraphs of text:

```
<HTML>
<HEAD>
  <TITLE>LINGUIST List History</TITLE>
</HEAD>

<BODY>

<P>The LINGUIST List is dedicated to providing information on language and language analysis, and to providing the discipline of linguistics with the infrastructure necessary to function in the digital world. LINGUIST maintains a web-site with over 2000 pages and runs a mailing list with over 22,542 subscribers worldwide. LINGUIST also hosts searchable archives of over 100 other linguistic mailing lists and runs research projects which develop tools for the field, e.g., a peer-reviewed database of language and language-family information, and recommendations of best practice for digitizing endangered languages data.</P>

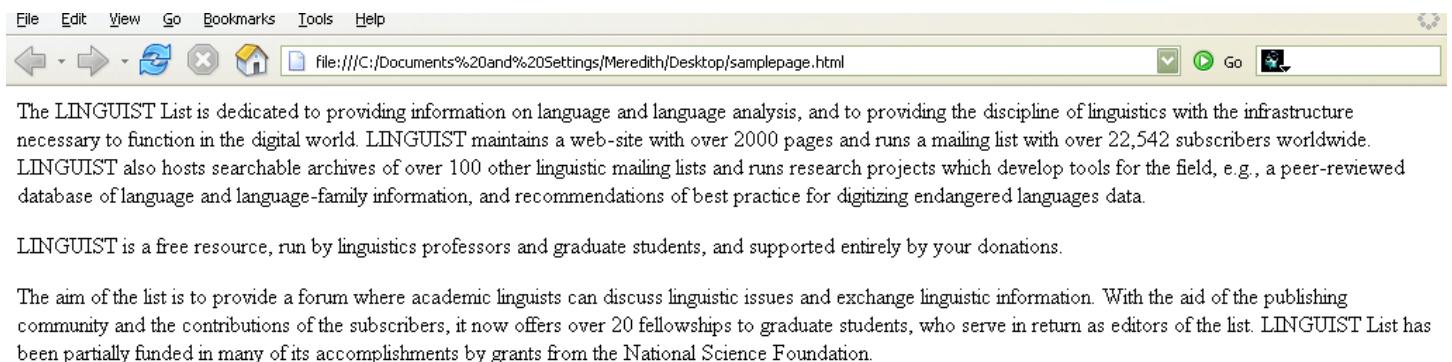
<P>LINGUIST is a free resource, run by linguistics professors and graduate students, and supported entirely by your donations.</P>

<P>The aim of the list is to provide a forum where academic linguists can discuss linguistic issues and exchange linguistic information. With the aid of the publishing community and the contributions of the subscribers, it now offers over 20 fellowships to graduate students, who serve in return as editors of the list. LINGUIST List has been partially funded in many of its accomplishments by grants from the National Science Foundation.</P>

<P>The LINGUIST List also offers access to the following resources: A worldwide directory of linguists; lists of language resources; Ask-a-Linguist and NoticeBoard facilities, a Job Board, and comprehensive Conference listings.</P>

</BODY>
</HTML>
```

On the actual web page, this appears as follows:



We might decide that this is too close to the top of the page. So, we insert a
 tag into the code to space it down:

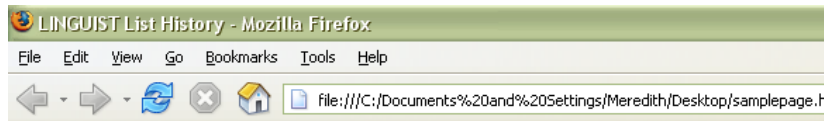
```

<HTML>
<HEAD>
  <TITLE>LINGUIST List History</TITLE>
</HEAD>

<BODY>
<BR>
<P>The LINGUIST List is dedicated to providing information on language and language analysis, and to
providing the discipline of linguistics with the infrastructure necessary to function in the digital world.

```

This gives us the following display:



The LINGUIST List is dedicated to providing information on language and language analysis, as necessary to function in the digital world. LINGUIST maintains a web-site with over 2000 page LINGUIST also hosts searchable archives of over 100 other linguistic mailing lists and runs rese database of language and language-family information, and recommendations of best practice fo

LINGUIST is a free resource, run by linguistics professors and graduate students, and supporte

Note the space between the browser toolbar and the beginning of the words. Next, we might decide to add some words in bold font, or indent a portion of the text. The tags used for this are and <BLOCKQUOTE>, respectively:

```

<BODY>

<P>The LINGUIST List is dedicated to providing information on language and language analysis, and to
providing the discipline of linguistics with the infrastructure necessary to function in the digital world.
LINGUIST maintains a web-site with over 2000 pages and runs a mailing list with over 22,542
subscribers worldwide. LINGUIST also hosts searchable archives of over 100 other linguistic mailing lists
and runs research projects which develop tools for the field, e.g., a peer-reviewed database of language
and language-family information, and recommendations of best practice for digitizing endangered
languages data.</P>

<BLOCKQUOTE>LINGUIST is a free resource, run by linguistics professors and graduate students, and
supported entirely by your donations.</BLOCKQUOTE>

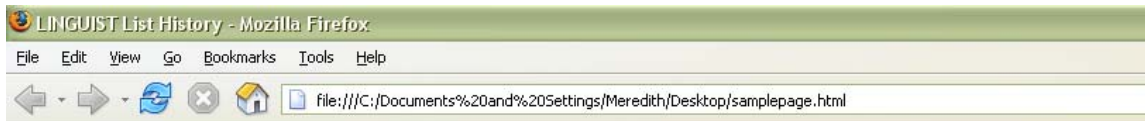
<P>The aim of the list is to provide a forum where academic linguists can discuss linguistic issues and
exchange linguistic information. With the aid of the publishing community and the contributions of the
subscribers, <STRONG>it now offers over 20 fellowships to graduate students</STRONG>, who
serve in return as editors of the list. LINGUIST List has been partially funded in many of its
accomplishments by grants from the National Science Foundation.</P>

<P>The LINGUIST List also offers access to the following resources: A worldwide directory of linguists;
lists of language resources; Ask-a-Linguist and NoticeBoard facilities, a Job Board, and comprehensive
Conference listings.</P>

</BODY>

```

Let's look at what this does to our text:



The LINGUIST List is dedicated to providing information on language and language analysis, and to providing the discipline of linguistics necessary to function in the digital world. LINGUIST maintains a web-site with over 2000 pages and runs a mailing list with over 22,54 LINGUIST also hosts searchable archives of over 100 other linguistic mailing lists and runs research projects which develop tools for the database of language and language-family information, and recommendations of best practice for digitizing endangered languages data.

LINGUIST is a free resource, run by linguistics professors and graduate students, and supported entirely by your donations.

The aim of the list is to provide a forum where academic linguists can discuss linguistic issues and exchange linguistic information. With the community and the contributions of the subscribers, **it now offers over 20 fellowships to graduate students**, who serve in return as List has been partially funded in many of its accomplishments by grants from the National Science Foundation.

Finally, let's take the last paragraph and change the information so it appears as a list:

<BODY>

<P>The LINGUIST List is dedicated to providing information on language and language analysis, and to providing the discipline of linguistics with the infrastructure necessary to function in the digital world. LINGUIST maintains a web-site with over 2000 pages and runs a mailing list with over 22,542 subscribers worldwide. LINGUIST also hosts searchable archives of over 100 other linguistic mailing lists and runs research projects which develop tools for the field, e.g., a peer-reviewed database of language and language-family information, and recommendations of best practice for digitizing endangered languages data.</P>

<P>LINGUIST is a free resource, run by linguistics professors and graduate students, and supported entirely by your donations.</P>

<P>The aim of the list is to provide a forum where academic linguists can discuss linguistic issues and exchange linguistic information. With the aid of the publishing community and the contributions of the subscribers, it now offers over 20 fellowships to graduate students, who serve in return as editors of the list. LINGUIST List has been partially funded in many of its accomplishments by grants from the National Science Foundation.</P>

<P>The LINGUIST List also offers access to the following resources:</P>

- A worldwide directory of linguists;
- Lists of language resources;
- Ask-a-Linguist and NoticeBoard facilities;
- A Job Board; and
- Comprehensive Conference listings

</BODY>

This code gives us the following result:

database of language and language-family information, and recommendations of best practice for digitizing endangered language

LINGUIST is a free resource, run by linguistics professors and graduate students, and supported entirely by your donation

The aim of the list is to provide a forum where academic linguists can discuss linguistic issues and exchange linguistic information community and the contributions of the subscribers, **it now offers over 20 fellowships to graduate students**, who serve in return as List has been partially funded in many of its accomplishments by grants from the National Science Foundation.

The LINGUIST List also offers access to the following resources:

- ◆ A worldwide directory of linguists;
- ◆ Lists of language resources;
- ◆ Ask-a-Linguist and NoticeBoard facilities;
- ◆ A Job Board; and
- ◆ Comprehensive Conference listings

This is only a small portion of the formatting you can do with HTML. A partial but comprehensive list of tags will be included in Appendix A of this handout. The next thing we often want to do is add images and links to our web pages.

Using Images: Let's make the page pretty

The tag is used to embed (insert) an image onto your web page. This tag is “unpaired”, which means it requires no closing tag. The syntax for inserting an image is as follows:

```
<IMG SRC="images/apple.gif" ALIGN="center" ALT="Granny Smith Apple">
```

The first attribute, “SRC”, identifies the file path where the image can be found. If the image is in the same directory as the web page, you do not need a full URL (i.e., <http://www.linguistlist.org>...), but instead you can use a **relative** link. We'll discuss the difference between relative links and **absolute** links in the next section.

The second attribute is the same one we used with our <P> tag—it describes the alignment, or placement, of the image on a page. The final attribute, “ALT”, gives the browser a text description of the image to display, in the case that the image can't be found or loaded.

You can do a number of quite interesting things with images; however, many of these HTML tasks are beyond the scope of a beginner course. The following is a complete list, with descriptions, of the available attributes for the IMG tag:

SRC	Required. The URL of the image file.
ALIGN	Values include: top, bottom, middle, left, or right.
HSPACE	Specifies the amount of space to be left between the image and surrounding text or content, on the horizontal axis (left or right).
VSPACE	The same as HSPACE on the other axis (top and bottom).
HEIGHT	Specifies the height of the image.
WIDTH	Specifies the width of the image.
BORDER	Sets the width of the border surrounding the image.
ALT	Specifies alternative text.

Links: Relative, Absolute, and Name

What we call “links” are actually two different types of code in our HTML page: **hyperlinks**, which connect to a web page which is not in the same domain or directory as the present page, and **hypertext**, which is a type of link that connects to another document in the same domain or directory, or even within the same page. Here are some terms to examine before we look at the syntax of a link.

Absolute Link: You can identify an absolute link because it is a full URL. This is known as a **complete reference**, and is necessary for when you need to link to external content—that is, content that is on another web server or host. For example, “<http://www.google.com>” is an absolute link.

Relative Link: A relative link, by contrast, does not contain a complete reference; instead, it specifies only the filename, or possibly a *path* through the folders and then a filename. For example, “fruits/apple.doc” is a relative link.

Name Link: This type of link is commonly called an “anchor”, although this can be confusing since, technically, all links are anchors. This type of link, however, is used to “name” a place in the document to link to, which allows for navigation of lengthy web pages without a lot of scrolling.

Target: This is an attribute of the link tag, telling the browser where to display the page. For most browsers, the default place to display a link is in the same window—that is, by replacing the current page with the destination page in your browser.

For a link to another page or document, you use the tag:

```
<A HREF="http://domain/folder\(s\)/filename.html">Display text for the link</A>.
```

Obviously, if you're creating a relative link, you would omit the “http://domain” part:

```
<A HREF="folder/filename.html">Display text for the link</A>
```

In contrast, an anchor link requires a different attribute in the <A> tag: You designate a location within the page as follows:

```
<A NAME="Oranges">Here be Oranges</A>
```

Then, to allow a user to navigate directly to the section on oranges, you might have a link at the top of the page which allows the user to “jump” to this named location:

```
<A HREF="#Oranges">Go to the section on Oranges</A>
```

Finally, you can use the TARGET attribute to tell the browser to open a link in a new page or tab. For example, if I have a link from the LINGUIST List home page to Google, I might want it to open in a separate window. Although there are arguments about the actual efficiency of this tactic, many consider that it helps keep traffic—users—on your page. Here’s what that code might look like:

```
<A HREF="http://www.google.com" TARGET="_blank">Go to Google!</A>
```

Keep in mind that only the part displayed between the opening and closing tag will be displayed on the web page. That is, in the previous example, your link will appear like this:

[Go to Google!](#)

One last thing to keep in mind is that you can designate an image as a link. That is to say, you can arrange so that when someone clicks on a picture of an orange (perhaps with the word “Oranges” printed across it), they will be linked to the section on oranges, whether it’s on the same page or a completely different document. The syntax for this is as follows:

```
<A HREF="#Oranges"><IMG SRC="fruit/orange.jpg" ALT="Florida Oranges" BORDER="0"></A>
```

Where #Oranges refers to a page-internal link (as shown above).

HTML Tables: This is where it gets cool

One of the most powerful tools HTML gives us is the ability to create a table. As you might guess, at its heart, a table is simply an array of rows and columns, defined by three (or four) tags:

- <TABLE>:** Defines the bounds of a table. Tables may be nested inside one another.
- <TR>:** Defines a row of a table. Must be nested within the <TABLE> tag.
- <TD>:** Defines a single cell in a single row. Must be nested within a <TR> tag; more than one <TD> tag can be nested within a single <TR>
- <TH>:** This is called the table header tag, and is used like a <TD> tag in the first row of the table. Many programmers do not use this tag.

While the tags for laying out a table are deceptively simple, the breadth of what you can do with a table is really limited only by your imagination. Table attribute allow you to set height and width of rows and columns; text can be aligned by cell, so you can center text around a point (our LSA pages offer many examples of this). You can arrange text and pictures, create menus and lists, and so forth.

Although tables offer excellent control over placement, alignment, and formatting, there are some limitations. Again, the full use of tables is really beyond the scope of a beginning tutorial; however, learning to work with basic tables is not too difficult.

Colors, Fonts, and Other Assorted Beasties: When to use CSS

HTML has a set of tags which function to cover the rest of the formatting—mostly, of colors, fonts, and so forth. The tag allows you to control all aspects of the font (much like the toolbar in MS Word, that it, you can select the font “face”, the color, and the size. Additionally, the attribute “BGCOLOR”, which can be used in the <BODY> or <TABLE> tags, allows you to select a fill color for the whole page or a portion of it.

However, CSS (or Cascading Style Sheets) has developed into a coding language that allows a user to set a number of attributes for HTML tags in a way which is more specific but also more general, more powerful. CSS is mostly a topic for another entire tutorial; however, allow me to give one example of the power of CSS.

Let’s say you have a web site that has, conservatively, two dozen pages. As you build your site, you realize that all of your tables are tending toward one of three types. With HTML, you are forced to set the attributes for each of those tables at the tag level; that is, you have to repeat the attributes every time you open a <TABLE> tag. With CSS, you can define three “classes” of table, and simply reference the class you desire at the tag level. So, HTML would give you this:

`<TABLE WIDTH="300" BORDER="1" BORDERCOLOR="blue" ALIGN="center" BGCOLOR="gray">`
(which you would have to type every time you create a table of this type)

With CSS, however, what we do is store all the information of the attributes—width of 300, background color is gray, etc., elsewhere, and give it a name. Then, when we want to create a table with that styling, we just reference the name, giving you the following:

`<TABLE CLASS="bordered-gray">`

The tag is cleaner, all the information is transferred. An even more powerful feature of building your styles this way, however, is realized when you decide to make **all** the borders on this table style thicker. Instead of going to each of two dozen pages and checking for this table type, you change one number in the CSS style information, and all the tables will automatically change to match! CSS is much more complex, of course, but this will serve as an introduction. It's sometimes difficult to know when to use CSS; often, only experimentation and the needs of the project at hand can tell you.

Special Characters: Ampersand What?

Most browsers are unable to render certain special characters. This is of particular importance to us at LINGUIST List because we often deal with IPA versions, names with umlauts, and so forth. The ampersand (&) is a signal that a special character is about to be defined, and the semi-colon (;) marks the end of the special character. There is an extensive list of possible special characters; LINGUIST List has created a utility for looking them up.

As an example: Every browser interprets "<" as the beginning of a tag. So, if you're trying to display what a tag should look like on the web, you might type `<TAG>`. However, nothing will show up, because the browser checks for an HTML tag called "TAG" and finds nothing. Thus, if you want to display that example, you need to code the following:

HTML code: `<TAG>`
Display: `<TAG>`

Where 'lt' and 'gt' stand for 'less than' and 'greater than', the names of the symbols you're coding. Many last names in German provide another example:

HTML code: `Jürgen`
Display: Jürgen

"Good" Design and Best Practices

The last section of this tutorial has to do with the ticky-tacky things that make you a "good" programmer. Once you've mastered HTML and other programming languages, it's still possible to be careless or sloppy in your work. Therefore, there are a number of "rules" of good programming you can follow:

1. Indent your code.

Because each HTML file is simply line after line of code, it's important to make sure that your pages, when viewed in a code editor, are easily readable. The simplest way to do this is by **indenting** your code. As the name suggests, all you need to do is use the enter and tab keys to space your code out.

For example, the following, continued over an entire page, will get pretty tiresome to read through:

```
<P>Paragraph of text and so forth, nothing really interesting but relevant to your page.</P>
<TABLE>
<TR>
<TD>
Cell content
</TD>
<TR>
<TD>
Cell 2 content
</TD>
</TR>
</TABLE>
```

```
<BR>
</BODY>
</HTML>
```

This example, however, shows code that's much clearer, and, as a result, much easier to read:

```
<P>Paragraph of text and so forth, nothing really interesting but relevant to your page.</P>

<TABLE>
  <TR>
    <TD>Cell content</TD>
  </TR>
  <TR>
    <TD>Cell 2 content</TD>
  </TR>
</TABLE>

<BR>

</BODY>
</HTML>
```

So, indenting your code is not only a courtesy to anyone who needs to make changes to your page after you, but also a way for you to organize the tags on your page so you can easily find content and make alterations.

2. Comment your code.

This goes hand-in-hand with your indentation. ColdFusion in particular allows you to use a special set of characters to delimit a **comment**, which will only be visible to someone viewing the *original* source code (that is, not the browser source, but the true file source).

```
<!--This is a comment in ColdFusion. It will appear in gray italics and will never show up on your page as it's sent to the browser. This comment does nothing as far as code!-->
```

Commenting serves one extremely important purpose—you can explain, at length, the purpose, function, or authorship of any thing on the page. Because we often have several people who work on a single page, comments are crucial to keep things straight.

A good example of a comment is one that says, "This is the table that contains the top menu options". Another comment might say, "Radio button for Yes/No added by John Linguist on Nov. 30, 2006". Finally, it's often a good idea to leave a general comment at the top of the page, listing the page author, date created, purpose and location. As changes are made, this general comment can be expanded, leaving the page with a "history" of changes right at the top.

3. Pay attention to nesting

Often you may encounter a situation such as the following: perhaps a large portion of a page (such as the body in many Intranet pages) is formatted by a table. However, within one smaller area (within one of your table cells), you require the type of control you can only get from a table. **Nesting** is a term we use to mean that all the tags are in the proper order. Nesting rules also apply to ColdFusion code, and even text attributes like italic and bold tags.

Incorrect: `Heading`

Correct: `Heading`

Generally speaking, the first tag opened should be the last tag closed, and each tag opened should be closed in reverse order.

This seems relatively simple, but when you have 800 lines of code all containing tables, it can get difficult to see which table you're looking at on any given line. However, the next guideline can help overcome this.

4. Validate your work.

Both ColdFusion and W3 (on the web) offer tools to **validate** your work; that is, to check for errors in your code. The W3 validator is very powerful, but only validates HTML code. The validator built into

ColdFusion will test both HTML and check your CF tags, but since we're running a relatively outdated version, some of the conventions have changed.

As a result, I usually run both validators on each page, and correct whatever pops up to make my page error-free. Referring to the previous section, the validate functions are very good at catching nesting errors you may have made.

5. Do basic testing of your page(s), including a check for spelling errors!

Sometimes, it's easy to get so caught up in your coding that you misspell a simple word like "grammar". Therefore, you should not only check for gross errors, but as your page nears completion, or even if you're only working on a small section of a preexisting page, you should check your work carefully in the browser. Make sure there's no funky spacing, misspelling, bold text that shouldn't be, and so forth.