*/ June 2000 / version 4.5.2 /* *version history* */*

## About this Tutorial

We created this tutorial way back in 1994, when the web was young.

**WRITING HTML WAS CREATED** to help teachers create learning resources that access information on the Internet. Here, you will be writing a lesson called *Volcano Web*. However, this tutorial may be used by anyone who wants to create web pages. You can get a sense of the results by looking at our illustrious alumni and kudos or what people say about the tutorial.

By the time you have reached the end of this tutorial you will be able to construct a series of linked web pages for any subject that includes formatted text, pictures, and hypertext links to other web pages on the Internet. If you follow the steps for the Basic Level (lessons 1-14) you will develop a page about volcanoes and if you go on to the Advanced Level (lessons 15-29), you will create an enhanced volcano web site.

For faster performance, you can download an archive of all files used in this tutorial. Most of the lessons can be done off-line. If you are having trouble connecting to this site, try our experimental servers, Jade or Zircon but please be nice to these machines; they are doing other work for us.

## Why Create Web Pages?

If you've come this far, you likely have an answer.

**THE WEB IS BECOMING AN INTEGRAL PART** of our working (and playing) world. You cannot spit anymore these days without hitting a URL (if you do not know what a URL is, you will find out here). In a very short time span, the web has revolutionized the way we access information, education, business, entertainment. It has created industries where there were none before.

Being able to develop information on the web might be a job skill, a class requirement, a business necessity, or a personal interest. Unlike any other previous medium, the ability to "write" HTML allows you to potentially connect with millions of other people, as your own self-publisher.

## Objectives

This tutorial covers the steps for writing HTML files using illustrative examples for creating web pages.

**IN THESE LESSONS YOU WILL:**

- identify and use different HTML formatting codes.
- create and modify HTML documents using a simple text editor.
- write a series of web pages that present information, graphics, and provide hypertext links to other documents on the Internet.

And maybe you will have some fun!

## What is HTML?

**H**yper**T**ext **M**arkup **L**anguage

**PUT MOST SIMPLY,** HTML, is a format that tells a computer how to display a web page. The documents themselves are plain text files (ASCII) with special "tags" or codes that a web browser knows how to interpret and display on your screen.

This tutorial teaches you how to create web pages the old-fashioned way -- by hand. There are software "tools" that allow you to spin web pages without touching any HTML. But if you are serious about doing more than a page or two, we believe a grounding in the basics will greatly accelerate what you can do.

## Getting Ready

We will assume you have a basic knowledge of how to use your web browser menus, buttons, and hypertext links.

Everything you create in this tutorial is designed to run from any desktop computer; it does not depend on access to a web server or specialized computer programming.

### YOU WILL ALSO NEED A TEXT EDITOR PROGRAM

capable of creating plain text files e.g. SimpleText for the Macintosh or NotePad for Windows. ***We strongly urge that you use the most basic text editor while you learn HTML and then later you can explore HTML "editors"*** If you use a word processor program then you **must** save your files as plain ASCII text format. You should also be familiar with switching between multiple applications as well as using the mouse to copy and paste selections of text.

If you download the tutorial files, you can do nearly all of the lessons off-line.

We suggest that you proceed through the lessons in order, but at any time you can return to the index to jump to a different lesson. Within each lesson you can compare your work to a sample file for that lesson. Each lesson page has a link to a concise summary of the tags as well as links to other reference sites.

For convention, all menu names and items will be shown in **bold** text. All text that you should enter from the keyboard will appear in `typewriter style.`

## Keep in Mind

Some pointers to help you out, since we will never admit knowing everything.

a. Use the **Favorites** or **Bookmark** feature of your web browser to mark the lesson index page so you can easily navigate to other lessons.
b. We've aimed to write instructions generic to (almost) **any** web browser; sometimes the menu names or features may not match the web browser you are using.
c. This tutorial **will** show you how to create web pages that can see outward to the world. It **will not** tell you how to let the world see them; to do this you need to locate an Internet Service Provider that provides web server space. Try http://thelist.internet.com/ or http://www.webisplist.com/. Also, you can search for a free web page hosting service from Freewebspace.net
d. Creating pages is one thing, designing web sites is another. We cannot highly enough recommend the Yale C/aIM WWW Style Manual. Sun Microsystem's Guide to Web Style, and the Sevloid Guide to Web Design.
e. When you are ready for the big time, see web pages like you have never seen web pages at Dave Siegel's Casbah and High Five sites. Trudge on over to his Web Wonk to get the details. It will amaze you.
f. Refer to the HTML tag summary page as a reference. You can get to it by following the hypertext link at the top of every lesson page.
g. If you are having trouble, see the Writing HTML FAQ (Frequently Asked Questions) before writing us for help. We get lots and lots of e-mail. Too much.

## Who Did This?

Roll the credits!

**THIS IS A PROJECT** of the Maricopa Center for Learning and Instruction (MCLI). Writing HTML was developed by Alan Levine, instructional technologist at the Maricopa Community Colleges. Our former intern, Tom Super, provided invaluable instructional design support. Many others have given helpful suggestions, corrected typos, and expressed their thanks!

Once your web pages become available on the Internet, please list them on our Writing HTML Alumni page using our registration form.

Thanks to some great volunteer efforts, Writing HTML is also available in other languages:

- Spanish / Español *v3.0* (thanks to Arturo García Martín and Andrés Valencia)
- Icelandic / Íslenska "Námsefnisgerð í HTML" *v 4.5.2* (thanks to Gudjon Olafsson)
- Korean *v4.5* (thanks to Dr. Byeong choon Lim, Department of Computer Education Chuncheon National University of Education)
- Italian "Corso di HTML" *v4.5.1* (thanks to Cristiana Cavicchi)
- Japanese *v 4.0.`* (thanks to kazuaki mizota)

Or you can try the online translation tools from AltaVista's Babelfish:

Select a Language... Translate

## Time to Get Started!

**IF YOU ARE READY,** go to the index of lessons or go directly to the first lesson.

h a p p y  w e b b i n g

And have fun.

---

**Writing HTML**
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/

*/ June 2000 / version 4.5.2 /* *version history* */*

# Frequently Asked Questions (FAQ)

So you have reached a stumbling block in the tutorial? Do not worry! It happens often!

Perhaps yours is a question that has come up before. You may also want to review the introductory comments about the tutorial, the tag summary, or the reference list.


- Where is the download archive?
- I thought I should be doing the tutorial off line, but then can't access my pages because my web browser keeps trying to dial up my PPP.
- I've created my web pages, but why can't anyone else see them on the Internet?
- I've created my web pages on my desktop computer -- how do I get them to a WWW server?
- I've updated my web pages but do not see the changes in my web browser. Why?
- When I load my web pages into my web browser, why do I see odd characters at the top of the screen?
- Why do I see an icon with a question mark rather then my inline image?
- Could you please help with a tag that makes all HTML commands inside the tag text/ignored?
- How can I make a link that will the download a file?
- How do you create a counter that shows how many times someone has visited your page?
- I downloaded the Windows Zip archive and when I click on the Start link it cannot find locate file TUT/INDEX.HTM. Why?
- I can't get the `volc.html` file to load on my browser? I'm using Internet Explorer, is there anything I need to do?
- This tutorial is great! Can I make copies?
- How can I have a sound play when my page opens?
- Can I make a web page from webTV?
- How can I force the text not to wrap at the edge of the browser window?
- How do I get rid of the underlining of hyperlinks?
- What is this fascination about cheese in your lessons?

# Where is the download archive?

In January 1998, we changed the download format for the Windows versions of the tutorial from .ZIP files (which many people were unable to properly decompress) to an executable (.EXE) file. See the most current links for downloads from our page at:

http://www.mcli.dist.maricopa.edu/tut/download.html

# "I thought I should be doing the tutorial off line, but then can't access my pages because my web browser keeps trying to dial up my PPP. How do you run it off line for Web page design?"

Most web browsers have a **Preferences** or **Options** menu where you put the address of your favorite "home" page--that is, every time you launch the browser, it attempts to connect to this site. Some browsers have an option where you can specify it to start with a **blank** or **empty** page. Another approach is to cancel the connection when your modem tries to dial. Another idea (which you can do easily when you finish our tutorial) is to create your very own Home Page that sits on the hard drive. Use your web browser to **Open...** or **Open Local...** and find the page. Use your mouse to copy the address or file path indicated in the URL field (usually near the top of a browser window) and then paste it into the area of your Preferences/Options that says "Home Page". This way, when ever the web browser starts, you see your custom page with all the links you like, and you do not have to wait or even connect to an Internet server.

# "I've created my web pages, but why can't anyone else see them on the Internet? What's the URL to my hard drive?"

When you create your web pages, think of them as being able to see out to the entire Internet world (when you are connected to the network). BUT the entire world cannot see back into your computer since it does not have a WWW address on the Internet. If you want to *publish* on the Internet, you must first locate an Internet Service provider that rents space on its World Wide Web server. If you are at a school or a large company, contact your network administrators. You may want to contact the company that provides your access to the Internet and ask if they rent web server space. If you are shopping for a net provider, try

MecklerMedia's Provider List or WebISPList.

Another option is to use the free web page hosting service offered by Geocities or you can search for other free services sing the tools at Freewebspace.net

---

# "I've created my web pages on my desktop computer -- how do I get them to a WWW server"

So you have arranged somehow to get web server space! Generally, WWW servers are UNIX computers and you will have to find a utility to transfer files from your desktop computer to the WWW server. If you do not have a program, search the ShareWare.com for a "ftp" utility. If the WWW server you will use is a Macintosh or Windows-based computer, you may be able to transfer the files over your local network. This is one question you will have to ask of whomever is providing you access to the WWW server.

---

# "I've updated my web pages but do not see the changes in my web browser. Why?"

First, double-checked that you have **Saved** your HTML file from your text editor. The try using the **Reload** option in your web browser. Or, the browser may be looking at another copy of the HTML file; in the browser, use **Open File...** to read in the intended document.

---

# "Why don't I see the text in my `<title>....</title>` tag on my Web page?"

Recall from lesson 1 that the `<title>....</title>` tag is part of the information in the HEAD of your HTML file; only the BODY is displayed on the page. The text in the title tag should appear on the menubar of your web browser and it is how the browser will track your pages from its navigation/history menus. It's not uncommon to write what appears to be redundant HTML:

```
<html>
<head>
<title>New Products from Zippy Communications</title>
</head>
```

```
<body>
<h1>New Products from Zippy Communications<h1>
......
......
</body>
</html>
```

The same text is used twice -- once for the web browser to identify the page and once in the `<h1>` tag to put the same title on the page.

---

## "When I load my web pages into my web browser, why do I see odd characters at the top of the screen."

If you are using a word processing program to create your HTML files, be sure that you are saving them as plain text (ASCII) format -- these characters are hidden formatting codes. For Windows users, do not use the Write application -- it will add a bothersome "1" at the top of the screen. Your best bet is to start out by using the simplest text editor possible -- the Windows NotePad or TeachText/SimpleText for the Macintosh. Once you know the basic tags, then go looking for a program to help with the shortcuts.

---

## "Why do I see an icon with a question mark rather then my inline image?"

This icon means that your web browser could not locate the image file. first check to see that it is in the same folder/directory that you reference in the `<img src= "...">` tag. Next make sure the spelling of the file name exactly matches the file name written in the `<img src= "...">` tag

---

## "Why do I see an icon with a broken corner rather then my inline image?"

In this case, the external file is a format not recognized by your web browser. Make sure that the file is in the GIF format.

---

## Could you please help with a tag that makes all HTML

## commands inside the tag text/ignored?

***Bad news first...***

There is no such tag. Even if you use `<pre>...</pre>` tags, your browser will interpret any HTML as... HTML.

***Good news next...***

All you need to do is substitute the "special characters" (see lesson 9) to replace all occurences of the < and > characters:

- Replace all "<" with "<"
- Replace all "<" with "<"

This will display them as the characters and not interpret them as HTML.

---

## "How can I make the downloading function work? Is it just to specify where my zip-file is, the path to it? Or do I have to make a FTP server on our server. Is that all there is to it or is there some other magic working behind the scene on your server that I need to be aware of to make it work on our server?

No magic necessary. Just build your `<a href=.... >` links to point at the file. Even when you access files locally (like from your hard drive, your web browser will know how to handle the files. For Windows files, `.zip` and `.exe` files are pretty standard. Macintosh files on the other hand should always be compressed as BinHex (`.hqx`). Most web servers are preset to transmit files who's names end in these extensions.

---

## "How do you create a counter that shows how many times someone has visited your page?"

Counters require programs that run from a web server, which is really beyond the scope of just "Writing HTML." There are scads of information for counters at the Yahoo Access Counts

page. See also Web Counter or Internet Counter for a free service to add web page "hit-o-meters".

---

# I downloaded the Windows Zip archive and when I click on the Start link it cannot find locate file TUT/INDEX.HTM. Why?

We no longer provide the downloads in .ZIP format and have made it into a hopefully easy to use .EXE file. See the links from the download page.

---

# I can't get the `volc.html` file to load on my browser? I'm using Internet Explorer, is there anything I need to do?

With all the browsers out there, we had to write the directions to be generic. Here is how you open a local file in Microsoft Internet Explorer:

1. Select **Open...** from the **File** menu.
2. This allows you to type in a URL or provide the file path to a local file (the latter is what you want to do). The easiest way is to click the **Browse** button and use the dialog box to select the `volc.htm` file on your hard drive.
3. The easiest way is to arrange your desktop so that adjacent to the Explorer window you can see the folder/directory window that contains your HTML documents -- you can then just click, drag the icon for your file and drop it into the Explorer window.

---

# "This tutorial is great! Can I make copies?"

Yes, you can download the entire tutorial and use at your location. However, you must make sure that you give credit to the Maricopa Center for Learning and Instruction and the Maricopa Community Colleges. You may NOT sell it for profit or alter the content without permission.

---

# How can I have a sound play when my page opens?"

Generally, we recommend against doing this. To the viewer it can arrange from annoying to

obtrusive. You should provide the viewer the **choice** to hear a sound.

But if you insist... use the <embed> tag to point to a sound file (AIFF, WAV, or MIDI formats):

```
<center>
<embed src="sounds/groovy.wav" WIDTH=144 HEIGHT=60 autostart=true>
</center>
```

---

# "Can I make a web page on webTV"

I cannot say I have first hand knowledge, but others have written us and said it was possible. You can find the answers (and more) from the webTV Resources site a collection of resources collected by webTV users.

While webTV is primarily a **viewing** technology for the web, with some patience and some pointers, you may be able to use it as a creation tool. In our opinion, though, if this is anything other than a hobby, get a real tool for the job.

---

# "How can I force the text not to wrap at the edge of the browser window?"

There are some page designs where you may not want the text content to wrap-- notably a large table of data perhaps in a <pre>...</pre> or perhaps a timeline where you would like the user to use the scroll bar to navigate through content laid out in horizontal layout.

There is a subtle variation of the line break tag, namely the No Break tag <nobr>...</nobr> which tells the browser to not wrap whatever is inside, which could be text, pictures, or any content. The usage would be something like:

```
<nobr>
<h1>Come Scroll with me, away to the right, as I list
out all of the long answers to the
meaning of life accessible only to those that can scroll,
scroll, scroll...</h1>
</nobr>
```

Another example is a framed page where the lower frame contains a horizontal scrolling list of links to images.

# "How do I get rid of the underlining of hyperlinks?"

Historically this was not an option- it was an option for the person viewing your set to set in their web browsers. However, the features avilable to browsers that support Cascading Style Sheets (version 4.0 browsers) can accomplish "un-underlined" links.

Just place the following code inside the <HEAD>...</HEAD> of your HTML file:

```
<style type="text/css">
<1--
A:link, A:visited, A:active { text-decoration: none }
-->
</style>
```

Use this keeping in mind that many people may not know test is hypertext without the familiar underlining.

# What is this fascination with cheese in your lessons?"

A fair enough question, as we insert whimsical web examples in lessons featuring "Sir Longhorn", "the great Cheese Crusade of 1167", "Holy Cheese from Switzerland", etc. 4, 8a, 8d, 9, 10, 18, 20, 22, 27b, 28a, 28b, 29a, and 29e!

There is no meaning, just picking something silly, but if you think that web sites about cheese are weird, check again with CheeseNet!

**Writing HTML:** Frequently Asked Questions (FAQ)
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
<Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/faq.html

*/ June 2000 / version 4.5.2 /* *version history* */*

# l e s s o n s

Below are links to all of the lessons in this tutorial. Most of the lessons can be done off-line if you download to your computer an archive of the tutorial pages. We've provided links at the top of every lesson page to a brief summary of all HTML tags covered in these lessons. If you are having trouble, first check the Frequently Asked Questions also linked from the top of every lesson page.

## HTML 101

How the web works

0. **Standardly Speaking About HTML**

## Building a Foundation... Nuts 'n Bolts HTML

Basic tags for formatting pages to HTML 2.0 standards. These codes will make your pages viewable to the widest audience range.

1. **Creating Your First HTML Document**
2. **Modifying an HTML Document**
3. **Headings: Six Levels Deep** <h1> <h2> ...
4. **Breaking up into Paragraphs** <p> <br> <hr>
5. **Doing it with *Style*** <b> <i> <tt>
6. **Lists, Lists, and Lists** <ul> <ol>
7. **Graphics and File Formats**
    a. Inline Images <img...>
8. **Linking it with Anchors**
    a. Links to Local Files <a href="file.html ...>
    b. URLs: Web Pointers http:, ftp:, gopher:..
    c. Links to the World: Internet sites <a href="http:// ...>
    d. Links to Sections of a Page <a name= ...>
    e. HyperGraphic Links <a href= ...><img...></a>
9. **Preformatted Text** <pre>
10. **Special Characters** é ©

11. **Definition Lists** <dl> <dt> <dd>
12. **Address Footers and E-Mail Links** <address> <a href=mailto:...>
13. **You can Blockquote Me on That** <blockquote>
14. **Lumping vs. Splitting**

# Beyond the Basics

Modify and enhance your web pages with features available in HTML 3.2.

While we cannot provide instruction in as great detail on the more complex things you can include in your web pages, we provide links to other resources that may assist you.

15. **Standard and Enhanced HTML**
16. **Colorful And Textured Backgrounds** <body bgcolor=...>
17. **Don't Blink, Don't Marquee**
18. **Spiffing Up Text** <font color=... face=... size=... > <sup> <sub> <u> <strike>
19. **Easy Horizontal Rules** <hr>
20. **Extra Alignment** ,  <img vspace=..., hspace=...
21. **Setting the Table** <table..>
22. **More for Images and Lists** <BORDER=0..>, <ol type=..>
23. **Clickable Image Maps** <map...>
24. **META in your HEAD** <META...>
25. **Target That Window** <a href=... target=...>
26. **Web Page, You've Been Framed** <frameset cols=... > <frame src=... >
27. **A Wee Dose of JavaScript** <script language=JavaScript... >
    a. Alerts and Rollovers <a href=... onClick="alert('...')" onMouseOver="...
    b. Dynamic Content document.write('..
    c. Custom Window Openers window.open('..
    d. Swapping Images onMouseOver=... onMouseOut=...
28. **Adding some FORM to your webs**
    a. Forming Forms <form..>
    b. Form Action by email and CGI <form action=..>
    c. Form Action by JavaScript
29. **Multimedia in Your Page**
    a. Animated My GIF!
    b. Movie Time <embed src=...>
    c. Sound of [web] Music
    d. Hit Me With a Shockwave
    e. Small Cup of Java (to go) <applet code=...>

## The Next Generation

Moving your web pages into the future with HTML 4.0 features and then some

HTML has come a long way since we wrote this tutorial in 1994! We had planned to add new lessons for Dynamic HTML, Cascading Style Sheets, and perhaps even XML. However, as these are much more comprehensive concepts than HTML (and would greatly increase the size of this package), our next plan is to develop brand new, separate tutorials.

Besides fixes for typos in the current lessons there would be no major updates before August 2001. We have selected other reliable tutorials on these subjects in our references section.

Until then, keep on writing great HTML.

## Post-Graduate Work

Things to do and look at once you've mastered the content here.

- If your work is available on the web, be sure to register as a Writing HTML Alumni.
- Dave Siegel's Web Wonk will show you how to create artful web pages that look like no other web pages you have seen (except for those that mimic the style!). See also his How to Create Killer Websites a book, a web site, and an experience.
- The Yale C/aIM WWW Style Manual should already be on your bookmark list! This is high octane stuff. We also like Writing for the Web by Jakob Nielsen, PJ Schemenaur, and Jonathan Fox at Sun Microsystems, and the Sevloid Guide to Web Design. For more on shaping raw content into effective web pages, see James West's Information Design Tutorial.
- Always keep within a mouseclick's distance Kevin Werbach's Bare Bones Guide to HTML, webreference.com, HTML Goodies, and Dr. HTML.

---

**Writing HTML:** The Lessons
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/lessons.html

# 0. Standardly Speaking About HTML

Ahhh, there are always **rules** to follow. For HTML, fortunately, the rules are few in number and what they offer is large...

## Objectives

This is just an introduction to some concepts behind HTML. After this lesson you will be able to:

- Express the importance of HTML standards
- Describe some of the differences between HTML 2.0, HTML 3.2, and HTML 4.0

---

## Lesson

HTML, or **H**yper**T**ext **M**arkup **L**anguage, is how a web browser displays its multimedia documents. The documents themselves are plain text files (ASCII) with special "tags" or codes that a browser knows how to interpret and display on your screen.

### About those standards

No kidding -- the World Wide Web is **exciting.** It is everywhere. It has exploded beyond everybody's expectations (Well, back in 1994 we thought it would be big ;-)

Keep in mind that the thing that makes the Web (and the Internet in general) work are agreed-upon rules ("standards") that allow users of almost any kind of computer able to communicate and share information.

*Where does HTML fit into all of this?*

What we cover in this tutorial is aimed toward producing documents that comply with current HTML standards.

By using "standard" HTML, your work is going to be most widely "shareable" in the fast changing future of the 'net. The early set of standards, known as HTML 2.0, are supported by nearly all web browsers in

use right now.

Things got somewhat more complicated with the features included in HTML 3.2 since Netscape and Microsoft have introduced many features that go beyond standard HTML, and were at first supported by certain web browsers. The web really took off in popularity during the time of the 3.2 standard. By its original design, HTML was **not** designed as a formatting tool, yet people have found ways (some might say "tricks") to attempt to use HTML for precise web page formatting.

The current set of proposed standards is HTML 4.0 which contain more features for HTML and some attempts to reduce the complexities of different web browsers. This version is starting to move towards a more "logical" method of formatting web pages, via "Style Sheets" which allows the precise formatting web designers wish for, and in a way to separates it from the content, making it easy to update the design of a web site. However, it will take some time before this functionality is common and there are still bothersome differences between different web browser software (some "standards", yes?) These "standards" turn out to be recommendations as no one has the authority to enforce them!

What does this mean? For accessibility on the widest range of possible web browsers and versions out there, stick with the most basic set of HTML code. Of course, this may limit what you'd like to put in a web page! If you include HTML that may look snazzy only in Netscape but not Internet Explorer, you may turn people away from your site. Not only that, viewers of your web pages may not only be using different browsers, but their monitor size and fonts may not be the same as on the system you designed the pages.

After all, you are probably not going to spend all of this time designing web pages that are for your viewing only! The idea is to make something that the world can view. So the first section of lessons will take you through the most widely accepted features of HTML. From there, you can make the decision to use more of the "deluxe" features.

# Review Topics

1. What is HTML?
2. Why should you be concerned about differences in HTML standards?

# Coming Next....

Time to start writing! Are you ready? In the next lesson you will see how to juggle three open windows as you write your first lines of HTML.

**GO TO.... | Lesson Index | next: "Creating Your First HTML Document" |**

**Writing HTML** Lesson 0: Standardly Speaking

©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine

Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut0.html

# 1. Creating Your First HTML Document

You are about to embark on a journey that will transform you from a mere **Internet Surfer** of the Web to an **Internet Author of Multimedia!**

## Objectives

After this lesson you will be able to:

- Identify the meaning and purpose of HTML tags.
- Open up a workspace for creating new HTML documents.
- Use a text editor to create the basic HTML structure for any web page.
- Insert non-displayed comments into your HTML files.
- Open your document within your web browser to see how it is displayed.

---

## Lesson

Now that you know what HTML is, let's start using it.

> (Quick quiz -- what do those letters stand for? If you read the previous lesson you would know!).

### What are HTML tags?

When a web browser displays a page such as the one you are reading now, it reads from a plain text file, and looks for special codes or "tags" that are marked by the < and > signs. The general format for a HTML tag is:

```
<tag_name>string of text</tag_name>
```

As an example, the title for this section uses a **header** tag:

```
<h3>What are HTML tags?</h3>
```

This tag tells a web browser to display the text **What are HTML tags?** in the style of header level 3

(We'll learn more about these tags later). HTML tags may tell a web browser to bold the text, italicize it, make it into a header, or make it be a hypertext link to another web page. It is important to note that the ending tag,

```
</tag_name>
```

contains the "/" slash character. This "/" slash tells a web browser to stop tagging the text. Many HTML tags are paired this way. If you forget the slash, a web browser will continue the tag for the rest of the text in your document, producing undesirable results (as an experiment you may want to try this later).

> **NOTE: A web browser does not care if you use upper or lower case. For example, <h3>...</h3> is no different from <H3>...</H3>**

Unlike computer programming, if you make a typographical error in HTML you will not get a "bomb" or "crash" the system; your web page will simply look, well... wrong. It is quick and easy to go inside the HTML and make the changes.

Your browser has a small but open vocabulary! An interesting aspect of HTML is that if the browser does not know what to do with a given tag, it will ignore it! For example, in this document you are viewing, the header tag for this section *really* looks like this:

```
<wiggle><h3>What are HTML tags?</h3></wiggle>
```

but since your browser probably does not support a **<wiggle>** tag (I made it up, perhaps in the future it could cause the text to wave across the screen?), it proceeds with what it knows how to do. If I were programming a new web browser, I might decide to add the functionality for the **<wiggle>** tag into my software.

## Opening Up Your Workspace

To complete the lessons in this tutorial, you should create a second web window (this allows you to keep this window with the lesson instructions and one window as your "workspace"), plus open your text editor application in a third window.

> **NOTE: This is a good place to remind you that we will provide directions that are somewhat general as the menu names and file names can differ depending on which web browser you are using. If our instructions say, "Select Open Location... from the File Menu" and your web browser does not have that *exact* choice, try to find the closest equivalent option in your own web browser.**

So you will want to be pretty comfortable jumping between different applications and windows on your

computer. Another option is to print out the lesson instructions (but we really do not want to promote that kind of excessive tree carnage).

Here are the steps for setting up your "workspace":

1. From the **File** menu of your web browser, select **New Window** or **New Web Browser** (The exact name of the menu command can be different depending on what browser you are using). A second web window should appear. Think of the first window as your "textbook" and the second clone window as your "workspace" for completing the HTML lessons.

   **NOTE: The only reason to have two windows here is so that you can read the instructions for the lessons and also view your working document. It is *not* mandatory to have two windows open; it just makes your work easier. You could also bookmark this web page or jump back here via your Go or History menu.**

2. Next, you need to jump out of the web browser and open your text editor program.

   **NOTE: You will need to move back and forth between the different windows to complete these lessons. This can be a challenge depending on the size of your monitor. You may choose to resize the three windows so that they all fit on your screen or layer your windows so you can click on any of them to bring it to the front.**

   **If you are using a word processor program to create your HTML, be sure to save in plain text (or ASCII) format.**

If you are just starting out, we most STRONGLY recommend that you use the simplest text editor available -- *SimpleText* for the Macintosh or the Windows *NotePad*. Why not use those nifty HTML editors? It is sound instructional design that you first learn the concepts and THEN look for shortcuts or helpers that make the work less tedious.

Also, it will help you if you first create a new directory/folder on your computer that will be your work area. You can call it **`workarea`** or **`myspace`** or whatever you like; just make sure that you keep all of the files you create in this one area. It will make your life simpler... well, at least while working on this tutorial!

## Creating Your HTML Document

An HTML document contains two distinct parts, the head and the body. The **head** contains information about the document that is not displayed on the screen. The **body** then contains everything else that is displayed as part of the web page.

The basic structure then of any HTML page is:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<!-- header info used to contain extra information about
     this document, not displayed on the page -->
</head>

<body>

<!-- all the HTML for display -->
        :      :
        :      :
        :      :
</body>
</html>
```

The very first line:

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

is not technically required, but is a code that tells the browser what version of HTML the current page is written for. For more information, see the W3C Reference Specification.

Enclose all HTML content within **<html>...</html>** tags. Inside is first your **<head>...</head>** and then the **<body>...</body>** sections.

Also note the **comment** tags enclosed by **<!-- blah blah blah -->**. The text between the tags is NOT displayed in the web page but is for information that might be of use to you or anyone else who might look at the HTML code behind the web page. When your web pages get complicated (like you will see when we get into tables, frames, and other fun stuff about 20 lessons from now!), the comments will be very helpful when you need to update a page you may have created long ago.

Here are the steps for creating your first HTML file. Are you ready?

1. If it is not open already, launch your text editor program.
2. Go to the text editor window.
3. Enter the following text (you do not have to press RETURN at the end of each line; the web browser will word wrap all text):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>Volcano Web</title>
</head>
<!-- written for the Writing HTML Tutorial
by Lorrie Lava, February 31, 1999       -->
<body>
In this lesson you will use the Internet to research
information on volcanoes and then write a report on
your results.
</body>
</html>
```

**NOTE: Look where the `<title>...</title>` tag is located. It is in the `<head>...</head>` portion and thus will not be visible on the screen. What does it do? The `<title>` tag is used to uniquely identify each document and is also displayed in the title bar of the browser window.**

**In lesson 3 you will learn how to add a string of text for a title that will appear directly on your web page.**

**Also note that we have inserted a comment tag that lists the name of the author and the date the document was created. You could write anything in between the comment tags but it is only visible when you look at the source HTML for a web page.**

4. Save the document as a file called **`"volc.html"`** and keep it in the "work area" folder/directory you set up for this tutorial. Also, if you are using a word processor program to create your HTML, be sure to save in plain text (or ASCII) format.

   **NOTE: For Windows 3.1 users, you *must* save all of your HTML files with names that end in `.HTM`, so in this case your file should be `VOLC.HTM`. Do not worry! Your web browser is smart enough to know that a file that has a name that ends in `.HTM` is an HTML file.**

   **You can create files with names like `VOLC.HTML` if you use Windows95 or a later Windows operating system.**

   By using this file name extension, a web browser will know to read these text files as HTML and properly display the web page.

## Displaying Your Document in a Web Browser

1. Return to the web browser window you are using for your "work space". (If you do not have a second browser window open yet, select **New Window** or **New Browser** from the **File** window.)
2. Select **Open File...** from the **File** menu. (Note: For users of Internet Explorer, click the **Browse** button to select your file)
3. Use the dialog box to find and open the file you created, **`"volc.html"`**
4. You should now see in the title bar of the workspace window the text "Volcano Web" and in the web page below, the one sentence of **`<body>`** text you wrote, "In this lesson..."

---

# Check Your Work

Compare your document with a sample of how this document should appear. After viewing the sample, use the **back** button on your web browser to return to this page.

If your document was different from the sample, review the text you entered in your text editor.

A common mistake we hear is, "I cannot see the title!" You shouldn't! The text within the **`<title>...</title>`** tag is **NOT** displayed on the web page; you should see it in the title bar of the web browser window.

The most common mistake that beginners make here is that they try using a word processing program to type HTML and then are unable to open it in their browser, or if it does, the page is full of odd garbage characters. **When you are starting out, we urge you to use the most basic text editor such as the Windows NotePad or SimpleText for Macintosh.** Look for shortcuts later!

If you are looking for some free/cheap alternative text editors, our recommendations are EditPad (for Windows) and BBEdit Lite (for Macintosh)

# Review Topics

1. What are HTML tags?
2. Where is the text of the title tag displayed?
3. What steps are involved in creating a simple HTML document?
4. How do you create a comment tag?
5. How can you display your HTML document in a web browser?

# Independent Practice

Think of a topic for your own web page. Now create your own HTML text file that includes a **<title>** tag and a few introductory sentences. Save the HTML file and reload it in your web browser. You might want to create a different folder/directory for this file so you do not get it mixed up with all of the volcano pages you will create for this tutorial.

Keep this file handy as you will add to it in later lessons.

---

# Coming Next....

Your first web page is done!

But, to be honest, it is pretty short and not very exciting! In the next lesson you will modify and update your HTML document.

**GO TO.... | Lesson Index | previous: "Standardly Speaking" | next: "Modifying HTML" |**

---

**Writing HTML:** Lesson 1: Creating Your First HTML Document
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut1.html

# 2. Modifying an HTML Document

Now that you have created your first HTML document, you will learn how to swiftly make changes in your document and view the updates within your web browser.

# Objectives

After this lesson, you will be able to:

- Re-open the workspace for your web page.
- Make changes in your HTML document using the text editor.
- Reload the document in your web browser to see your changes.

---

# Lesson

## Re-opening Your Workspace

**Note: If you do not have the document from the previous lesson, download a copy now.**

To complete this lesson, you will need to create a second web browser window and re-open the text editor window you used in the first lesson. Here are the steps for re-opening your workspace (remember that the exact name of the menu commands may be different depending which web browser you are using):

1. If not open, create a new web browser window by selecting **New Window** from the **File** menu.
2. Use the **Open File...** command from the **File** menu to find and open the HTML file you created in the previous lesson.
3. Re-open your text editor program.
4. In the text editor, open the file (**`"volc.html"`**) you created in the previous lesson.

> **NOTE: If you are using Windows 3.1 computer then your file should be named `"VOLC.HTM"`. From now on, we will assume that you can easily re-open your workspace in this manner.**

# Making Changes in Your HTML Document

1. Go to the text editor window.
2. Below the text you typed from the previous lesson, press RETURN a few times and type the following text:

```
A volcano is a location where magma,
or hot melted rock from within a planet,
reaches the surface. It may happen violently,
in a massive supersonic explosion, or more
quietly, as a sticky, slow lava flow.
```

   Note that this text should be **above**the **</body>**and **</html>**tags at the bottom of your HTML file.

3. Select **Save** from the **File** menu to update the changes in your HTML file.

# Reloading the Document in your Web Browser

Return to the web browser workspace where the previous version of your file was displayed. Note that the new text you entered in the previous steps may not yet be visible. To see the changes, use the **Reload** button or menu item in your web browser. This instructs your web browser to read in the same HTML file and display it with whatever changes have been made. You should now see the new text that you entered.

Note that the web browser ignores all blank lines and extra spaces (carriage returns) that you enter in the HTML file. It will also ignore any extra space characters (beyond the one between words). However, when you are writing HTML, it will help you greatly to separate major sections by some blank lines... when you need to go back and edit content, it makes it easier to locate the correct location to make the changes.

Of course, there will be times that you **want** your web pages to have blank space between sections (e.g. between paragraphs). You just passed a location in this very page! In Lesson 4 we will learn how to do this.

### Drag and Drop Bonus!

There *may* be an easier way for you to load and view your HTML pages. You will have to arrange your computer desktop so that you can see the icon for your HTML files adjacent to your web browser window. Simply click and drag the icon for your **"vol.html"** or **"vol.htm"** file right into your web browser window. Voilà! your page will display if your computer supports drag and drop operations (It works for operating for Macintosh OS 7.5 and Windows 95 or newer).

---

# Check Your Work

Compare your document to this sample of how this document should appear. If your page looks different than the example, review the text you entered in the text editor. Make sure it matches the text instructions in the **Making Changes in Your HTML Document** section of this lesson.

# Review

Review topics for this lesson:

1. How did you re-open your workspace?
2. What steps did you use to make changes in your HTML document?
3. How did you display and view these changes in your web browser?

# Independent Practice

As you did in the lesson, modify your own HTML document that you started in the last lesson. Add a few more sentences and see if you can successfully reload the modified document into your web browser.

---

# Coming Next....

Now that you have an understanding of the editing process, we will add big and beautiful section headings to your HTML documents.

**GO TO.... | Lesson Index | previous: "Creating HTML" | next: "Headings" |**

---

**Writing HTML:** Lesson 2: Modifying an HTML Document
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine

Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut2.html

# 3. Headings: Six Levels Deep

As you see in this web page, the section headings ("Headings -- Six Levels Deep", "Objectives", "Lesson", "HTML Headings" ...) appear as different sizes and, perhaps, different colors and fonts. HTML provides tags for designating headings in six levels of significance. Your browser determines the exact font and size for display.

## Objectives

After this lesson, you will be able to:

- Identify the different levels of headings in HTML and the tags associated with them.
- Place different level headings within your HTML document and view the changes within your web browser.

---

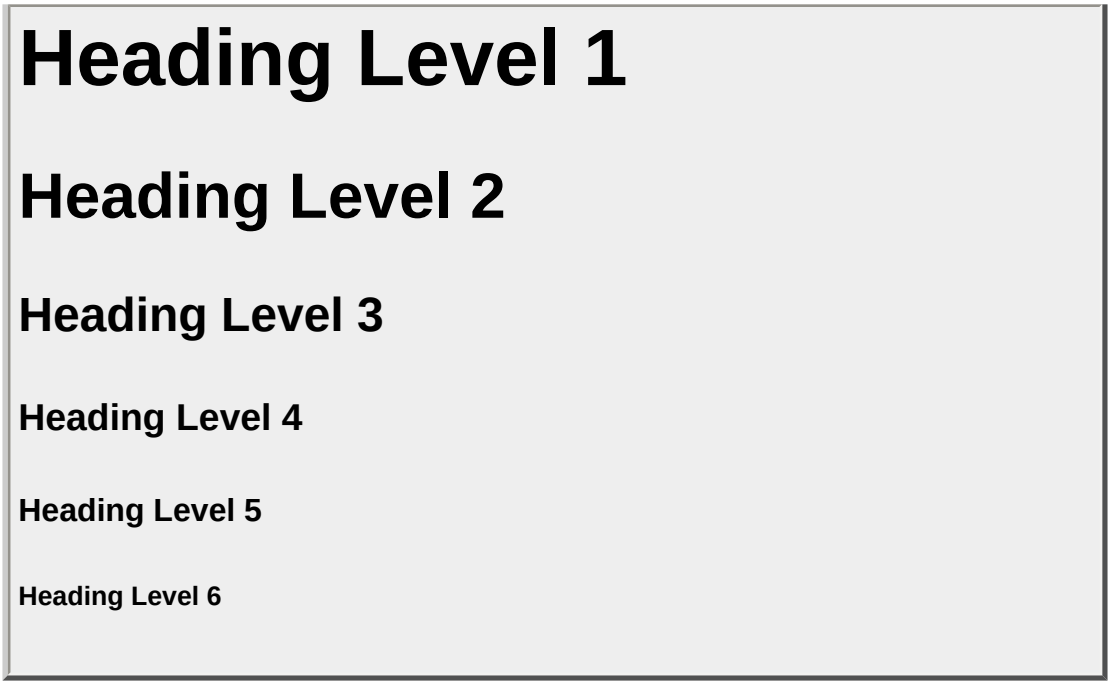
## Lesson

### HTML Headings

You created headings in HTML by "tagging" certain chunks of text with heading tags. The format for an HTML heading tag is:

```
<hN>Text to Appear in Heading</hN>
```

where **N** is a number from 1 to 6 identifying the heading size. Here are some examples of different heading sizes:




Heading levels range from level 1 (Most Important) to level 6 (Least Important). Like an outline, your heading levels should have logical, consistent order and be parallel.

## Placing HTML Headings in Your Document

**Note: If you do not have the working document from the previous lesson download a copy now.**

1. Re-open your workspace (if not already opened).
2. Go to the text editor window.
3. Open the HTML text file you created in lesson 2, **"volc.html"**.
4. First, we will use the tag to display the title as the biggest header, **<H1>**. Enter the following above the existing body text and **after** the **</head><body>** tags:

```
<h1>Volcano Web</h1>
```

5. Below the text already entered, create other headings for future sections of your *Volcano Web* page.

   Enter the following headings inside the body of your web page (Note that some are **H3** and others are **H2** tags):

```
<h2>Introduction</h2>

<h2>Volcano Terminology</h2>

<h2>Volcanic Places in the USA</h2>
```

```
<h3>Mount St Helens</h3>

<h3>Long Valley</h3>

<h2>Volcanic Places on Mars</h2>

<h2>Research Project</h2>

<h3>References</h3>
```

6. Save changes in your text editor.
7. Return to your web browser, **Open** and **Reload** the HTML file.

> **Note that on the computer you are using now, you can use the settings in your web browser to define the fonts and/or size of the headings. For example, on one computer you could have a browser display h1 tags as Times font and 36 point; h2 tags as Helvetica font and 24 point, etc. HTML codes designate only that the headers are of a certain type (h1 to h6); how it is displayed is controlled by the user of the web browser.**

---

# Check Your Work

Compare your work to this sample. If some of your headings do not appear correct, be sure to check that the starting tag and ending tags have the same heading level.

As an optional exercise, take a look at what happens when you make a typographical error. Open your HTML document in the text editor and delete the slash (/) in the **<h1>** tag, after the header **Volcano Web**:

```
          <h1>Volcano Web<h1>
[missing "/" ------------^^^]
```

Save the changes and reload into your web browser. Without the correct ending of the **h1** tag, your web browser interprets all of the succeeding text as part of that header! After trying this you should go back to your document and re-insert the slash in the correct spot.

# Review Topics

1. What are the different levels of headings in HTML?
2. What are the tags associated with these different levels?
3. What steps did you use in placing headings in your HTML document?
4. What happens if you forget a slash at the end of a header tag?

# Independent Practice

Add at least three headers of different levels to your own HTML document.

---

# Coming Next....

Breaking up text into paragraphs.

**GO TO.... | Lesson Index | previous: "Modifying HTML" | next: "Paragraphs" |**

---

**Writing HTML:** Lesson 3: Headings: Six Levels Deep
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut3.html

# 4. Breaking it up into paragraphs

So far you have created and modified HTML documents, and you should feel comfortable with the process of editing text and reloading it into your web browser. So now relax for this **fast** lesson on inserting paragraph breaks.

# Objectives

After this lesson, you will be able to:

- Identify the paragraph break tag in HTML.
- Copy text from the web page and paste it in another document.
- Insert paragraph breaks into the text of your HTML document and view the changes within your web browser.

---

# Lesson

## HTML Paragraph Breaks

We've seen earlier that a web browser will ignore all of the CARRIAGE RETURNS typed into your text editor. But, wherever a browser sees the **`paragraph`** tag, it inserts a blank line and starts a new paragraph. The HTML code for forcing a paragraph break is:

```
<p>
```

Note that this tag is special in that it does not require an ending tag; for now you do not need to use:

```
</p>
```

In a later lesson we will see why we want to use ** ** a closing ** ** for the more current HTML coding standards. For basic HTML coding, let's keep it simple for now.

Also, the **<h>** tags have a built in break so it is unnecessary to put ** ** tag before a header tag:

```
<p>
<h2>Blah Blah Blah Blah</h2>
```

## Inserting Paragraph Breaks

**Note: If you do not have the working document from the previous lesson, download a copy now.**

Follow the directions below to insert and view a paragraph break in your HTML document.

1. Re-open your workspace (if not already opened).
2. Go to the text editor window.
3. Open your working document, **`volc.html`**, in the text editor (if not already opened).
4. First we want to move the sentences **`("A volcano is ....")`** so that they are under the **Introduction** heading. Use the mouse to **cut** and **paste** this text in the proper location.
5. After these sentences, we want to add some more text. But rather than re-typing this in, from this web page use your mouse to select and **copy** the sentences:

```
Volcanoes have been a part of earth's history
long before humans. Compare the history of human
beings, a few million years in the making, to that of
the Earth, over four billion years in the making.
```

6. Now, return to your HTML document in the text editor, and **paste** this text after the existing sentences under the **`<h2>Introduction</h2>`** heading.
7. Save the changes in the text editor.
8. Return to your web browser.
9. If your working document is not visible, Use the **Open Local...** command from the **File** menu to open the document.
10. Select **Reload** from the **File** menu. You should now see the two sentences of the Introduction. We now want to put a paragraph break *between* these sentences.
11. Again, return to your HTML document in the text editor.
12. After the second sentence under **`<h2>Introduction</h2>`** (the one that ends " as a sticky, slow lava flow."), press RETURN (it is not necessary but it makes the HTML more readable as you work on it), and then enter the **`paragraph`** tag:

```
<p>
```

This section should now look like:

```
<h2>Introduction</h2>
```

```
A volcano is a location where magma,
or hot melted rock from within a planet, reaches the surface.
It may happen violently, in a massive supersonic explosion,
or more quietly, as a sticky, slow lava flow.
<p>
Volcanoes have been a part of earth's history long before
humans. Compare the history of human beings, a few million
years in the making, to that of the Earth, over four
billion years in the making.
```

13. Save the changes in the text editor.
14. Return to your web browser and **Reload** the document. The two sentences of the introduction should now be separate paragraphs.

---

## Other types of breaks

To separate major sections of a web page, use the **`horizontal rule`** or **`hr`** tag. This inserts a straight line like you see right above the heading for this section.

The HTML format for a **`horizontal rule`** tag is:

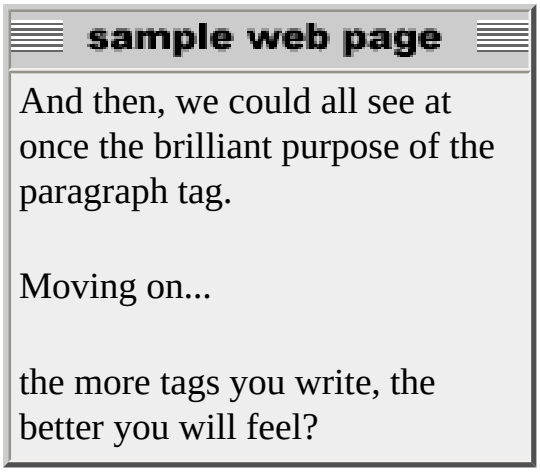
```
<hr>
```

Let's try it now! Put an **`hr`** tag above the **Introduction** heading. This will help to separate the opening sentence of the lesson from the other portions that will follow.

And finally, there is the **`<br>`** tag which forces text to a new line like the **`<p>`** tag, but without inserting a blank line. You might use this tag when making a list of items, when writing the lines of a poem, etc. Compare the differences between using the **`<br>`** and **`<p>`** in these two examples:

| **Paragraph <p> tags Only** | |
|---|---|
| **HTML** | **Result** |

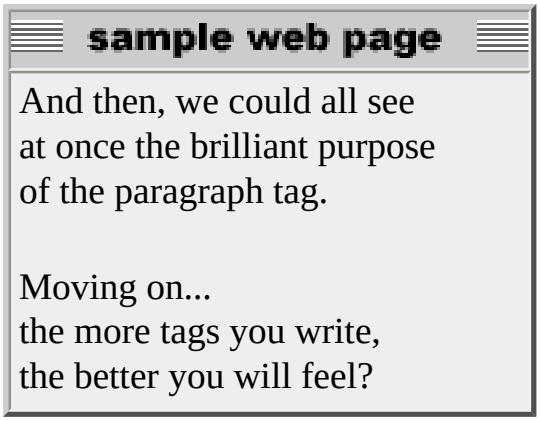
```
And then, we could all see
at once the brilliant purpose
of the paragraph tag.
<p>
Moving on...
<p>
the more tags you write,
the better you will feel?
```




## Paragraph <p> and Line Break <br> tags

| HTML | Result |
|---|---|

```
And then, we could all see<br>
at once the brilliant purpose<br>
of the paragraph tag.
<p>
Moving on...
<br>
the more tags you write,<br>
the better you  will feel?
```

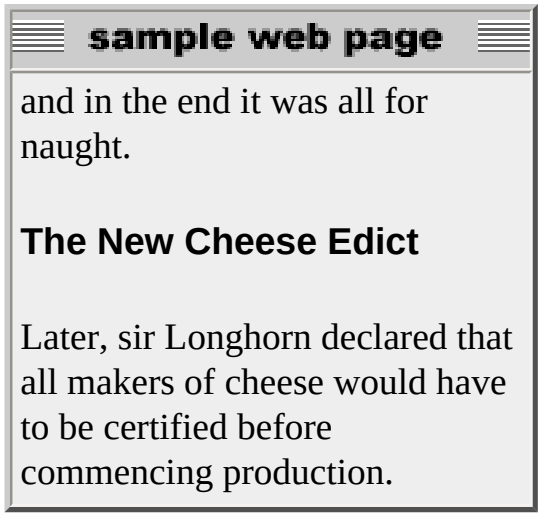



The **<br>** tag can be used for a different layout style for your section headings. If you notice, the **header** tags **<h1>, <h2>, ...** automatically insert white space above and below the text of the header tag. Some web page authors prefer a style that controls this white space.

## Section titles with Header Tags

| HTML | Result |
|---|---|

```
and in the end it was all
for naught.

<h4>The New Cheese Edict</h4>
Later, sir Longhorn declared
that all makers of cheese would
have to be certified before
commencing production.
```



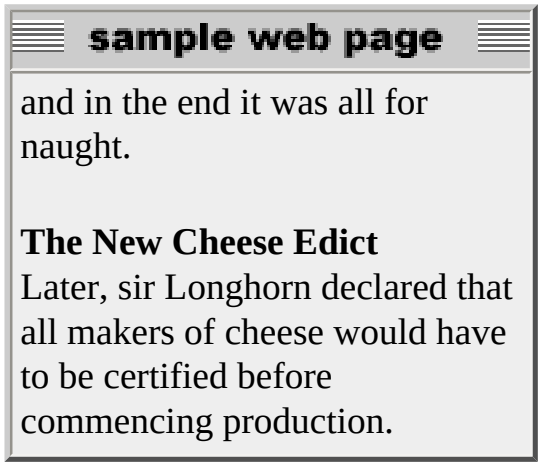

## Section titles with <b> and <br> tags

**NOTE! The <b> tag is covered in the next lesson but all it does is make the text bold.**

**HTML**

```
and in the end it was all
for naught.
<p>
<b>The New Cheese Edict</b><br>
Later, sir Longhorn declared
that all makers of cheese would
have to be certified before
commencing production.
```

**Result**




The difference may seem trivial now, but it opens up possibilities when later we learn to create text of different size and color for our section headings. For example, take a look at Web Pages That Don't Look Like Web Pages

# Check Your Work

If you would like, compare your web page to this sample. If your document was different from the sample, review how you entered the paragraph break, **<p>**, into the text editor. Make sure you entered it as instructed in the **Inserting Paragraph Breaks** section of this lesson.

# Review Topics

1. What is the HTML tag for a paragraph break?
2. What steps did you use for inserting a paragraph break in your document?
3. How did you display and view the changes in your web browser?
4. *** Extra Credit:** What is a horizontal rule **<hr>** tag? a **<br>** tag?

# Independent Practice

Use the **<p>**, the **<hr>**, or the **<br>** tags to create paragraphs or sections in your own document.

---

# Coming Next....

Do you remember your first font?
How about adding ***Style...*** to your text.

**GO TO.... | Lesson Index | previous: "Headings" | next: "Styled Text" |**

---

**Writing HTML:** Lesson 4: Breaking it up into paragraphs
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut4.html

# 5. Doing it with *Style*

Just like a word processor, HTML can tell a web browser to display certain portions of text in *Italic* or **Bold** Style or even a ***combination***.

## Objectives

After this lesson, you will be able to:

- Identify the HTML tags for the text styles: **bold**, *italic*, and typewriter `(mono-spaced)`.
- Enter text in your HTML document in these different text styles and view the changes within your web browser.

---

## Lesson

### HTML Style Tags

HTML offers several tags for adding style to your text. Just remember to be judicious and consistent in the use of styles; too much can make the text uncomfortable to read...

#### Style tags

| HTML | Result |
|---|---|
| `<b>This is Bold...</b>`<br><br>`<i>This is Italic...</i>`<br><br>`<tt>This is Typewriter...</tt>` | |

Note how you can combine the style tags as long as they are correctly nested, the italic tags are both *within* the bold tags. Note also, that the order does not matter.

| HTML | Result |
|---|---|

```
<i><b>This is Bold AND
Italic</b></i>

<b><i>And So is This</i></b>
```

Furthermore, you can also add style to the text that appears in heading tags. Note how the different style tags are opened and closed around the words they style and how the heading tags surround the whole text for the heading.

| HTML | Result |
|---|---|

```
blah blah blah

<h2><i>New</i> and
<tt>Improved!</tt></h2>

blah blah blah
```

## Entering *Styled* Text in Your HTML Document

**Note: If you do not have the working document from the previous lesson, download a copy now.**

Follow these steps to apply style tags to your HTML document.

1. Re-open your workspace (if not already opened).
2. Return to your HTML document, **`volc.html`**, in the text editor.
3. Find the word "volcano" in the first sentence of the Introduction. We are going to make this bold to highlight an important word.
4. Insert the tags to make this word appear as bold text:

```
<b>volcano</b>
```

5. Now we will modify the second paragraph with the **bold** and *italic* tags to emphasize a word. Enter **`<b>...</b>`** and **`<i>...</i>`** tags around the word **`billion`** so this section looks like:

```
<p>
Volcanoes have been a part of earth's history long
```

```
before humans. Compare the history of human beings,
a few million years in the making, to that of the Earth,
over four <b><i>billion</i></b> years in the making.
```

6. Finally, we will use the **`typewriter`**, tag to indicate a special word. Under the **Volcano Terminology** heading, enter the following:

```
The study of volcanoes, or <tt>Volcanology</tt>,
includes many odd terms.
```

7. **Save** in the text editor and **Reload** in your web browser.

---

# Check Your Work

Look at an example that shows these changes. It is important when using style tags to properly terminate the tag(s) with the proper **`</>`** tag. Otherwise, all succeeding text will inherit this text style. It can look bizarre.

# Review Topics

1. What are HTML style tags?
2. What are the different tags used for different styles of text?
3. What steps did you use in entering styled text into your HTML document?
4. ***Extra Credit:** How can these styles be useful in creating a web page or lesson?

# Independent Practice

Try using the style tags for **bold**, *italic*, and `typewriter` to the text of your own web page. See if you can successfully combine styles... that are pleasing to read.

---

# Coming Next....

How to put together lists of items, in both numbered and bulleted fashion.

**GO TO.... | Lesson Index | previous: "Paragraphs" | next: "Lists" |**

---

**Writing HTML:** Lesson 5: Doing it with ***Style***

©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine

Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut5.html

# 6. Lists, Lists, Lists

Lists can present items of information in an easy-to-read format. In fact, there is a list right here, lurking under the next heading!

# Objectives

After this lesson, you will be able to:

- Identify HTML codes for creating unordered, ordered, and nested lists for a web page.
- Place different list types within your HTML document and view the changes within your web page.

---

# Lesson

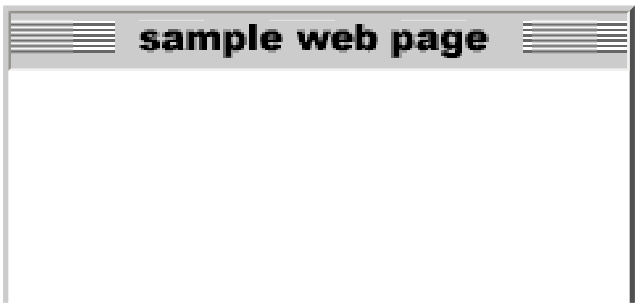
Many web pages display lists of items -- these may be items preceded with a "bullet" (Unordered) or a sequentially numbered list (Ordered).

These lists are easy to format in HTML, and they may even be nested (lists of lists) to produce an outline format. Lists are also handy for creating an index or table of contents to a series of documents or chapters.

## Unordered Lists

Unordered Lists, or **`<ul> .. </ul>`** tags, are ones that appear as a list of items with "bullets" or markers in the front. The bullet marks will depend on the particular version of your web browser and the font specified for displaying normal WWW text (e.g. for Macintosh, the bullets are the option-8 character -- in Times font this is a small square, in Geneva it is a large round dot).

Here is an example of an unordered list:



**My Unordered List:**

- Item 1
- Item 2
- Item 3

And this is the HTML format for producing this format:

```
<b>My Unordered List:</b>
<ul>
  <li> Item 1
  <li> Item 2
  <li> Item 3
</ul>
```

The **<ul>** tag marks the beginning and end of the list, and the **<li>** indicates each **l**ist **i**tem.

### Ordered Lists

Ordered lists are ones where the browser numbers each successive list item starting with "1." Note that the only difference is changing the **ul** tag to **ol** tag.

Using the example from above:

sample web page

**My Ordered List:**

1. Item 1
2. Item 2
3. Item 3

And this is the HTML format for producing this format:

```
<b>My Ordered List:</b>
<ol>
  <li> Item 1
  <li> Item 2
  <li> Item 3
```

```
</ol>
```

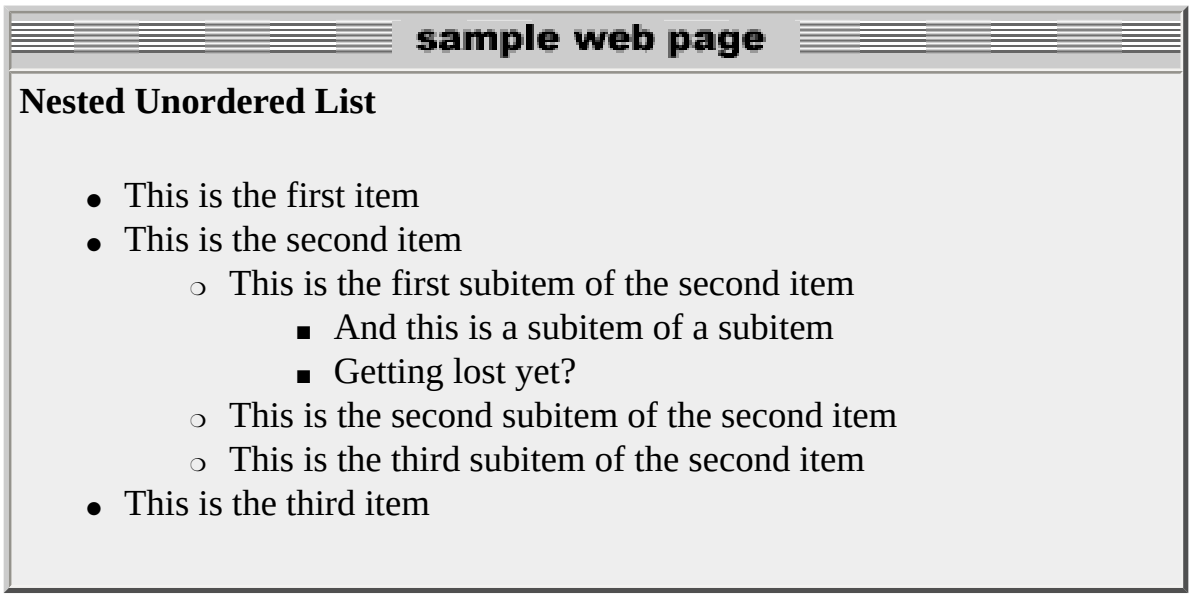
## Nested Lists

Ordered Lists and Unordered lists can have different levels; each will be indented appropriately by your web browser. Your major concern will be to verify that each list is properly terminated and the nesting order is correct.

It can start to look complicated with all of those <ol> <li> </ul> <li> tags floating around, but just try to remember the basic structure:

```
<ul>            <ol>
    <li>           <li>
    <li>           <li>
</ul>           </ol>
```

Here is an example of an unordered list with sublevels of other lists:




Note how the bullet marks change for different levels of the list.

And this is the HTML format for producing this format:

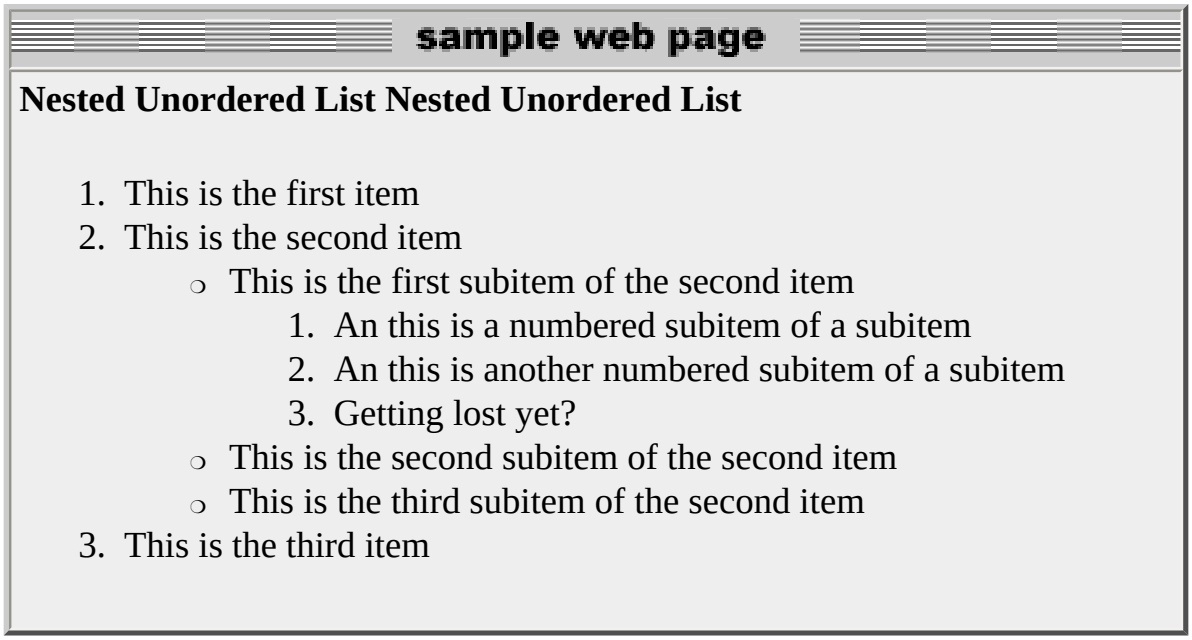
```
<b>Nested Unordered List</b>
<ul>
   <li>This is the first item
   <li>This is the second item
   <ul>
     <li> This is the first subitem of the second item
```

```
      <ul>
        <li> And this is a subitem of a subitem
        <li> Getting lost yet?
      </ul>
      <li> This is the second subitem of the second item
      <li> This is the third subitem of the second item
      </ul>
    <li>This is the third item
  </ul>
```

## Nested Lists -- Mixing them together

Not only can you include ordered lists within ordered lists, but you can also mix and match list types. Hold onto your hats! The HTML starts to look pretty ugly, but watch how lists completely contain other lists.

For example, this ordered list includes a nested unordered list:




And this is the HTML format for producing this format. Note how the HTML has been indented to make it easier to read:

```
<b>Nested Unordered List</b>
     <ol>
        <li>This is the first item
        <li>This is the second item
           <ul>
           <li> This is the first subitem of the second item
              <ol>
```

```
            <li> And this is a numbered subitem of a subitem
            <li> And this is another numbered subitem of a subitem
            <li> Getting lost yet?
            </ol>
         <li> This is the second subitem of the second item
         <li> This is the third subitem of the second item
         </ul>
    <li>This is the third item
</ol>
```

## Placing Lists in Your HTML Document

**Note: If you do not have the working document from the previous lesson, download a copy now.**

Using the **list** tags, you will now add an ordered and an unordered list to your *Volcano Web* page.

1. Re-open your workspace (if not already opened).
2. Open your HTML document in the text editor.
3. Under the **`Volcano Terminology`** header we will use an unordered list to display examples of technical words used in the study of volcanoes. Go to this section in your HTML document.
4. First add the following sentence.

```
How many of these do you know?
```

5. Now enter the HTML format to create the list of terms:

```
<ul>
    <li>caldera
    <li>vesicularity
    <li>pahoehoe
    <li>rheology
    <li>lahar
</ul>
```

6. Now we will use an ordered list to define the required parts of the assignment in this lesson. Under the **`Research Project`** heading, enter the following: (HINT -- this might be a good time to copy and paste from the web page, unless you enjoy typing in text!)

```
Your mission is to find information and report on a volcano,
other than the ones listed above, that has erupted in the last
100 years. Your reports must include:
<ol>
```

```
    <li>Type of volcano
    <li>Geographic location
    <li>Name, distance, and population of nearest major city
    <li>Dates of most recent and most destructive eruptions.
    <li>Other events associated with the recent eruptions
    (earthquakes, floods, mudslides, etc)
</ol>
<p>
Then, write a one page description on the major hazards to humans
in the vicinity of this volcano. Speculate on what you would do
if you were in charge of minimizing the risk to the population.
```

7. **Save** your HTML file and **Reload** in your web browser.

---

# Check Your Work

You may want to compare your document with a sample for this section. If your list is different than the list in the sample, review how you entered your list in your text editor. Make sure it matches the instructions in the **Placing Lists in Your HTML Document** section of this lesson.

# Review Topics

1. How are lists valuable in a web page?
2. What is the HTML tag for a unordered list?
3. What is the tag for a ordered list?
4. How might you set up a nested list?
5. What steps did you use in adding a list to your HTML document?

# Independent Practice

You may want to experiment with changing the ordered list you created to one that is unordered. Also, insert an ordered or an unordered list of items to your own WWW document. Be sure to verify that it displays correctly in your web browser. Can you create a list that includes sub-lists?

---

# Coming Next....

Making it more than just plain text -- adding pictures to your page.

**GO TO.... | Lesson Index | previous: "Stylized Text" | next: "Graphics" |**

---

**Writing HTML:** Lesson 6: Lists, Lists, Lists
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut6.html

# 7. Graphics à la "the Web"

Sending text over the Internet is just old fashioned e-mail. People have been doing it for decades! When you can include **Pictures**, your message can be much more informative! *Is the going rate still 1000 words per picture?*

# Objectives

After this lesson, you will be able to:

- Identify the graphic formats for the World Wide Web.
- Discuss key points to consider when including graphics in WWW documents.
- Download a graphic file to your computer.
- Use the correct HTML format for including pictures in your web page.

# Lesson

Lean back and relax! This lesson is mostly an introduction to graphics for the Web. But we'll have you do a little activity below.

## The Web's Graphic Format

There are numerous file formats for computer graphics... PICT, GIF, TIFF, PNG, not to mention EPS, BMP, PCX, JPEG...

It sounds like cryptic poetry. Bad poetry. Geek poetry!

The way a web browser displays graphics in HTML format indicates the location of a graphic file in a single format that can be interpreted by different types of computers. For example, when the information in that format is received by your Macintosh computer, the web browser knows to display it as a picture format for Macintosh. However, when that same information is received by your Windows browser, it is displayed as a Windows graphic.

In technical jargon, we would say that this picture format is **platform independent.** HTML itself is platform independent, since plain text characters can be understood by any computer.

The standard format that can display **within** a web page is GIF or Graphics Interchange Format. The GIF

compresses the picture information (reduces the file size) and translates it to binary code that can be sent over the Internet. GIF compression is most effective on graphics that have contiguous areas of solid color, and compression is even greater when the color is continuous in the horizontal direction. GIF images have the feature of defining a color to be "transparent" so images can appear to have non-rectangular boundaries. They can also be saved in the "interlaced" format so that when you see a web page, the images start to appear soon and "dissolve" to the final image.

The other file format used on the web is JPEG (named after the Joint Photographic Expert Group that designed this format). In the early web years, JPEG images were **not** displayed in the page but were displayed in a separate window, using an external "helper" application. But most web browsers these days support JPEG images to be displayed right in the web page too.

JPEG compression is very effective for photographic images where the colors can vary spatially over short distances ("grainy" images). JPEG offers some dramatic compression in filesize, sometimes by a factor of 10 (e.g. a 1500 kb file reduced to 150 kb), which may be at a trade-off for some image quality. JPEG images do not have the ability to have transparency.

For more information about these file formats, see the SITO page on Graphics File Compression. If you are in the mood for a great book, try Lynda Weinman's Designing Web Graphics.

More and more graphics programs have built-in features to save files as GIF format. Newer ones such as ImageReady from Adobe and Fireworks from Macromedia have been specifically designed for creating web graphics. You can find other shareware programs/utilities for converting graphics to web format from download.com.

## Some Points to Consider When Using Graphics

For this tutorial, you do not have to use one of these graphics programs. We will show you how to get a copy of the images that you will need.

However, as you begin to develop your own web pages, you should become familiar with creating pictures in either GIF or JPEG format. If your web pages include graphics, consider the following:

- Large and numerous images may look great on a high-end computer, but they will frustrate users who must wait for images to be sent over the network. As a suggestion, keep the total file size of all images on a web less than 100k (we aim for less than 50k each).
- Not all of us have a 21-inch computer monitor! Keep graphic images no wider than 480 pixels and no higher than 300 pixels to avoid forcing users to scroll or resize their web browser window.
- Color gradients may look pretty but for GIF images they do not compress as much as solid color areas and they can sometimes come out "banded".
- Some graphics programs offer options for "no dithering" when converting to GIF -- this can

reduce the amount of "noise" in a solid background.
- Many dark grey tones on Macintosh computers are not discernible on Windows computers.
- Rather than displaying all of the images on the web page, have them linked as external images that are downloaded only when a viewer clicks on a hypertext item (this will be covered later in lesson 8a and in lesson 8d). If you have numerous pictures to display, try to break the web page into a series of linked pages.
- A single image (e.g. a small "bullet") can appear several times in a web page with little added delay each time you use that same image.
- Many web browsers "cache" images (storing them on your computer) meaning that using the same file in several web pages will load them from the viewer's computer rather than loading them across the Internet.
- Most importantly, make sure that the images are ones that **add** meaning to your HTML documents.

You may design a beautiful web page, loaded with large pictures, that may load nicely from your computer, but may be excruciatingly slow by a viewer using a slow modem over a busy network. The 'net is a busy place and getting busier every second.

### Saving and Including Pictures in Your Web Page

For the next lesson you will first need to download a copy of a GIF image of a volcano (watch out for that hot lava!).

Just follow the instructions on the Lesson 7 Image Studio and then return here to complete this lesson.

---

# Check Your Work

Check to see that the file, **`lava.gif`**, is saved in the same directory/folder as your HTML file, **`volc.html`**. If it is not there, check to see if you accidentally saved it in another directory/folder. Then, move it to the correct location.

# Review Topics

1. What are the two graphic formats used for the World Wide Web?
2. How can a graphic file display on different computers?
3. What are some key points to consider when including graphics in web pages?
4. How did you save the lava graphic for use in your WWW document?

# Independent Practice

Surf the web and browse for pictures. Try to download at least one image that might be useful for your page. Just a few places you might try:

- ClipArt Connection
- CoolText Online Graphics Generator
- Ditto.com
- Yahoo's Clip Art
- Lycos' Picture Search
- Yahoo's ImageSurf
- The Free Site
- Free Graphics Store
- Barry's Clip Art Server

---

# Coming Next....

You have the image... Now, how in the **H T M L** do you display it in your document?

**GO TO.... | Lesson Index | previous: "Lists" | next: "Inline Images" |**

---

**Writing HTML:** Lesson 7: Graphics à la the Web
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut7.html

# 7a. Inline Graphics

WWW Mathematics:

```
Text + Pictures = Multimedia
Multimedia + WWW = Global HyperMedia
```

Got it?

# Objectives

After this lesson, you will be able to:

- Place an inline image within your HTML document.
- Select how the pictures align with surrounding text.
- Modify the inline image tag to account for viewers using a non-graphic browser.
- Specify the dimensions of inline image.

---

# Lesson

Let's see how with HTML you can include pictures like the "Big M" in a web page...

## HTML Tags for Inline Graphics

An "inline" image is one that appears within the text of a WWW page, such as this picture of "Big M".

The HTML format for the inline **`image`** tag is:

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

where **filename.gif** is the name of a GIF file that resides *in the same directory/folder* as your HTML document. By "inline", this means that a web browser will display the image in between text.

Note how the text immediately follows the "Big M" above. What if we want the "Big M" sitting on its very own line? To force the image to appear on a separate line,

simply insert a **paragraph** tag before the **image** tag:

```
<p> <img src="filename.gif">
```

## Alignment of Text and Inline Graphics

With an attribute to the **<img...>** tag, you can also control how text adjacent to the image aligns with the picture. The **align** attribute, added inside the **<img>** tag, can produce the following effects:

---

# align=top

```
<img align=top src="filename.gif">
```

---

# align=middle

```
<img align=middle src="filename.gif">
```

**sample web page**

**is for Maricopa Community Colleges** located in the Valley of the Sun, metropolitan Phoenix, Arizona. Our license plates say that we are the Grand Canyon State...

---

## align=bottom (default)

```
<img align=bottom src="filename.gif">
```

**sample web page**

**is for Maricopa Community Colleges** located in the Valley of the Sun, metropolitan Phoenix, Arizona. Our license plates say that we are the Grand Canyon State...

---

**Note how the text aligns only for one line... (shrink or stretch the width of your WWW browser window to see what happens.) In a later lesson, we will see a way to align blocks of text so that they flow around the side of an image.**

## Placing an Inline Image in Your HTML document

**Note: If you do not have the working document from the previous lesson, download a copy now. You will also need to have the GIF image available from the Lesson 7 Image Studio.**

In this exercise, you will add an introductory picture of a volcano to your lesson.

1. Re-open your workspace (if not already open).
2. Open your **`volc.html`** document with the text editor.
3. Above the **`<h1>Volcano Web</h1>`** heading, enter the following:

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

This HTML format will insert, at the very top of your page, the lava picture that you downloaded in the previous lesson.

4. Save and reload in your web browser.

In placing the image, you may have wondered why we did not need to put a **<p>** tag after the image. This is because the following text was a header. A web browser **always** inserts a paragraph break before and after an <h1,h2,h3...> tag.

## The `alt="..."` attribute

If your web pages will be viewed by users using a text-only browser (such as lynx), they will not be able to view any inline images. Or sometimes, users will turn off the loading of inline images to save time on downloading over slow network connections. An attribute for the **<img ...>** tag allows for substitution of a descriptive string of text to hold the place of the image.

Under these conditions, a viewer with a text browser will see a place holder so that the top of our lesson page looks like:

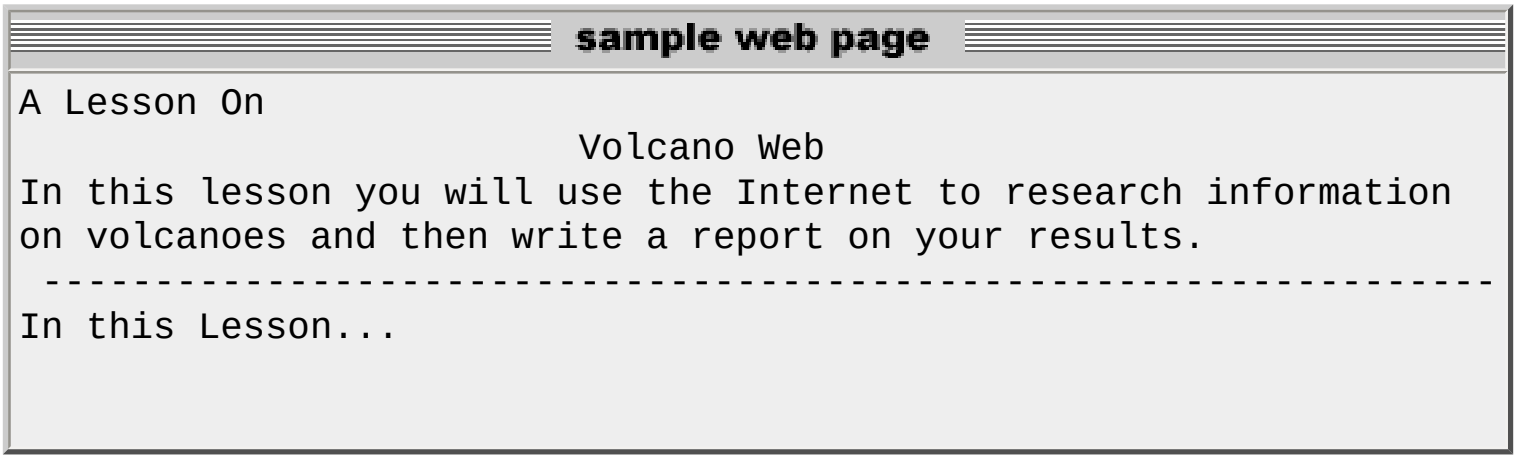
**sample web page**

```
[IMAGE]
                          Volcano Web
In this lesson you will use the Internet to research information
on volcanoes and then write a report on your results.
 ------------------------------------------------------------
In this Lesson...
```

This lets the viewer know that there is a graphic inserted at the top of this page. You could modify the **<img>** tag so that rather than using the place holder, it displays a text string. For example, in our lesson we could add "A Lesson on:" by modifying the **<img>** to read:

```
<img alt="A Lesson on:" src="lava.gif">
```

The **alt="..."** attribute replaces the place holder with a text string so that from a text-only browser (or when loading of images is shut off), it would now appear:




At this time, make this same edit to your HTML file for the **<img>** tag that displays the picture of the volcano.

# Height and Width attributes

Another option you may want to include in your **<img...>** tags are two attributes that give the dimensions of the image in pixels. Why? Normally, your web browser has to determine these dimensions as it reads in the image; the loading of your page can be faster if you specify these numbers in your HTML.

The format for including this option is:

```
<img src="filename.gif" width=X height=Y >
```

where **X** is the width of the image and **Y** is the height of the image in pixels.

You can usually use some sort of graphics program or utility to determine these numbers. Another way to find the dimensions of an image is to load it into your web browser -- you may be able to drag and drop the icon for the image into your browser window -- and the height and width will be displayed in the title bar of the browser.

For our example in this lesson, the **lava.gif** image is 300 pixels wide and 259 pixels high. So you should edit your **volc.html** file to read:

```
<img alt="A Lesson on:" src="lava.gif" width=300 height=259>
```

**NOTE: the order of the attributes inside the <img> tag does not matter.**

Often we are asked if you can alter the size of the image by inserting numbers other than the actual dimensions of the image. The answers is **yes** but the results may be undesirable. If you insert larger

numbers (to make the image bigger) the result will be a "blocky" picture. Sometimes this can be a useful effect on images with large areas of solid color. Take a look at our example of Going from Small to Big. If you use lower numbers (to make the image smaller) the result may be a distorted picture. Also, the full size image still has to be downloaded, so there is no real savings in terms of time to download the image. Any re-sizing of the image requires extra "work" by the web browser to recalculate the page layout.

You could experiment and see for yourself. We just might be wrong!

You can also specify the size of an inline image in dimensions that are percentages of the current browser window size, so that the image will resize itself if the viewer expands or reduces the size of their browser window. Take a look at our example of Percentage Scaling. *Caveat Emptor!* This may not work on all web browsers!

---

# Check Your Work

You may want to compare your work to this sample of how this document should appear. If your document is different from the example, check how you entered the image format in your text editor. Make sure you entered it as instructed in the **Placing an Inline Image in Your HTML Document** section of this lesson.

If you see a picture icon with a question mark:

first check that the HTML file and the image are in the same folder/directory. Then, you may want to make sure that the file name entered in the **`<img... >`** tag matches the name of the file.

> **NOTE: Some computer systems (UNIX) are case sensitive for the names of files, meaning that the file `"lava.GIF"` is NOT the same as `"lava.gif"`. Other computers (Macintosh) consider them as the same files. Even if your computer does not differentiate between capital and small case, we suggest that you be consistent in the naming of files and how they are referred to in your HTML and use all lower case letters. (Many WWW servers run UNIX).**

# Review Topics

1. What is the HTML format for an inline image?
2. What type of tag must you put before an image tag to make the image appear on a separate line?
3. How did you add the lava picture to your document?
4. What does the **`alt="...."`** attribute do? What does the **`height="...."`** attribute do?

# Independent Practice

Add an inline image to your web page using a GIF picture file that is stored on your computer or one that you have downloaded from the Internet.

---

# Coming Next....

Create links to other documents and files that you create as well as ones on the Internet.

**GO TO.... | Lesson Index | previous: "Graphics" | next: "Linking with Anchors" |**

---

**Writing HTML:** Lesson 8d: Links to Sections of a Page
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut7a.html

# 8. Linking it with Anchors

Relax... this lesson is quick and easy! In fact, it is just information for you to read...

## What is a URL?

The real power of the web is the ability to create hypertext links to related information. That other information may be other web pages, graphics, sounds, digital movies, animations, software programs, contents of a file server, a log-in session to a remote computer, a software archive, or an "ftp" site.

The World Wide Web uses an addressing scheme known as **URL**s, or **U**niform **R**esource **L**ocators (sometimes also called "Universal Resource Locator"), to indicate the location of such items. These hypertext links, the ones usually underlined in blue, are known as **anchors** (This should not be news to you as you followed several to get this far!).

In the next lessons we will:

- Review the concept of URLs.
- Find and copy URLs from your web browser to your HTML text document.
- Write an HTML anchor to link to another document in the samedirectory as our first document.
- Write an HTML anchor to link to another document in a different directory as our first document.
- Write an HTML anchor to link to another web document on the Internet.
- Write an HTML anchor that links to another section of the same document.
- Incorporate a graphic that acts as a "hyperlink" to another document.

Wow! That sounds like a lot to do! Don't worry -- it is no more complex than what you have done up to this point.

After all, without the hypertext, we would be only calling this **"Writing TML"** and not **Writing HTML**

---

## Coming Next....

Using URLs to connect documents together via hypertext links.

---

**Writing HTML:** Lesson 8: Linking it with Anchors
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut8.html

# 8a. Linking to Local Files

Can my document talk to my document? Well, they can at least be linked!

## Objectives

After this lesson, you will be able to:

- Create a link to an HTML document in the same directory/folder as your main document.
- Create a link to display a graphic image.
- Create a link to a file in a different directory/folder than your main document.
- Reorganize the structure of your web.

---

## Lesson

Now, you will take your first step of "anchoring" by creating a hypertext link to a second web page. These links are called "local" because they reside on the same computer as the working document (they do not have to venture out on the Internet). You will also be shuffling around the parts of your growing web site (do you see how this becomes more than just a "home page"?).

**Note: If you do not have the working document from the previous lesson, download a copy now.**

### Link to Local Files

The simplest **anchor** link is one that opens another HTML file in the same directory as the presently displayed web page. The HTML format for doing this is:

```
<a href="filename.html">text that responds to link</a>
```

Think of it as "a" for **a**nchor link and "href" for "**h**ypertext **ref**erence".

The filename must be another HTML file. Whatever text occurs after the first **>** and before the closing **</a>** symbols will be the "hypertext" that appears underlined and "hyper."

Now follow these steps to build an anchor link in your HTML document to a local file:

1. Open your HTML document, **volc.html**, in the text editor.
2. First, under the **Volcanic Places in the USA** heading, enter the following text which introduces the two volcanoes discussed in later sections.

```
Listed below are two places in the United States that are
considered "active" volcanic areas.
```

3. Below the "Mount St. Helens" heading, enter:

```
On May 18, 1980, after a long period of rest, this quiet
mountain in Washington provided <a href="msh.html">
detailed observations</a> on the mechanics
of highly explosive eruptions.
```

**The text "detailed observations" will link the viewer to a second HTML document called `msh.html`. This second HTML file does not yet exist; we will construct it in steps (5) and (6).**

4. Save and close your HTML document
5. Now, with your text editor, open a window for a **New** document.
6. Enter the following text in the new window:

```
<html>
<head>
<title>Mount St Helens</title>
</head>
<body>
<h1>Mount St Helens</h1>
The towering pine trees of this once-quiet mountain
were toppled over like toothpicks.
</body>
</html>
```

7. Save this file as **`msh.html`** in the same directory/folder as your working HTML file (**`volc.html`**).
8. **Reload `volc.html`** in your web browser.
9. Test the hypertext link for the words "detailed observations". When selected, it should connect you to the new page about Mount St. Helens.

## Anchor Link to a Graphic

In lesson 7a, we learned how to display an "inline" graphic that would appear in your web page. With the **`anchor`** tag, you can also create a link to display a graphic file. When the anchor link is selected, it will download the image file and display the image by itself in an empty page.

The simplest anchor link is to a file in the same directory/folder as the document that calls it. The format for creating a hypertext link to a graphic is the same as above for linking to another HTML document:

```
<a href="filename.gif">text that responds to link</a>
```

where **`filename.gif`** is the name of a GIF image file.

Now follow these steps to add a link to a graphic file in your HTML document:

1. Download a copy of a GIF image from the Lesson 8a Image Studio.
2. Open the **msh.html** file in the text editor.
3. Modify the text to include a link to the image of Mount St. Helens.

```
The towering pine trees of this once-quiet mountain
were <a href="msh.gif">toppled over like toothpicks</a>.
```

4. **Save** the **msh.html** file and **Reload** in your web browser
5. Now click on the link you just created in step (3).
6. A picture of blown down trees should be displayed.

## Links to other directories

The **anchor** tags can also link to an HTML document or graphic file in another directory/folder in relation to the document that contains the anchor. For example, in our lesson, we may wish to keep all of the graphics in a separate directory/folder called **pictures**. As you create more and more HTML files, keeping the image files in its own area will make things a bit more organized for you. Let's do that now:

1. From your computer system, create a sub-directory/folder called **pictures** in the same location where your **volc.html** file is stored.
2. Move the **msh.gif** file to this new sub-directory/folder.
3. Open the **msh.html** file in your text editor.
4. Edit the **anchor** tag for the graphic to read:

```
The towering pine trees of this once-quiet mountain
were <a href="pictures/msh.gif">toppled over
like toothpicks</a>.
```

**NOTE: With HTML you can direct your web browser to open any document/graphic at a directory level *lower* (i.e. a sub-directory or folder within the directory/folder that contains the working HTML file) by using the "/" character to indicate the change to a sub-directory called "pictures."**

---

**work area**

**volc.html**

**pictures**

**msh.gif**

---

5. Save the HTML document and **Reload** in your web browser.

If all went well, the link in the sentence describing the blown-down trees should now call up the graphic file stored in the **pictures** sub-directory/folder.

# Anchor Links to a Higher Level Directory

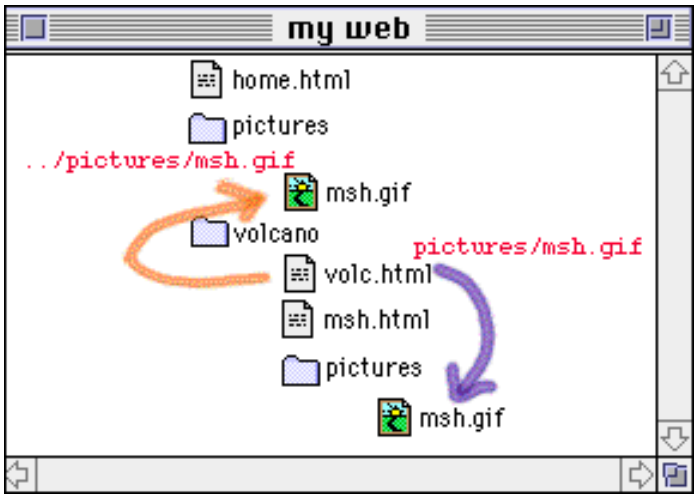
The types of links we have constructed here are known as "relative" links, meaning a web browser can construct the full URL based upon the current location of the HTML page and the link information in the **<a href=...>** tags. This is very powerful because you can build all your web pages on one computer, test them, and move them to another computer -- all the relative links will stay intact.

In this lesson we saw how to construct a hyperlink to a document that is stored in a directory **lower** than the working HTML page. Note that you can also construct a link that connects to a **higher** level directory as well by using this HTML:

```
<a href="../../home.html">return to home</a>
```

Each instance of **"../"** the URL of an anchor link tells the web browser to go to a higher level directory/folder relative to the current page; in this case to go up two directory/folder levels and look for a file called **home.html**.

In our example, let's say that our **pictures** sub directory was not in the same directory/folder as the **volc.html** file but was actually one level higher.




In the previous section we constructed a link from the **volc.html** file to the **msh.gif** file in a subdirectory:

```
<img src="pictures/msh.gif">
```

Now, we want to reorganize our web structure so that the pictures folder/directory is at a higher level. The link is now written:

```
<img src="../pictures/msh.gif">
```

so the web browser looks for a folder called "pictures" that is stored one level up from our **volc.html** file.

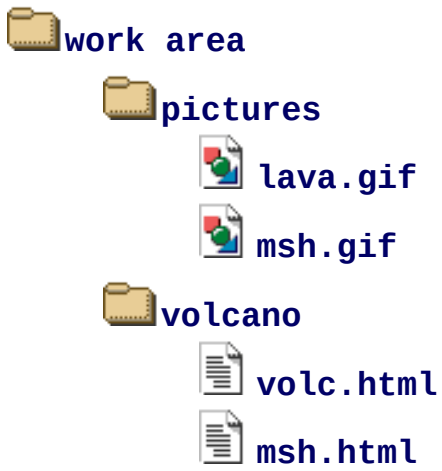
An advantage of this structure is that it would be easier to store a large number of graphics in this upper folder/directory that can be shared in other web pages. We may do another lesson on landforms that makes use of the pictures stored in this folder/directory.

So now it is time to do a little re-organizing of our HTML files. This requires that you are familiar with moving files and directories around on your computer. **Read this carefully! It may be feeling like it's getting complicated, but it will all be clear soon!**

1. First, create a new folder/directory and name it **volcano** (it is recommended to keep the file names in all lower case).

2. Now, move the two HTML files **volc.html** and **msh.html** into this new folder/directory.
3. Move the **pictures** folder/directory (along with the **msh.gif** file inside) so that it is in the **same** level as the new **volcano** folder/directory. Also, move the **lava.gif** file that we added in lesson 7a into the pictures folder.
4. So your entire **workarea** directory should now contain two subdirectories -- one that holds your HTML files (**volcano**) that holds the graphics (**pictures**):

---




---

5. We've moved some things around so now we will have to update the anchor links in our HTML files. First, look at the first local link we built in the **volc.html** file:

```
<h3>Mount St Helens</h3>
On May 18, 1980, after a long period of rest, this quiet
mountain in Washington provided <a href="msh.html">detailed
observations</a> on the mechanics of highly explosive
eruptions.
```

**NOTE: Since the msh.html file is still in the same relative directory as volc.html, we do not have to change any of this HTML! Can you see how relative file linking is one of the powerful features of HTML?**

6. But now let's look at the link to the picture of Mt. St Helens that we created in the **msh.html** file:

```
The towering pine trees of this once-quiet mountain
were <a href="pictures/msh.gif">toppled over
like toothpicks</a>.
```

Open this file in your text editor and edit the link to read:

```
The towering pine trees of this once-quiet mountain
were <a href="../pictures/msh.gif">toppled over
like toothpicks</a>.
```

This relative link tells the web browser to go up one level from the current folder/directory (**volcano**) and look there for another folder/directory called **pictures** that contains a GIF image called **msh.gif**

7. You will have to update the **<img...>** tag that displays the title graphic. Open the **volc.html** file in your text editor and modify the line just below the **<body>** tag to read:

```
<img alt="A Lesson on:" src="../pictures/lava.gif" width=300 height=259>
```

8. Save your file. You should then **Open** the **volc.html** file in your web browser and test the link to **msh.html** and then try the link to the picture of Mount St Helens.

# One More Small Change

This last small step may not be obvious, but we will explain it shortly. The last thing you should do in this lesson is to change the name of your working file from **volc.html** to **index.html**. You should do this using the normal way of editing a file's name from the computer desktop (on the Macintosh click on the file name; on Windows right-mouse click on the icon and select the option for **Rename**). Note also for Windows users that if you use a special editor program to create HTML files, you will not see the ".html" extension on the desktop file name, so in that case, you would change the file name from **volc** to **index** because under the hood, the computer knows that there is a ".html" at the end.

Why are we doing this? Let's say you have finished this lesson and are ready to store it on a World Wide Web server for the world to see. And let's assume that the Internet address for this server at Big University is:

```
http://www.bigu.edu/
```

And your file will be stored in a series of directories:

```
--= top level of server: www.bigu.edu
   /courses
        /science
            /geology
                /volc.html
```

so that the URL for the Volcano Web might be:

```
http://www.bigu.edu/courses/science/geology/volcano/volc.html
```

Pretty long, eh? Now here is the promised explanation -- on most WWW servers you can designate one standard name that is the "default" web page for that directory and on most systems that name is.... **index.html**. What this means is that the Internet address:

```
http://www.bigu.edu/courses/science/geology/volcano/
```

is equivalent to

```
http://www.bigu.edu/courses/science/geology/volcano/index.html
```

This might make you think that it is a lot of energy to cut 20 letters out of a URL! But it does tend to make your URL

look a bit more professional -- If you were creating the Longhorn Cheese Home page,

```
http://www.cheese.com/longhorn/
```

looks less redundant in print than

```
http://www.cheese.com/longhorn/longhorn.html
```

which comes into play when people read about your URL and are trying to connect by typing it into their web browser.

Note also that this special file name **`index.html`** is used on a most web servers but it might also be **`default.htm`**-- check with the people that run your web server.

---

# Check Your Work

Compare your web page with a sample of how this document should appear. You will first see your *Volcano Web* page. When you click on the hypertext for **`detailed observations`**, your web browser will display a new page. Finally, when you click on **`toppled over like toothpicks`**, your web browser will display in an external window a picture file that is stored in a sub folder/directory.

Use the web browser's **back** button twice to return to this page. If your web page was different from the sample, review the text you entered in the text editor.

# Review Topics

1. What were the steps you used in creating a link within your document to a local file?
2. What steps did you use to create a link which displayed a graphic in an external window?
3. How did you create a link to a file in a lower directory/folder than your main document? a higher directory?
4. What is the significance of a file called **`index.html`** on a WWW server?

# Independent Practice

Create a second HTML document that uses the HTML formatting that you are familiar with at this point. Return to the first one you created and make an anchor that links to this new one.

---

# Coming Next....

Wow! That was a lot of work! In the next lesson you will learn how use the HTML for linking to resources "out

there" on the Internet.

**GO TO.... | Lesson Index | previous: "Anchors" | next: "URLs" |**

---

**Writing HTML:** Lesson 8b: Linking to Local Files
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut8a.html

# 8b. URLs: Pointers to the Internet

## URL? Earl? Yurl? hurl? gyrl?

It's getting time to link to that Big Wide Web using the web's addressing scheme.

## Objectives

After this lesson you will be able to:

- Identify the function of Uniform Resource Locators (URLs).
- Recognize the structure of a URL.
- Examine the URLs in the hypertext links of any web page.

---

## Lesson

**Note: For this lesson, you will not need your HTML text file.** This is another low-effort lesson!

### What is a URL?

The Uniform Resource Locator (URL) is what the WWW uses to find the location of files and documents from computers on the Internet. On your web browser screen, the URL for this document is typically displayed in the upper part of the Web browser window. The URL includes:

- an identifier for the type of Internet server;
- an Internet address; and
- a file path to the particular item of interest.

The URL is what you will need to build a link from the web page that you are creating to connect to some other piece of information available on the Internet. For more information, see Curling Up To URLs (v0.2)

### How are URLs Structured?

The structure of a URL is:

```
type://in.ter.net.address/directory/sub-directory/.../filename
```

The **"type"** indicates the type of Internet server being accessed:

**http**
: a web server, "HttP" stands for HyperText Transfer Protocol

**gopher**
: an Internet Gopher site or menu driven directories of files and information

**ftp**
: an anonymous File Transfer Protocol (FTP) site, archives of files.

**telnet**
: initiates a Telnet session to log on remotely to another computer When selected, your web browser will launch a Telnet external program and connect to the specified site.

**WAIS**
: Wide Area Indexed Server -- a site to search a collection of subject oriented documents by keywords

**file**
: A file on your local computer system (hard drive, floppy, local file server)

The **type** is always followed by **"://"** and the Internet address of a remote computer. This is in the structure of:

```
host.domain.domain.domain
```

For example:

```
machine.department.college.edu
123.45.6.78
office.company.com
agency.branch.gov
machine.organization.country
```

If the URL is to the main level of this host (its "home page"), then the URL is terminated with a slash "/". If you are linking to a sub-directory or a file, you must also add the exact path to that item using the slash character to indicate the entire file path.

**Note: For most web servers spelling does count! So does capitalization! File names on UNIX computers are case sensitive, meaning that a file named**

```
SpiffyText.html
```

**is a different file than**

```
spiffytext.html
```

## Experimenting With URLs

Note that URLs can link to any site, directory, subdirectory, text file, image, digital movie, or sound file on any Internet site that is set up for public access. The best way to see different URLs is just to move your mouse over any hypertext link in any web page -- if you look at the bottom of your web browser, it should display the URL that you would connect to if you clicked on that link. You could go to a big site such as Yahoo and "peek" at URLs (did you see the URL for Yahoo when you moved your mouse over the link in the this sentence?)

Here is an easy way to copy a URL for a link in any page. You first must access the "secret" pop-up menu from any hypertext link in a web page -- click and hold the mouse for Macintosh; click the right mouse button for Windows and Unix. From this menu, select **Copy This Link Location** (or similar menu item). After releasing the mouse button, jump to any text document and select **Paste** from the **Edit** menu. Voilà! You've just nabbed a URL from a link in the web page (this way, you can copy a URL without even visiting the page it links to!)

---

# Review Topics

1. What purpose do URLs serve for the World Wide Web?
2. Where are URLs found on a WWW screen?
3. What is the basic structure of an URL?
4. How can you see the URLs that a hypertext link will jump to?

# Independent Practice

Find some sites on the Internet that intrigue you. For each one, record its name and its URL displayed near the top of your browser window. See if you can copy and paste the URLs into a text document. You will use this list later to add links from your own web pages to these sites that you found.

# Coming Next....

You will use URLs in **anchor** tags to create links to files on the Internet for your *Volcano Web* page.

**GO TO.... | Lesson Index | previous: " Links to Local Files" | next: "Links to the Internet" |**

**Writing HTML:** Lesson 8b: URLs: Pointers to the Internet
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut8b.html

# 8c. Links to the World: Internet Sites

You've been revving your engines, and itching to hit the Information SuperHighway, right? Here, we will extend our use of **anchor** tags to create links to resources out there on the Internet.

## Objectives

After this lesson you will be able to:

- Create an anchor to link to an Internet site.
- Quickly copy the URL for a site and use it in your HTML document.

---

## Lesson

**Note: If you do not have the working document from the previous lesson, download a copy now.**

### HTML for Anchors to the Internet

Linking to a site on the Internet combines what we have worked on earlier in the lessons on **Links to Local Files** (Lesson 8a) by incorporating what we have learned about **URLs** (Lesson 8b). The full HTML format for an **anchor** link to an item on the Internet is:

```
<a href="URL">Text to Activate Link</a>
```

where **URL** is the full Uniform Resource Locator, in quotes, the address for the Internet site where you want to send the viewer. The string **Text to Activate Link** is what will show up as hypertext in your web browser (usually but not always) underlined and in blue. When a viewer clicks on this hypertext, the web browser will link them to the Internet site indicated by the URL. Remember that a URL can be a link to another World Wide Web (WWW) server, Gopher server, FTP site, or any text, graphic, sound, video file on these servers.

Now, we will add a hypertext link to a site that has information about volcanoes on the planet Mars. Follow these directions to add anchor links on your HTML document:

1. Open your **index.html** file in the text editor.
2. Below the heading, **Volcanic Places on Mars**, enter the following text:

```
<a href="http://solarviews.com/eng/mars.htm">
Mars</a> has its fair share of volcanic landforms,
including the largest known volcano in the solar system,
<a href="http://solarviews.com/r/mars/olympus.jpg">
Olympus Mons</a>
```

> **Note: We've made a link to two different types of information. The first hyperlink connects to a web page that describes information about the planet Mars. The second is a link to a large JPEG image of a Martian volcano.**

3. **Save** and **Reload** in your web browser.

> **Note: We have shown you how to link directly to an image from another web server. You could quite easily use an off-site URL in your IMG tags for your own web pages. We strongly urge you to contact the site's creator and ask permission. See an example of an e-mail request we made for this tutorial.**
>
> **In some cases, web site owners are penalized or charged for excessive accesses. For more information on this issue, see a Plea from the Widows Web.**

## A Quick 'n Easy Way to Enter URLs in Anchor Tags

As you navigate among different web pages, the URL of the currently visible page can be viewed at the top of the web browser window (You may have to look for a menu option to **Display URLs**). For example, in this document, the URL looks something like:

```
http://www.mcli.dist.maricopa.edu/tut/tut8c.html
```

You can use your mouse to select and **copy** a URL from the web browser display and then paste it in the **anchor** tag of your HTML document. This is much more efficient than writing URLs down on paper (some are quite long!).

Now we will add some links to other sites that we will list under the **References** section of our Volcano lesson. One such site that might offer relevant information is the US Geological Survey.

Follow these steps:

1. Open your HTML document **index.html** in the text editor

2. Under the heading "References", enter:

```
<ul>
<li> <a href="">Educational Resources from the
USGS</a>
</ul>
```

> **NOTE: We've constructed the hypertext link but we still need to enter a URL between the quotes.**

3. Connect to the US Geological Survey Education Index.
4. From the web page, use the mouse to **Select** the URL as displayed in the URL display field.
5. From the **Edit** menu, **Copy** the URL.
6. Now, return to your HTML document **`index.html`**
7. Click the mouse once between the two quote marks you inserted in step #2 and **Paste** the text you copied in step #5. The final **`anchor`** tag should look like:

```
<a href="http://info.er.usgs.gov/education/index.html">
Educational Resources from the USGS</a>
```

> **Note: You have just set up the HTML structure for an Unordered List, with each list item a hypertext link to a site that offers information about volcanoes. For a review of lists, see lesson 6**

For additional practice, explore some of the following Internet sites with resources on Geology and/or Volcanoes. Copy the URLs and construct the hypertext links to at least two more sites in your **References** section:

**Sites with Information on Volcanoes:**

Alaska Volcano Observatory * Cascades Volcano Observatory * Michigan Tech Volcanoes Page * NASA Earth Observing System (EOS) IDS Volcanology Team * Volcano Information Center * Volcano/Earth Science-Oriented Servers * Smithsonian Global Volcanism Program (GVP) * Volcano Watch Newsletter * Volcanoes Online

**Sites with General Information on Geology:**

Yahoo: Earth Science * Earth Resources Observation Systems (EROS) Data Center * Tradewave Galaxy * NASA Observatorium * United States Geological Survey * WWW Virtual Library * US Geological Survey Volcanic Hazards Program

---

# Check Your Work

Compare your web page with a sample of how this document should appear. If your web page was different from the sample or the links do not properly connect to a remote site, review the text you entered in the text editor. Note that your list of references may be different from the example.

# Review Topics

1. What is the address for an item on the Internet?
2. What steps did you take for creating an anchor link to the NASA Internet Site?
3. What shortcut was identified for putting an URL into an anchor link?
4. Tell a colleague or friend about any other links that you put in your document.

# Independent Practice

Create anchor links from your own web page that connect to some of the URLs addresses you have discovered.

---

# Coming Next....

In the next lesson you will create links that connect to different sections of an HTML document.

**GO TO.... | Lesson Index | previous: "URLs" | next: "Named Anchors" |**

---

**Writing HTML:** Lesson 8c: Links to the World: Internet Sites
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut8c.html

# 8d. Links to Sections of a Page

You have seen how to link to other web pages, whether they are ones you created or have found elsewhere on the Internet. What if you wanted to connect to a specific section **within** a document? YOU CAN!

## Objectives

After this lesson you will be able to:

- Create a hidden reference marker for a particular section within an HTML file.
- Build a hypertext link that will connect to a particular section within an HTML file.
- Build a hypertext link that will connect to a particular section within another HTML file.
- Create a hypertext table of contents for a web page.

---

## Lesson

**Note: If you do not have the working document from the previous lesson, download a copy now.**

### The Named Anchor

A **named anchor** is a hidden reference marker for a particular section of your HTML file. This might be used to link to a different section of the same page if it is long, or to a marked section of another page. For example, on this page you are viewing, I could create a hidden marker called "check" that marked the heading below "Check Your Work". Then, I write an anchor link that connects to this section of this document. Once you click on a link to this named anchor, your web browser will jump so this line is at the top of the screen.

Here is an example you can try right now. Go to Check Your Work. When you get there look for a link that will return you to this very spot.

The HTML format for creating the named anchor is:

```
<a name="NAME">Text to Link To</a>
```

or for the link you just tried above:

```
<a name="check">Check Your Work</a>
```

Notice how this subtly differs from the anchor link **<a href=...** that we learned about in lesson 8a.

## Writing a Link to a Named Anchor

When you want to create a hypertext link (or an "anchor link", see lesson 8a) to a named anchor, use the following HTML:

```
<a href="#xxxxx">Text to act as hypertext</a>
```

or for the link you just tried that sent you to the section below:

```
Go to  <a href="#check">Check Your Work</a>
```

The "#" symbol instructs your web browser to look through the HTML document for a named anchor called **"xxxxxx"** or in our example **"check"**. When you select or click on the hypertext, it brings the part of the document that contains the named anchor to the top of the screen.

## Adding Named Anchors to the *Volcano Web* Lesson

Now, we will build a table of contents for our lesson that will appear at the top of our *Volcano Web* page. The entries for this will be the headings we have already created. Each will act as a hypertext link to a particular section of our lesson.

The first step is to create a named anchor for each of the heading sections in your *Volcano Web* lesson:

1. Open your **index.html** file in the text editor
2. Find the heading for the **Introduction**. Change it from:

   ```
   <h2>Introduction</h2>
   ```

   so that it looks like:

   ```
   <h2><a name="intro">Introduction</a></h2>
   ```

   > **NOTE: You have just marked the line that contains the header "Introduction" with a hidden reference marker, the named anchor "intro".**

3. In a similar manner, change all header **<h2>** tags for the other sections:

   ```
   <h2><A NAME="term">Volcano Terminology</a></h2>

   <h2><A NAME="usa">Volcanic Places in the USA</a></h2>
   ```

```
<h2><A NAME="mars">Volcanic Places on Mars</a></h2>

<h2><A NAME="project">Research Project</a></h2>
```

4. If you **Reload** now from your web browser, you will not notice any visible change. The named anchor tags we have just added are hidden from the user's view.

## Adding Links to a Named Anchor in the Same Document

Now we will set up a table of contents that will appear at the top of the screen. We will use an unordered list (see lesson 6 for more on lists) to create this list:

1. If not already open, open your **`index.html`** file in the text editor.
2. Below the first sentence under the **Volcano Web** heading enter the following text:

```
<hr>
<b>In this Lesson...</b>
<ul><i>
<li>Introduction
<li>Volcano Terminology
<li>Volcanic Places in the USA
<li>Volcanic Places on Mars
<li>Research Project</i>
</ul>
```

> **NOTE: This index is marked off above and below by a solid line using the <hr> tag. The *Italic* style is used on the entries for the text. At this point we have only entered the text of the index entries. Below we will add the HTML to make the links active.**

3. **Save** and **Reload** into in your web browser.

Finally, we want to make each item of the list act as a hypertext link to another section of the document. To do this, we will use a variation of the basic anchor link lessons 8a. Rather then linking to another file, we link to one of the named anchors (the hidden markers that you created above) within the same HTML file. We indicate a named anchor by preceding the reference marker name with a "#" symbol:

1. Open your **`index.html`** file in the text editor
2. Go to the first list item in your index section. Change it from:

```
<li>Introduction
```

to look like:

```
<li><a href="#intro">Introduction</a>
```

3. You should now be able to fill in the other links to named anchors so that the section looks like:

```
<hr>
<b>In this Lesson...</b>
<ul><i>
<li><a href="#intro">Introduction</a>
<li><a href="#term">Volcano Terminology</a>
<li><a href="#usa">Volcanic Places in the USA</a>
<li><a href="#mars">Volcanic Places on Mars</a>
<li><a href="#project">Research Project</a></i>
</ul>
```

4. **Save** and **Reload** into your web browser. When you click on an item in your index, the browser should display the associated section at the top of your screen.

## Adding Links to a Named Anchor in Another Document

You can create a link that jumps to a section of another HTML document that is marked by a named anchor. The HTML for building a link to a named anchor in another local HTML document is:

```
<a href="file.html#NAME">Text to activate link</a>
```

and if the document exists on another web site:

```
<a href="http://www.cheese.org/pub/recipe.html#colby">Colby Cheese</a>
```

In lesson 8a we created a hypertext link that jumped from the section of our lesson on Mount St. Helens to **`msh.html`**, a separate HTML file. Now we will add a link in that second document that will return to the original section of the main Volcano page.

1. Open your **`msh.html`** file in the text editor
2. Near the bottom of the HTML (but above the **`</body>`** tag) enter the following text:

```
<hr>
<a href="index.html#usa">Return to <i>Volcano Web</i></a>
```

> **NOTE: We have included the *Italic* Style tag ... within the text marked by the named anchor "usa".**

3. **Save** and **Reload** into your web browser. When you click on one of the hypertext links at the bottom of the **`msh.html`** page, you should jump back to the "Volcanic Places in the USA" section of the

*Volcano Web* lesson.

In this case the link is just the name of another HTML file, **`msh.html`**, in the same directory as the **`index.html`** file. However, you can use a full URL (see lesson 7) to link to a named anchor in an HTML file on a remote computer. For example, to create a link to the "Introduction" section of an HTML file stored on the MCLI WWW server, the syntax would be:

```
<a href="http://www.mcli.dist.maricopa.edu/tut/final/index.html#intro">
   Introduction to Volcano Web</a>
```

The reference marker **`"#anchor_name"`** is tacked onto the end of the URL.

---

# Check Your Work

Compare your web page with a sample of how this document should appear. If your web page was different from the sample or the named anchor links do not properly connect, review the text you entered in the text editor.

Example of using the link to return to section on describing the named anchor...

# Review Topics

1. How do you identify a named anchor?
2. What are the steps for creating a link to a different section within a document?
3. How do you modify an anchor link to connect to a named anchor in another HTML document?
4. How do you create a table of contents for a web page?
5. What is the difference in function between the tags **`<a href="...>`** and **`<a name="...>`** ?

# Independent Practice

Create named anchors for the headings in your own web page and build an index that will link to these sub-sections.

---

# Coming Next....

In the next lesson you will learn how to make a picture act as a hypertext link.

**GO TO.... | Lesson Index | previous: "Links to the Internet" | next: "HyperGraphics" |**

---

**Writing HTML:** Lesson 8d: Links to Sections of a Page
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut8d.html

# 8e. HyperGraphics

Text does not have a monopoly on being "hyper"... expand the versatility of your web pages by having **pictures** act as hyperlinks (Just try clicking the cube!).

## Objectives

After this lesson you will be able to:

- Insert a graphic button in your web page that links to another HTML document.
- Insert a small graphic that acts as a "postage stamp" button for a link to display a larger copy of the image.

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

### A HyperLink Button

From the previous lessons, you have (hopefully) become comfortable with creating hypertext, text within your documents that connects a viewer to related information. You can also use inline images (see Lesson 7a) to act in a "hyper" manner. If you recall, in an earlier lesson we linked some text in our **`Volcano Web`** page, **`index.html`**, to a second page, **`msh.html`**, that described Mount St. Helens. Now in the latter page, we want to add a button that when clicked will link a reader back to the main lesson page.

The way to do this is to put the HTML tags for inline images *within* the hypertext portion of the anchor tag:

```
<a href="fileX.html"> <img src="graphic.gif">
Go to Document X</a>
```

In your web page, this HTML code will display an inline image and the text **`Go to Document X`**. Both will act as hyperlinks; clicking on either the text or the picture will tell your web browser to go to

the HTML file **fileX.html**. The image alone could be a hyperlink. In the World Wide Web, a "HyperGraphic" generally is surrounded by a colored box matching the color of hypertext on your web page, so you know that it is "active".

> **NOTE: Many browsers now can alter the color of hypertext and some pages have suppressed the display of the outline around HyperGraphic links. Generally, you can identify a hyperlink area on a web page by looking for a change in the cursor as it passes over a "hot" region. The cursor usually changes from an arrow to a hand when it passes over an active link.**
>
> **From a design standpoint, we recommend that if you use pictures to act as hyperlinks that you offer also a text link or use the ALT= attribute in the <IMG...> tag in case the viewer has turned off the loading of images.**

Now we will create a "hyper" graphic button. First, you need to get a copy of an arrow button from the Lesson 8e Image Studio.

You should now have a copy of the image file somewhere on your computer (You should move it to the **pictures** folder/directory in your workarea).

Next, add the HTML to make the button work:

1. Open the second HTML file, **msh.html** in your text editor.
2. At the bottom, modify the last line to:

```
<hr>
<a href="index.html#usa"> <img src="../pictures/left.gif">
Return to <i>Volcano Web</i></a>
```

> **Note: The inlineimage tag (<img...>) is completely within the anchor between the > that marks the end of the URL and the </a> that marks the end of the hypertext. Also note how the <i> tag is used within the active hypertext to emphasize the title of the lesson. And finally, we have used the <hr> tag to put a horizontal rule or a line above the button graphic (for more on this tag see lesson 4).**

3. **Save** the HTML file.
4. Return to your web browser, and select **Open Local** from the **File** menu to read in the **msh.html** file.
5. Select **Reload** from the **File** menu to update the changes.
6. Test the hyperlink to the Mount St. Helens page and the new button that should return you to the *Volcano Web* page.

# "Postage Stamp" Images

Previously, we advised against using large inline images in your web pages because viewers might have to wait a long time for images to download to their computer. One way around this is to create miniature-size copies of the graphic, or "postage stamps" which are displayed as inline graphics. Then, using the same steps as above, you can make the "postage stamp" image act as a hyperlink that links to the full size image. In this way, the large images are downloaded **only** if the viewer decides to see it.

First, you need to get a copy of the two image files from the Lesson 8e Image Studio. (These files should be stored in your **`pictures`** folder/directory of your workarea).

Next, create the postage stamp link in your main text file:

1. Open the **`index.html`** file in your text editor.
2. Under the heading **Long Valley** enter the following:

```
This field seismometer measures earthquakes associated
with subsurface volcanic forces and may help to predict
future events. It sits on a plateau known as the "Volcanic
Tableland" formed by a major eruption  600,000 years ago.
<p>
<a href="../pictures/seismo.jpg">
  <img src="../pictures/stamp.gif" ALT="link to large image"
  WIDTH="62" HEIGHT="85">
  -- [full size image, 55k] --
</a>
```

**The inline image, `stamp.gif` acts as a hyperlink to a larger image, `seismo.jpg`. When a user clicks on either the "postage stamp" or the text "-- [full size image, 55k] --", your web browser will display the larger image in a browser page.**

**Note the use of the dimensions of the `stamp.gif` image in the `<img...>` tag as well as the `ALT=...` attribute.**

**In our hypertext link we provide information that this image is 55k in size. By doing this, you provide the viewer the choice if they want to download an image of that size... If the link leads to something that is 1.6 Mb, as a viewer you might want to know that before you tried to view such a large file size.**

3. **Save** and **Reload** in your web browser.

---

# Check Your Work

Compare your web page with a sample of how this document should appear. If your web page was different from the sample, review the text you entered in the text editor. Some of the more common mistakes are discrepancies between spelling of the file names and the HTML code for the anchor links or not having the image files in the same directory as the HTML files. If you see an icon of a broken picture:

then it usually means the HTML does not match the file listed in the **<img>** tag or that the image is not in a GIF or JPEG format.

# Review

Review topics for this lesson:

1. How did you create the graphic button in your web page?
2. How are "postage stamp" links useful in including graphics in your web page?
3. How did you create your "postage stamp" link in your document?

# Independent Practice

Try to add buttons that link two web pages to each other. In a later lesson we will learn how to avoide the "box" around a hypergraphic.

---

# Coming Next....

Use the **`preformat`** tag to create a table of text in your Volcano lesson.

**GO TO.... | Lesson Index | previous: "Named Anchors" | next: "Preformatted Text" |**

---

**Writing HTML:** Lesson 8e: HyperGraphics
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut8e.html

# 9. Preformatted Text

How do you

```
display
    text
 where       space,            TABS   and   carriage return
           characters     count?
```

## Objectives

After this lesson you will be able to:

- Create a table of aligned text
- Control the placement of text on a page when tabs and spaces are important.

---

## Lesson

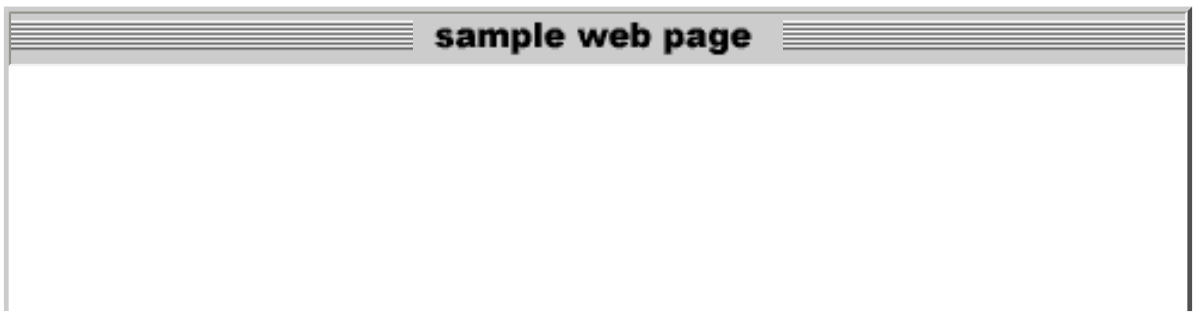
**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In previous lessons we have seen that a web browser will ignore extraneous space characters, tabs, and carriage returns in your HTML files. However, there are some instances where it will be important to maintain these aspects of page layout. In this lesson, the example will be where we want to insert a table of text with aligned columns.

The **`preformat`** tag instructs your web browser to display the text **exactly** as typed in the HTML document, including spaces, tabs, and carriage returns. A browser typically displays such text as a

m o n o s p a c e d

type, meaning a font in which every character has the same width. Here is example of what the **`preformat`** does:



```
<pre>
 We have indented with 5 space characters.
 And used the carriage return to jump
 to
 a
 new
 line.

       Here      we
       use
       spaces    to
       create    a
       text      table.
</pre>
```

Without the **<pre>** and **</pre>** tags, the same HTML produce:

**sample web page**

We have indented with 5 space characters. And used the carriage return to jump to a new line. Here we use spaces to create a text table.

With the **preformat** tag, it helps if the text editor you are using can display in a monospaced font (such as "Courier" or "Monaco"); if not, you will have to count spaces when aligning text into columns (and you will mutter bad words under your breath).

For our Volcano lesson, we want to add a table under the **Introduction** section that lists several well-known volcanoes, when they erupted, and the volume of erupted material. To do this:

1. Open the second HTML file, **index.html** in your text editor.
2. Under the last portion of the **Introduction** section, place a header of level 4 (<h4>) with the text **Volumes of Some Well-Known Volcanic Eruptions** (If you do not remember how to make headers, see lesson 3).
3. Below this heading, enter the following text **exactly** as follows (this would by an opportune time to cut and paste from this web page!):

```
<pre>
  Eruption                    Date          Volume in km^3
  --------                    ----          --------------
  Paricutin, Mexico           1943                1.3
  Mt. Vesuvius, Italy        79 A.D.               3
  Mount St. Helen, Washington 1980                 4
  Krakatoa, Indonesia         1883                18
  Long Valley, California    pre-historic       500 - 600
  Yellowstone, Wyoming       pre-historic          2400
</pre>
```

**In this example we use space characters to make the first column left justified and the other 2 columns center-justified. The dashes are used to highlight the column headings.**

4. **Save** and **Reload** in your web browser.

---

# Check Your Work

Compare your web page with a sample of how this document should appear. If your web page was different from the sample, review the text you entered in the text editor. If the columns are not aligned, then it is likely that you are missing or have too many space characters.

# Review

Review topics for this lesson:

1. How does the **`preformat`** tag change the way that a web browser interprets HTML?
2. In what other situations might you use this tag?

# More Information

You can still use HTML tags inside text that is marked by the **`preformat`** tag. For example we can add a hypertext link and some style tags that are **within** the **`<pre>...</pre>`** tags:

| HTML | Result |
|---|---|

```
<pre>
This tutorial is copyright
1995, 1996

    <b>Be sure to tell all of
    your friends about it!</b>
            <i>We
                 Welcome
                   Your
                     Feedback</i>
fondly,
<a href=
"http://www.mcli.dist.maricopa.edu/">
mcli</a>
</pre>
```

**sample web page**

```
This tutorial is copyright
1995, 1996

    Be sure to tell all of
    your friends about it!
                 We
                   Welcome
                     Your
                       Feedback
fondly,
mcli
```

**Note that the HTML tags do not count as spaces; they are ignored within the preformat region.**

Some web page developers will use the **<pre>...</pre>** tags with carriage returns in between to add white space between text or graphics in their web pages -- especially if they wish more white space than provided by the **<p>** tag. For example:

| HTML | Result |
|---|---|
| `Cheese was long since`<br>`abolished from the Orient.`<br>`<pre>`<br><br><br>`</pre>`<br>`...until Sir Longhorn arrived with the`<br>`great Cheese Crusade of 1167.` | **sample web page**<br>Cheese was long since abolished from the Orient.<br><br><br><br>...until Sir Longhorn arrived with the great Cheese Crusade of 1167. |

# Independent Practice

Add a table or chart to your HTML document using the **preformat** tag.

---

# Coming Next....

Use special character sets to add some âccèñt to your web pages.

**GO TO.... | Lesson Index | previous: "HyperGraphics" | next: "Special Characters" |**

---

**Writing HTML:** Lesson 9: Preformatted Text
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut9.html

# 10. Special Characters

How do you say...

**>>>Æ ñ Þóßÿ ?**

**NOTE: If the above characters do not display various accents or diacritical markers, then your web browser does not support the ISO character set. You would likely want to skip this lesson.**

## Objectives

After this lesson you will be able to:

- Use the HTML codes for ISO Latin 1 characters to display accent marks for non-English letters.
- Override the HTML use of < and > symbols when you need them in a document.
- Add extra spaces in between words and letters in a document.

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

### Accent Marks

Sometime you may need to use a special character in an HTML document, an accent or diacritical mark. The ones that are known as ISO These special characters are marked in HTML as:

```
&xxxx;
```

where XXXX is the code name for the special character. To create the special character for the German umlaut **(ü)**, we need to use the HTML:

```
&uuml;
```

For example, in the **Terminology** section of our Volcano lesson, we want to add an explanation of a technical term that was used to describe a particular type of volcanic flow. This term **`nuee ardente`** is from the

French term for "glowing cloud"; but to use the proper spelling we need an "acute" accent, so that the word appears as **nuée ardente**. In this case, we replace the first **e** in **nuee** with the HTML for the acute accented "e" **é**:

```
nu&eacute;e ardente
```

For reference on these codes, see the list of special ISO characters.

Now we will add a sentence to our HTML document that uses an accented letter:

1. Open the HTML file, **index.html** in your text editor.
2. Under the list of terms of the Volcanology Terminology section enter the text:

```
The term <I>nu&eacute;e ardente,</I> or
"glowing cloud" was first used by La Croix (1904)
in his description of the volcanic flows he observed in
the 1902 eruption of Mt Pel&eacute;e, a historically
active volcano on the island of Martinique.
```

   **NOTE: We have applied the acute accent mark for two "e" letters in this sentence. It may look strange! Be sure that you replace the letter with the sequence that displays the same letter with the accent mark.**

3. **Save** and **Reload** the HTML file.

## HTML Escape Sequences

The HTML for the accent mark is an example of the more general class of tags known as **escape sequences**. In entering HTML so far, you may have wondered what you do when you need to use a < (less-than) or a > (greater-than) sign? These two characters, plus the & (ampersand) have special meaning in HTML and cannot be used as typed. Instead, use the **escape sequences:**

```
&lt;  is used for <
&gt;  is used for >
&amp; is used for &
```

Now let's apply one of these symbols in our Volcano lesson. In the previous lesson, we added a table that lists several volcanoes and how much material was erupted from each. Let's say one of the values (500-600) for Long Valley is not very accurate (often such values are estimates), and we would like the entry to read >450 & <700. To do this:

1. Open the HTML file, **index.html** in your text editor.
2. Under the heading of **Volumes of Some Well-Known Volcanic Eruptions**, find the line for Long Valley in our table:

```
Long Valley, California     pre-historic        500 - 600
```

and change it to:

```
Long Valley, California     pre-historic       &gt;450 &amp; &lt;700
```

**NOTE: Although we are using the escape sequences within a preformatted text, note how a web browser properly interprets and displays the special characters. The escape sequences can thus be used in all portions of an HTML document including headings and anchor links.**

3. **Save** and **Reload** the HTML file.

## Extra Spaces

As you may have seen, a web browser will ignore all extraneous spaces in your HTML files. However, there may be times when you really want to have more than one space. When? Some writers like to have two spaces following the period at the end of the sentence. What if you wanted to indent the first sentence of every paragraph? How about having a single word with its individual letters spaced far apart?

An HTML code for adding a space character is the special character known as the "non-breaking space":

```
 
```

Here are some examples of how you might use the non-breaking space:

| HTML | Result |
|---|---|

**Two non-breaking spaces are used to spread the letters in a word farther apart**

```
<b><tt>
C   H   E   E  
S   E
</tt></b>
```

| HTML | Result |
|---|---|

**Two non-breaking spaces are used to indent the first sentence of each paragraph**

```
    When Sir Longhorn
had tragically died, no one was left to
carry on his tradition.
There was much sadness
in the land.
And no cheese.
<p>
    But then the young genius
Sheila Colby discovered the missing
ingredient. And once again, cheese
was plentiful.
```

**sample web page**

When Sir Longhorn had tragically died, no one was left to carry on his tradition. There was much sadness in the land. And no cheese.

But then the young genius Sheila Colby discovered the missing ingredient. And once again, cheese was plentiful.

| HTML | Result |
|---|---|

**One extra space is used to add an extra space after the end of each sentence.**

```
    When Sir Longhorn
had tragically died, no one was left to
carry on his tradition.  
There was much sadness in
the land.  
And no cheese.
<p>
    But then the young genius
Sheila Colby discovered the missing
ingredient.   And once again, cheese
was plentiful.
```

**sample web page**

When Sir Longhorn had tragically died, no one was left to carry on his tradition. There was much sadness in the land. And no cheese.

But then the young genius Sheila Colby discovered the missing ingredient. And once again, cheese was plentiful.

You may want to experiment with different ways to use the non-breaking space. At this time, we will not modify our HTML documents, but you may, if you wish, add the code for indenting each opening sentence of all paragraphs using two instances of the special code for the non-breaking space.

For more information on paragraph indentation, see Jim Barchuck's Stupid HTML Indent Tricks.

---

# Check Your Work

Compare your document with a sample of how this document should appear. If your document was different from the sample, review the text you entered in the text editor. Be sure that you have correctly bracketed the escape sequences with the & and ; characters.

# More Information

Here are some more special charcters that you may find useful:

| Name | HTML | Result |
|---|---|---|
| Copyright | © | © |
| Trademark | ® | ® |
| Cent | ¢ | ¢ |
| Degree sign | ° | ° |
| double-less than | « | « |
| micron | µ | µ |
| Midline dot | · | · |
| Negation, continuation line | ¬ | ¬ |
| Paragraph | ¶ | ¶ |
| Plus/Minus | ± | ± |
| British Pound | £ | £ |
| double greater than | » | » |
| Section | § | § |
| Yen | ¥ | ¥ |

See also the extensive list of special characters from WebMonkey.

# Review

Review topics for this lesson:

1. In HTML, what is the correct way to display a German umlaut **(ü)**?
2. What happens if you do not use an escape sequence for **<** and **>**?
3. Why would you need a special escape sequence for the & character?
4. How can you indent paragraphs?

# Independent Practice

In your own HTML document, add a foreign word that requires a special accent or a mathematical expression that uses the < or > symbol. Or, add some extra spaces to indent your paragraphs.

---

# Coming Next....

Build a bibliography using a **descriptive list.**

**GO TO.... | Lesson Index | previous: "Preformatted Text" | next: "Definition Lists" |**

---

**Writing HTML:** Lesson 10: Special Characters
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut10.html

# 11. Definition Lists

Yet another variety of

- lists
  - lists
    - lists...

## Objectives

After this lesson you will be able to:

- build a list of items with indented text block definitions
- create a bibliography with a definition list

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In lesson 6 we saw how to create two types of lists: ordered **`<ol>...</ol>`** and unordered **`<ul>...</ul>`** lists. We now introduce a third variety, the **definition list.** Unlike the lists we have seen earlier, the definition list marks its entries not with a bullet marker or a number, but by its pattern of indentation.

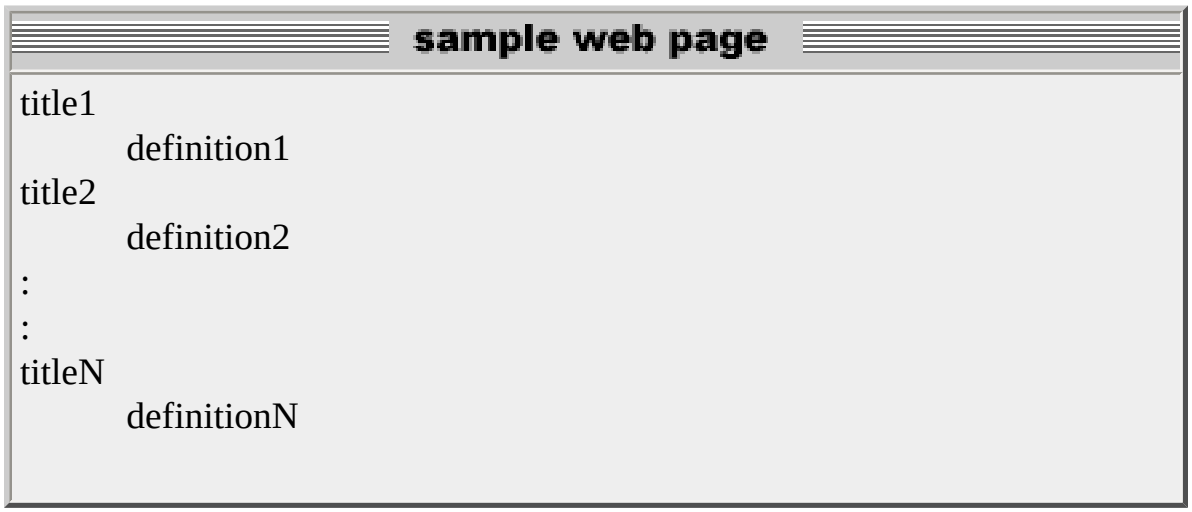
The format for a definition list tag is:

```
<dl>
<dt>  title1
<dd>  definition1
<dt>  title2
<dd>  definition2
                :
                :
```

```
                :
<dt>  titleN
<dd>  definitionN
</dl>
```

The **<dl> .... </dl>**; tags include alternating pairs of titles <**dt**> and definitions <**dd**>. A Web browser will typically generate the list with each definition indented to offset it from the title.

Viewed in a web browser, the above example looks like this:




The definition list might be used as a glossary , but for our example we will use it to create a short bibliography for our *Volcano Web* lesson:

1. Open the HTML file, **index.html** in your text editor.
2. After the unordered list under the heading **References** enter the following:

```
<h3>Bibliography</h3>
Check your library for these books:
<dl>
<dt>Cas, R.A.F. and Wright, J. V. (1987).
<dd><I>Volcanic Successions: Modern and Ancient.</I>
London: Allen &amp; Unwin.

<dt>La Croix, A. (1904)
<dd><I>La Montagna Pel&eacute;e et ses &Eacute;ruptions.</I>
Paris: Masson

<dt>Lipman, P.W. and Mullineaux (eds). (1981)
<dd><I>The 1980 Eruptions of Mount St. Helens, Washington.</I>
U.S. Geological Survey Professional Paper 1250.
```

```
</dl>
```

**NOTE: We have used some of the Special Characters for the ampersand symbol ("&") in the first reference and for the accent marks in the second reference. If you are unfamiliar with the HTML special characters, see lesson 10**

3. **Save** and **Reload** into your web browser.

# Check Your Work

Compare your document with a sample of how this document should appear. If your document was different from the sample, review the text you entered in the text editor. Do not forget the **<dl>...</dl>** tags that mark the whole list. One common mistake is switching the **<dt>** and **<dd>** tags.

# Review

Review topics for this lesson:

1. How does the definition list differ from the ordered and unordered lists?
2. In what instances might you use a definition list?
3. What is the difference between the **<dt>** and the **<dd>** tags?

# Independent Practice

Use a definition list to add a glossary or bibliography to your own HTML page.

## More Information

You can include other ordered/unordered lists within a definition list. For example, let's say we are making a list of the major mineral groups, with a description of their characteristics, and a sublist of minerals in each group and how they are used in society. We wish it to look like (just a few entries are shown):

**Oxides**

Combinations of metal ions with Oxygen, comprises the major ores extracted in mining operations

- Hematite (iron ore)
- Magnetite (iron ore, magnetic mineral)

- Corundum (gemstone, abrasive)

**Sulfates**

Metal ions combine with the Sulfate ion (SO4), atomic structure sometimes can allow bonding of water molecules

- Gypsum (plaster)
- Barite (drilling mud)

The HTML to produce this is:

```
<dl>
<dt><b>Oxides</b>
<dd>Combinations of metal ions with Oxygen, comprises the major ores
extracted in mining operations
    <ul>
    <li>Hematite (iron ore)
    <li>Magnetite (iron ore, magnetic mineral)
    <li>Corundum (gemstone, abrasive)
    </ul>
<dt><b>Sulfates</b>
<dd>Metal ions combines with the Sulfate ion (SO4), atomic structure
sometimes can allow bonding of water molecules
    <ul>
    <li>Gypsum (plaster)
    <li>Barite (drilling mud)
    </ul>
</dl>
```

---

# Coming Next....

Add an informative "signature" with a link for sending e-mail.

**GO TO.... | Lesson Index | previous: "Special Characters" | next: "Address Footers and E-Mail Links" |**

---

**Writing HTML:** Lesson 11: Definition Lists
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine

Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut11.html

# 12. Address Footers and E-Mail Links

Hey! You created a snazzy web page -- autograph it with a footer! Let people on the web send you an e-mail message right from your web page!

## Objectives

After this lesson you will be able to:

- insert a stylized footer at the bottom of a page
- create a hypertext link that will send an e-mail message

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

A worthwhile feature of web pages is a "footer" at the bottom of a page that provides information about the author and the document, maybe the last date it was modified, as well as a means to send the author a message by e-mail.

This is the place for the **`address`** tag which just stylizes a block of text in italic and offsets it to a new line.

It is a good idea to make footers brief, informative, and consistent between your different web pages. Some useful information to include is:

- Title or subject of the current page
- Date it was created/updated
- Copyright if appropriate (or even meaningful?)
- Name and e-mail for the web page author
- Name and hypertext link to affiliation/organization

As examples, see the footers at the bottom of every web page in this tutorial. In composing your own style, take a look at other web pages for ideas. ***Imitation still is a very high form of flattery!***

The HTML format for the **`address`** tag looks like:

```
<address>
  text text text text text text text text text
  text text text text text text text text text
</address>
```

Note that all other HTML inside the **`address`** tag is legal, so we might modify it with bold tags, line breaks, and a hypertext link tag:

| HTML | Result |
|---|---|

```
<address>
<b>Page Title</b><br>
Last Updated February 31, 1999<br>
Web Page by Alan Levine
(alan.levine@domail.maricopa.edu) <br>
<a href="http://www.mcli.dist.maricopa.edu/">
Maricopa Community Colleges</a><br>
</address>
```

**sample web page**

**Page Title**
Last Updated February 31, 1999
Web Page by Alan Levine (alan.levine@domail.maricopa.edu)
Maricopa Community Colleges

Now, suppose someone was reading your page and wanted to send you a comment on how nice your page looked. They would have to write down your e-mail address, launch another program, and send you a message. Wouldn't it be great if you could send email from your Web browser? Well, most web browsers now can!

The way to do this is to create a hypertext link with the **mailto** type in the URL (see lesson 8b for a refresher). Create an email hypertext link like this:

```
<a href="mailto:alan.levine@domail.maricopa.edu">send an e-mail to alan</a>
```

When the text **send an e-mail to alan** is clicked, the web browser will bring up a screen where you can compose a message and send it to me. What's more, you can also insert a default subject line for the e-mail message (NOTE: this may not work on all browsers):

```
<a href="mailto:alan.levine@domail.maricopa.edu?subject=hi from lesson 12">
   send an e-mail to alan</a>
```

Try it! Send me a note! send an e-mail to alan

And there is more you can do by adding on to the **mailto** link. If you wanted to send the same message to more than one address, say the President and Vice-President, you just put the email addresses separated by commas (note that in your HTML code this should be one long line, we have broken it up so it displays better here):

```
<a href="mailto:alan.levine@domail.maricopa.edu,pres@whitehouse.gov,
     vice-pres@whitehouse.gov?subject=hi from lesson 12">
     send an e-mail to alan, the pres, and the vice-pres</a>
```

Let's say the Vice-President should only be carbon copied ("cc:") on this message. To do this, just add another string after the subject using **cc=** and the email address. Note that the Subject= string and the cc= string are separated by a **&**:

```
<a href="mailto:alan.levine@domail.maricopa.edu,pres@whitehouse.gov,
     ?subject=hi from lesson 12&cc=vice-pres@whitehouse.gov">
     send an e-mail to alan, the pres; cc: the vice-pres</a>
```

And lastly, you can try insert a default message using the syntax **body=** and the text that should be placed in the body part of the email message window:

```
<a href="mailto:alan.levine@domail.maricopa.edu,pres@whitehouse.gov,
     ?subject=hi from lesson 12&cc=vice-pres@whitehouse.gov
     &body=Hi there, I think Alan deserves a dinner at your place.">
     send an e-mail to alan, the pres; cc: the vice-pres, with a default
message</a>
```

Please do not try sending these messages! Alan likes e-mail but don't bother the folks in the Whitehouse!

Now, let's return to our Volcano example. Note that you can have any text (or graphic) act as the hypertext link. So in the previous example, we would modify the HTML to have the internet address create the link for e-mail.

**HTML**

```
<address>
<b>Page Title</b><br>
Last Updated February 31, 1999<br>
Web Page by Alan Levine
<a href="mailto:alan.levine@domail.maricopa.edu">
(alan.levine@domail.maricopa.edu)</a> <br>
<a href="http://www.mcli.dist.maricopa.edu/">
Maricopa Community Colleges</a><br>
</address>
```

**Result**

**sample web page**

**Page Title**
Last Updated February 31, 1999
Web Page by Alan Levine
(alan.levine@domail.maricopa.edu)
Maricopa Community Colleges

And finally, we recommend that you also put in the footer the URL for the page. Why? What if someone prints out your web page but does not bookmark it or write down its URL? Including the URL on the page provides a handy reference. Just modify the above example (note how this HTML is **below** the **`<address>...</address>`** tag:

**HTML**

```
<address>
<b>Page Title</b><br>
Last Updated February 31, 1999<br>
Web Page by Alan Levine
<a href="mailto:alan.levine@domail.maricopa.edu">
(alan.levine@domail.maricopa.edu)</a> <br>
<a href="http://www.mcli.dist.maricopa.edu/">
Maricopa Community Colleges</a><br>
</address>
<p>
<tt>
URL: http://www.mcli.dist.maricopa.edu/tut/
</tt>
```

**Result**

**sample web page**

**Page Title**
Last Updated February 31, 1999
Web Page by Alan Levine
(alan.levine@domail.maricopa.edu)
Maricopa Community Colleges

URL:
http://www.mcli.dist.maricopa.edu/tut/

Now it is time to add a footer to your HTML file. For this example, we assume you are "Lorrie Lava" a staff Volcanologist at Big University (feel free to put your own information in place of what is below):

1. Open the HTML file, **`index.html`** in your text editor.
2. At the bottom of the document (but above the </body></html> tags), add the following:

```
<hr>
<address><b>Volcano Web</b> <br>
created by Lorrie Lava, <a
href="mailto:lava@pele.bigu.edu">lava@pele.bigu.edu</a><br>
Volcanic Studies, <a href="http://www.bigu.edu/">Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
<tt>URL: http://www.bigu.edu/web/index.html</tt>
```

**NOTE: We've used several HTML tags that have been covered in previous lessons. Also note the extra <p> tag at the bottom; this makes sure the last line of text is always visible.**

3. **Save** and **Reload** the HTML file.

# Check Your Work

Compare your document with a sample of how this document should appear. If your document was different from the sample, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. What does an **`address`** tag do?
2. Does an **`address`** tag have to be at the bottom?
3. How do you create a tag that will e-mail to you? to someone else? With a subject line?

# Independent Practice

Add an address footer and e-mail links to your own HTML documents.

---

# Coming Next....

Yet another way to break up those long boring sections of text! The BLOCKQUOTE...

**GO TO.... | Lesson Index | previous: "Definition Lists" | next: "Blockquotes" |**

---


©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

# 13. You Can Blockquote Me on That!

Yet another simple HTML tag for re-arranging your text:

> **"What is going on here?"** asked the `<blockquote>` tag to the `<p>` and `<br>` tags. They replied, "We are not sure, but you are very different from us!"

## Objectives

After this lesson you will be able to:

- Insert a block of text that is indented from the body text
- Apply style tags within blockquote text

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In traditional writing, quotations of three or more sentences are set off from the main text as an indented block of text. HTML also includes this capability via the **`<blockquote>....</blockquote>`** tag:

```
<blockquote>
"This is a long quotation from a very famous person. Since it is so long
and interesting, it should really be set off from the rest of the text.
This indicates clearly that the quote is from someone other than the writer."
</blockquote>
```

which yields:

> blah blah blah

blah blah blah

**contains more information about Pliny and his observations.**

**See also how the `<blockquote>` tag forces a paragraph break above and below the block of text.**

**And finally, these NOTES (like this one) we have used through the tutorial have made use of the `<blockquote>` tag!**

3. **Save** and **Reload** the HTML file.

# Check Your Work

Compare your web page with a sample of how this document should appear. If your document was different from the sample, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. What does the **`<blockquote>`** tag do?
2. Why don't you need a **`<p>`** tag before a blockquote?

# Independent Practice

Experiment with the **`<blockquote>`** tag in your own web page. Do not just think in terms of using it only for quotations. The tag can be effective for adding some variety to your web page layout. The **`<blockquote>`** tag is one way to avoid having many pages full worth of plain text paragraphs.

Some developers will use two or three (or more) nested **`<blockquote>`** tags to create an effect of wider margins. For example,

```
<BLOCKQUOTE>
  <BLOCKQUOTE>
    <BLOCKQUOTE>"Oh, for more tags," he lamented.
      <p>Replied she, "Remember, dear, that while HTML may
      be limited in terms of control
      over page layout, there is a great
      deal of potential in the creative use
      of the number of tags."
    </bLOCKQUOTE>
  </bLOCKQUOTE>
</bLOCKQUOTE>
```

produces:

**sample web page**

> "Oh, for more tags," he lamented.
>
> Replied she, "Remember, dear, that while HTML may be limited in terms of control over page layout, there is a great deal of potential in the creative use of the number of tags."

If had just used one set of **<BLOCKQUOTE> ... </bLOCKQUOTE>** tags, we would have seen:

**sample web page**

"Oh, for more tags," he lamented.

Replied she, "Remember, dear, that while HTML may be limited in terms of control over page layout, there is a great deal of potential in the creative use of the number of tags."

You cannot predict the exact amount of spacing this will provide on the sides of the pages, but it is an easy and effective variation for presenting text.

---

# Coming Next....

Divide a single web page into logical, connected "chunks"...

**GO TO.... | Lesson Index | previous: "Address Footers and E-Mail Links" | next: "Lumping v.s. Splitting" |**

---

**Writing HTML:** Lesson 13: Blockquotes
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut13.html

# 14. Lumping vs. Splitting

Wowza! You made a single web page! But, NOW, my HTML-literate friend, it is time to transform your ordinary long-scrolling "page" into a logically connected "web" of information.

## Objectives

After this lesson you will be able to:

- Transform a single web page to a series of linked pages
- Create a template for multiple web pages
- Build navigational features for connecting multiple web pages

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

Are you a **lumper** or a **splitter**? Neither? Both?

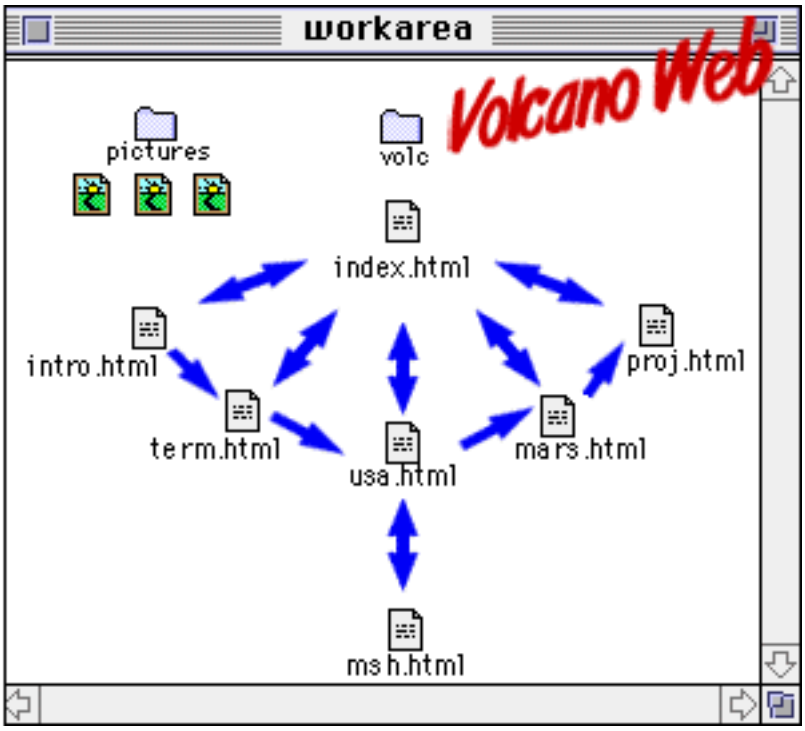
For organizing information, sometimes it's better to "lump" things together; other times it is better to "split" them apart. Scrolling through long web pages is often tedious. Long, single web pages take longer to load over networks when compare to a series of smaller pages.

In many cases, you can identify logical points to "split" information into multiple web-pages. However, there is no magical formula, and opinions will vary. You should strike a balance between pages with breaks that parallel the content yet avoid forcing the reader to click through too many screens of options and sub-options before getting to the desired information. It also becomes important to build in hypertext links that help the reader navigate your information web as well as providing visual clues about their location within the web.

So far we have built one web page with a link to a shorter page. In lesson 8d we created a list of links that works as a table of contents by connecting them to named anchors for the different sections of the *Volcano Web* lesson. These same divisions might be sensible breakpoints for splitting the single long page into sub-pages.

Up to now we have created a directory called **volc** that contains our two HTML files (**index.html**, the lesson, and **msh.html**, a second web page). We also have a second directory called **pictures** that contains our graphic image files.

We will now split the single Volcano Web file into a series of web pages, linked as shown in this schematic diagram. The entry point is a main index/cover page, **index.html** that has links that point to each of the other parts of our lesson:

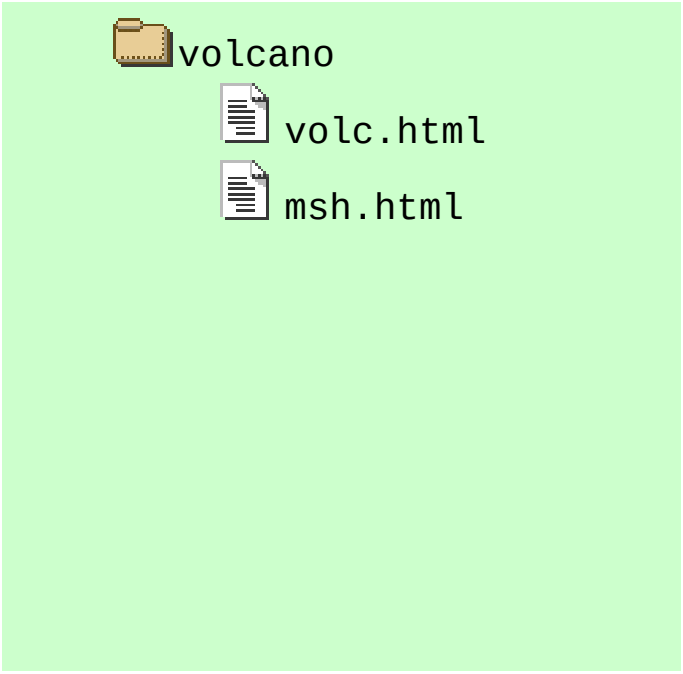


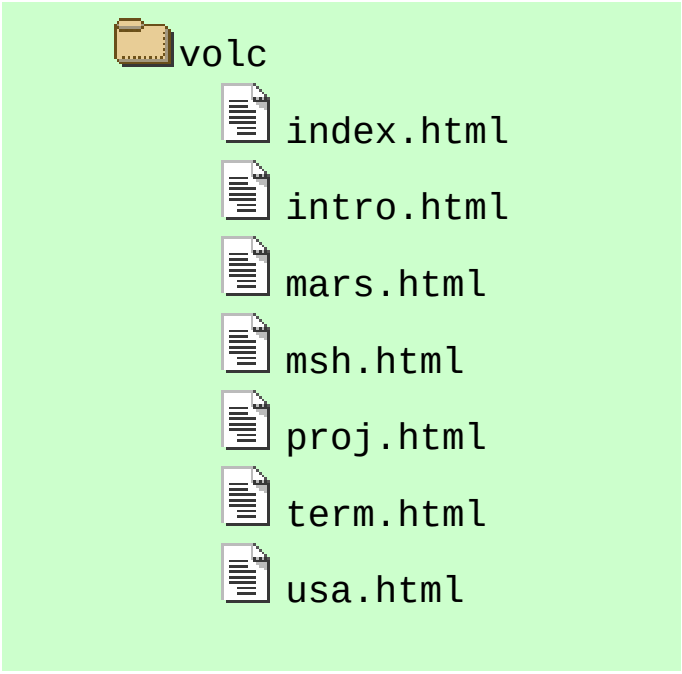

- **Introduction**
  **[intro.html]**
- **Volcano Terminology**
  **[term.html]**
- **Volcanic Places in the USA**
  **[usa.html]**
- **Volcanic Places on Mars**
  **[mars.html]**
- **Research Project**
  **[proj.html]**

Each part of the lesson will link back to the index as well as to the preceding and following pages. Also note the two-way link between **usa.html** and **msh.html**

Or if you prefer a more traditional directory listing structure:

| existing file structure | new file structure |
|---|---|
| work area<br>pictures<br>lava.gif<br>left.gif<br>msh.gif<br>seismo.jpg<br>stamp.gif | work area<br>pictures<br>lava.gif<br>left.gif<br>msh.gif<br>seismo.jpg<br>stamp.gif |




**NOTE:To complete this lesson, we will have to create quite a few new files and do a fair amount of copy/pasting from the files you have been working on. Be sure that you are comfortable jumping around between the different application and document windows on your computer.**

**Also, we have changed the name of the `volcano` directory/folder to a shorter `volc`. (We like trying to keep our URLs from getting too long, but also not shrinking them too short that they are cryptic)**

The first thing we will do is create the new **`index.html`** file, which will be the "cover" page for our *Volcano Web* lesson:

1. First make a copy of the **`index.html`** file you have been working on and name it **`old.html`** or something like that. This is a very good habit when you start making major changes to your web pages!
2. Now open the original version of the **`index.html`** file in your text editor. This is our lesson that we have been working on to this point.
3. We are going to use the picture, the opening text/quotation, and the table of contents as the content for our title page. To do this, we will remove the sections that will be "split" out to other web pages.

   Delete the sections from **Introduction**to **References**, that is everything between:

```
<hr>
<h2><A NAME="intro">Introduction</a></h2>
A <b>volcano</b> is a location where magma,
```

```
or hot melted rock from within a planet, reaches the surface.
 :
```

*and*

```
 :
<dt>Lipman, P.W. and Mullineaux (eds). (1981)
<dd><I>The 1980 Eruptions of Mount St. Helens, Washington.</I>
U.S. Geological Survey Professional Paper 1250.
</dl>
```

You may want to compare your HTML file to an example of how it should look at this point.

4. Now look at the section labeled "In this Lesson..." In our previous work, we used hypertext links to jump to a named anchor (e.g. **<a name="term">..</a>**) in the same document (see lesson 8a). Now we will modify these anchor links so that each jumps to another web page (which we will create below).

   Find the portion that reads:

```
<b>In this Lesson...</b>
  <ul><i>
  <li><a href="#intro">Introduction</a>
  <li><a href="#term">Volcano Terminology</a>
  <li><a href="#usa">Volcanic Places in the USA</a>
  <li><a href="#mars">Volcanic Places on Mars</a>
  <li><a href="#project">Research Project</a></i>
  </ul>
```

   and edit it to read:

```
<b>In this Lesson...</b>
  <ul><i>
  <li><a href="intro.html">Introduction</a>
  <li><a href="term.html">Volcano Terminology</a>
  <li><a href="usa.html">Volcanic Places in the USA</a>
  <li><a href="mars.html">Volcanic Places on Mars</a>
  <li><a href="proj.html">Research Project</a></i>
  </ul>
```

   Compare your HTML file to an example of how it should look at this point.

**NOTE: Be sure you understand the difference between a link written:**

```
<a href="#quest">go to questions</a>
```

**and another one written:**

```
<a href="quest.html">go to questions</a>
```

---

The next thing we will have to do is create the individual files for the other parts of our lesson. It will be easier if we first create a template file that we can modify for each of the different pages.

1. In your text editor, create a new file called **`temp.html`**
2. In this file, put the following HTML
   (If you wish, you can copy an example templatefile):

| HTML | Notes |
|---|---|
| `<html>`<br>`<head>`<br>`<title>XXXXXXXX</title>`<br>`</head>`<br>`<body>` | **HEAD:** In the head portion of each document, XXXXXXXX is the name of that section |
| `<h5>Volcano Web /`<br>`<a href="index.html">Index</a> /`<br>`<a href="xxxx.html">back</a> /`<br>`<a href="xxxx.html">next</a></h5>` | **NAVIGATION:** At the top of each page we use a small header (h=5) to create navigation links. **`Index`** points back to the main cover page. **`next`** and **`back`** link to the following and preceding pages. You will have to fill in the appropriate file name in for **`xxxx.html`**. Notice how this provides a common visual clue to each of our web pages. |
| `<h2>XXXXXXXX</h2>`<br>`:`<br>`:`<br>`:` | **HEADER:** Use a header=2 to put a title for that page. |

```
<hr>
<address>
<b><a href="index.html">
Writing HTML</a> :
XXXXXXXX </b><p>
created by Lorrie Lava,
<a href="mailto:lava@pele.bigu.edu">
lava@pele.bigu.edu</a> <br>
Volcanic Studies,
<a href="http://www.bigu.edu/">
Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
```

**ADDRESS FOOTER:** Note how the footer is now set up to indicate the name of the main web page (with a link back to it) as well as a line of text that indicates the name of the current section **XXXXXXXX**. Placing the name of the page here adds another important visual clue to the location of this page in the structure of the web we are creating.

```
<tt>URL:
http://www.bigu.edu/web/xxxxxxxx.html
</tt>
<p
</body>
</html>
```

**URL:** Be sure to modify the line that indicates the document's URL to reflect its file name **xxxxxxxx.html**

3. Now you should make 5 copies of the template file and make the appropriate changes to the template:

| File Name | Section | Notes |
|---|---|---|
| intro.html | **Introduction** | As this is the first section, remove the line from the navigation section: <a href="xxxx.html">back</a> |
| term.html | **Volcano Terminology** | |
| usa.html | **Volcanic Places in the USA** | |
| mars.html | **Volcanic Places on Mars** | |

| | | |
|---|---|---|
| `proj.html` | **Research Project** | As this is the last section, remove the line from the navigation section: **`<a href="xxxx.html">next</a>`** |

4. Now, open the old **`index.html`** file (that we re-named **`old.html`**) in your text editor. For each of the new files, you will have to copy the HTML that was underneath that section's **`<h2>...</h2>`** header and paste it into the new files you created in the previous step. Note that **Volcanic Places in the US** and **Research Projects** both include sub-sections that have **`<h3>...</h3>`** headers.
5. Finally, you will have to modify the link in **`msh.html`** file. Previously, it returned to a named anchor in the main lesson (the section for **Volcanic Places in the US**) where now it should link to the **`usa.html`** file. Open **`msh.html`** file in your text editor and edit the line to read:

```
<a href="usa.html">
<img src="../pictures/left.gif" alt="** ">
Return to
<i>Volcano Web</i></a>
```

Just to be consistent, you should also make the footer look like:

```
<hr>
<address>
<b><a href="index.html">
Volcano Web</a> : <a href="usa.html">
Volcanic Places in the USA</a> :
Mount St. Helens</b> <p>

created by Lorrie Lava,
<a href="mailto:lava@pele.bigu.edu">
lava@pele.bigu.edu</a><br>
Volcanic Studies,
<a href="http://www.bigu.edu/">
Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
<tt>URL: http://www.bigu.edu/web/msh.html</tt>

</body>
</html>
```

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. In this lesson we created quite a few files and it is very easy to make typographical errors.

# Review

Review topics for this lesson:

1. What are some advantages of short, multiple web pages over a single, long web page?
2. What would have happened if we did not modify the hypertext link in the **`msh.html`** file?
3. What were the navigational features we added to our lesson?

# More Information

Stylistically, your web pages are more readable if the hypertext links are integrated into the text of the content. This becomes more important as you create more web pages that have hypertext to link them together. Compare:

**"click here" links**

**sample web page**

In the spring of 1980, most people living in the vicinity of Mount St. Helens took heed of the scientists' warning about an impending volcanic eruption. (Click here to see a picture of Mount St. Helens) However, several were insistent on staying in their homes and sadly perished in the May 18 event. In that same year, measured increases in seismic recording devices caused scientists to warn of a possible event in Long Valley, California, and order a large evacuation of the Mammoth resort area. (Click here to see a seismometer) However, no such event occurred, and residents were angrily resentful for what they perceived as a false warning that caused great economic loss.

**integrated hypertext links**

**sample web page**

In the spring of 1980, most people living in the vicinity of Mount St. Helens took heed of the scientists' warning about an impending volcanic eruption. However, several were insistent on staying in their homes and sadly perished in the May 18 event. In that same year, measured increases in seismic recording devices caused scientists to warn of a possible event in Long Valley, California, and order a large evacuation of the Mammoth resort area. However, no such event occurred, and residents were angrily resentful for what they perceived as a false warning that caused great economic loss.

The **`"Click here..."`** hypertext not only disrupts the flow of the text, but the link text **`"here"`** is not related to the intended item. **As a suggestion, avoid writing any lines like `"click here to return to the home page"`.** Instead, write a clean link, e.g. **`<a href="home.html">Home Page</a>`** -- the clicking is inherent in the use of the web browser. Make the content readable and choose the link words to clearly indicate that the link leads to something related.

# Independent Practice

Take a look at the web page you are developing. Is it getting very long? Is there a logical division where you could "split" the page? Set up a cover/page index to your web pages and design appropriate links for navigating between them. Then design a template for for your "sub-pages."

Now ask some friends/colleagues to view your pages. Do they prefer the "split" pages or the "lumped" one? Could they easily negotiate their way through your information?

---

# Coming Next....

NOW we get to the fancier stuff with advanced HTML... Fasten your seat-belts!

**GO TO.... | Lesson Index | previous: "Blockquotes" | next: "Standard and Enhanced HTML" |**

---

**Writing HTML:** Lesson 14: Lumping v.s. Splitting
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut14.html

# 15. "Standard" and "Enhanced" HTML

Enter into the next realm of HTML
-- where you shall find more things you can do but more considerations as HTML becomes less "standard" (and less simple).

## Objectives

Since you have worked so hard up to this point, here is short lesson without any work to do! After this lesson you will be able to:

- Decide whether to use HTML that may not work for all web browsers
- View the source HTML of any web page

---

## Lesson

In lesson 0, we introduced you to the concept of "standard" HTML. The subsequent lessons in this tutorial will show you how to use some other HTML tags that may not be viewable on all web browsers. Here, we will just review some of the related issues before you venture ahead into the "danger zone."

How about a little history?

Long, long, ago ("in a galaxy far away?")... well around 1990, the World Wide Web was a text-based system based upon the HyperText MarkUp Language. The tags and interpretation were all built upon standards (HTML 1.0) set by an international committee. This was the key to the "web" becoming "world wide" because, by following the standards, the information was completely independent of the computer from which it was viewed.

Even when NCSA Mosaic burst upon the scene in 1993 as the first graphical web browser (created by whiz kid Marc Andreessen and others), the standards were followed to the letter, which at that time were updated to HTML 2.0.

The web started to get popular.

Extremely popular.

Insanely popular.

Other programmers began to build web browsers that offered the same functionality as Mosaic (because they supported all of the HTML features contained within the international standards). A group that included the original developer of Mosaic formed a new company -- its mascot was "Mozilla", ("Mosaic" + "Godzilla") with a brand new web browser known as Netscape Navigator.

Netscape was faster and more reliable than the NCSA Mosaic. It had a cool mascot! NetScape grew popular quickly perhaps because it contained functionalities that included all of HTML 2.0 PLUS more tags for things that you could not do in HTML 2.0. These "extensions" or "enhancements" have caused (and still cause) a great deal of arguments between HTML purists and those that like the "less than standard" features that Netscape added.

The Mozilla Netscape was **immensely** popular and quickly grabbed 3/4 of the web-browser pie. Now, in HTML, you could include colored backgrounds to your pages, formatted tables of text, text that wrapped around the side of images, and more. You began to see web pages that said, **"This page optimized for Netscape"**. Other browsers began to include support for the Netscape "HTML 2.0+" features. As the major online services opened up to the web, the browser market got even more crowded (and noisy).

The international commission was faced with a dilemma, as the market was largely demanding these "non-standard" tags to become part of HTML. As the rules for HTML 3.0 were being developed, they began to include most (but not all) of the tags Netscape had introduced. The standards process seemed to move to slowly for many people.

And the battle grew bigger into 1996 when Microsoft introduced their own special HTML tags. Would HTML become more Babel-like? For more information about the HTML battle, we refer you to Andy King's HTML 3.0 and Netscape. The most recent action has been the proposal of HTML 3.2, which encompasses most of the features supported by the big players in the browser arena now, Netscape and Microsoft.

The most recent position statement reflecting HTML 4.0 is the suggested standard that should be adopted by all web browsers. This latest evolution is meant to be a step that will provide a great deal of flexibility for future changes without them being mere "bolt-on" tags that have occurred to date. As we will see in later sections, Style Sheets provide a very efficient means for updating an entire web site with more consistent appearances, plus new features for easily internationalizing content and providing more accessibility features for the visually impaired.

**So what does this mean for you?** As you develop web pages, you should consider what your readers will be using to access your pages. Perhaps you are a teacher in one school or an information department

in a company that is sure all of their users will be using a particular browser. Then you can be comfortable designing and testing on only one browser. However, we consider this a short sighted approach that may down the road force you to do massive, tedious updates to your HTML coded pages.

More commonly, you will be "publishing" web pages from an Internet server and have no idea what browser is being used or even what kind of computer it is used on. You can add special warnings to your page. You can stick closer to the standards that are most widely supported on all web browsers. Even if you do use special tags, there are usually ways to have an alternative that will not cause havoc for users of other browsers.

**Most importantly... do not become fixated on how the page looks on just your own computer!** Your readers may have different browsers, different fonts, different text color preferences, different monitor sizes -- all of which may cause the display to vary in size, layout or appearance from how it looks on your computer. If you can try out your web pages on different computers, stretch and shrink the browser window, switch the standard fonts.

Fortunately, the original design for HTML has a very open and forgiving set of rules -- if a browser encounters a tag it does not know how to deal with or display, it simply ignores these tags.

For example, let's say my browser supports the **<drip>...</drip>** tag. This tag makes all text inside appears normal and then slowly "drip" toward the bottom of the page **(editorial note -- I MADE THIS UP!)**, a fancy effect for my home page:

```
<drip><H2 align=center>Welcome</H2></drip>
to my sloppy home page!. Look out
for the puddles!
```

which works like a charm on **my** home made browser because is has been programmed to understand how to display the **<drip>...</drip>** tags. On **your** browser that does not support this feature, you will see:

If your browser doesn't support this tag it just skips over it completely, rather than bombing or presenting an error message.

It's a brilliant concept, isn't it?

# Peeking at the Source

If you have not learned this already, the best secret for learning how to design web pages is to ... "creatively borrow". We are not advocating stealing HTML! But, if you find a web page design that you like, or find yourself asking, "How did they do that?" the easy thing to do is to look at their HTML code! It is sitting there waiting for you.

This is one of the best responses to the question, "Why should I learn all of this darn HTML gunk when I can just use a visual editor such as _____"? You cannot learn much from other websites if you depend upon a helper application to make your web pages. But... if you can "peek under the hood" of a web page and examine its HTML, you can understand and perhaps re-purpose interesting design techniques.

The exact menu names for doing this are different depending on which web browser and version you are using. Typically, your web browser menu will have an item called **View** from which you can select **Source** or **Page Source**. When you select this, it will download the HTML source code corresponding to the URL of the current page in view, and display this HTML code for you to see.

Another way to grab the HTML source of a page in view is to select **Save as...** from the **File** menu of your browser. This will bring up your familiar Save dialog box where you can select a location on your computer's drive and a file name. You should also see in the dialog box a pop-up menu labeled **Format** -- be sure to select the option labeled **Source**.

> **NOTE: This will save for you the HTML file for the page in view... but not any of the images used in the page. Some newer web browsers as well as third party tools provide the functionality to download everything in that web page as a single package.**

And a third way to get to the HTML source is to access the hidden menu-- right mouse click for Windows and Unix users, control-click for Macintosh users-- on any blank area of a web page.

As practice, see how quickly you can see and save the HTML source code for this lesson page.

As we go on into these more advanced lessons, the instructions will get a bit longer and more complicated. But you've gotten this far ok!

# Review

1. Why should you care about the standards of HTML?
2. In what settings can you be most comfortable using non-standard HTML?
3. What happens when a browser encounters an HTML tag it does not understand?
4. If you see an interesting design of someone's web page, how can you learn how it was constructed?

# More Information

- **WebReview** provides a detailed chart that shows you what tags are supported on different web browsers:
  http://www.webreview.com/1999/10_29/webauthors/10_29_99_3a.shtml
- **ZDNet's Tag Library** has a summary of HTML history and a tag by tag description
  http://www.zdnet.com/devhead/resources/tag_library/

## Coming Next....

Jazz up the page *behind* the text by **coloring** the page or adding a texture file.

**GO TO.... | Lesson Index | previous: "Lumping v.s. Splitting" | next: "Backgrounds" |**

---

**Writing HTML:** Lesson 15: "Standard" and "Enhanced" HTML
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut15.html

# 16. Colorful and Textured Backgrounds

Do not settle for that drab old grey page! Put a bold CoLor or textured pattern behind the text.

## Objectives

After this lesson you will be able to:

- Create a solid color background for a web page.
- Calculate the hexadecimal code for a color value.
- Change the color of text and hypertext link items.
- Create a textured background from a graphic file.

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

The background of your page should be just that -- in the background. As we add different colors or even patterns, keep in mind that it should not interfere with the readability of text. But don't take our words for gospel, draw your own conclusions from an example of what happens when you do not think about the impact of a noisy background.

For the pages of this tutorial, we have used a solid white color that makes for a clean and non-interfering (even if not dramatic) backdrop. No, it is not very exciting, but it is readable.

With some modifications to the **<body>** tag (introduced way back in lesson 1), you can add a solid color background to your web page. But before we show you how to do the fancy color stuff, we must first talk about RGB color values and their "hexadecimal" representation.

## "Hex-Dec" and Color Basics

In a web browser, you have at your disposal many system colors to color text and backgrounds. Each color is identified by its

**Red**- **Green**- **Blue** (RGB) values, three numbers that range from 0 to 255, each of which represents the intensity of the **Red**, **Green**, or **Blue** component of the desired color. Maximum values of all three (R=255, G=255, B=255) produce the color **white** and minimal values (R=0, G=0, B=0) produce **black**. All other colors are represented by different of RGB triplets.

Here is the tricky part. Rather than identifying a **color** as something like **"102,153,255"** each number is converted from base 10 (normal everyday numbers, digits from **0,1,2,3,4,5,6,7,8,9**) representation to hexadecimal, base 16 (digits from **0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**). Why? Hexadecimal is more easily and more efficiently understood by computers. So for the color example above, we would write in hexadecimal as **"6699FF"**. In this example, **"66"** is the **Red** value, **"99"** the **Green**, and **"FF"** the **Blue**.

Here are some hexadecimal examples of different colors:

| Color | Hex Code |
|---|---|
| | FFCCCC |
| | 33FF66 |
| | 663300 |
| | 000077 |
| | EEEEEE |
| | 444444 |

| Color | Hex Code |
|---|---|
| | 3300FF |
| | AA0000 |
| | 9900FF |
| | FFFF00 |
| | 888888 |
| | 000000 |

Now, don't panic about having do a bunch of numerical conversions! There are many tools that will let you click on a color and they will provide the hexadecimal representation. Many color tools are available from those folks at Yahoo.

But better yet, many browsers support standard shorthands for these 16 colors (those Windows VGA favorites):

| Color | Name |
|---|---|
| | aqua |
| | blue |
| | gray |

| Color | Name |
|---|---|
| | black |
| | fuchsia |
| | green |

| | | | |
|---|---|---|---|
| | lime | | maroon |
| | navy | | olive |
| | purple | | red |
| | silver | | teal |
| | white | | yellow |

# Solid Color Backgrounds

> **NOTE: You may want to first try a test to see if your browser supports solid color backgrounds.**

For our *Volcano Web*, the first thing we will do is add a color background to the **index.html** file. The HTML format for adding a solid color background involves modifying the **<body>** tag to read:

```
<body bgcolor=#XXXXXX>
```

where **XXXXXX** is the hexadecimal representation (indicated by the # sign in front of it) of the desired color.

If you recall, the image we use for the opening has pictures of volcanoes on a black background -- so if we were to use the same black color for the background of the web page, the picture would merge well into our page:

1. Open the **index.html** file in your text editor.
2. Find the **<body>** tag and change it to:

   ```
   <body bgcolor=#000000>
   ```

3. **Save** and **Load** your HTML file in your web browser

If you did things correctly, your browser should have changed the background to a solid black. **But you may have noticed that you cannot see your text!** Why? Well, the default color for text is also black, so you now have black text on a black background! Fortunately, we have some other options to add to the body tag to color the text and the hypertext items:

```
<BODY BGCOLOR=#XXXXXX TEXT=#XXXXXX LINK=#XXXXXX VLINK=#XXXXXX>
```

where the XXXXXX values will determine:

- **BGCOLOR** = the color of the background (default is **grey**)
- **TEXT** = the color of normal body text (default is **black**)
- **LINK** = the color of an item that is a hypertext link (default is **blue**)
- **VLINK** = the color of a hypertext item recently visited (default is **purple**)

You can now add some of these other color values by changing the tag to read:

```
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
```

**NOTE: the order of the items in the <BODY> tag does not matter**

You should now modify the <BODY> tags in all of your HTML files (fast and easy to do by copying and pasting the above example for the new **<body>** tag).

# Textured Backgrounds

**NOTE: You may want to first try a test to see if your browser supports textured backgrounds.**

Solid colors add some variety to web pages -- but you can go even farther by adding a textured background. You use a small image file (GIF or JPEG) and the browser will "tile" the web page with repeated copies of the image. Some of the things you should keep in mind are:

- **file size:** Adding a background texture will require that an additional graphic file be downloaded. We suggest that the image files be less than 10k in size.
- **readability:** Be selective! Many background texture files are more distracting than enhancing for readability. Try to use background textures that are very light (with dark text) or very dark (with light text). Select for high contrast with the text and its background.
- **effect:** In the first web browsers that used backgrounds, the page would not appear until the background file was downloaded. This might mean for a slow connection, your reader might have to wait long for the background image to arrive before even seeing any content! However, later browsers download the background **last** so the page first is grey, then the text and graphics appear, and lastly the background arrives. REMEMBER! The load time for your pages will likely be slower (considerably for older modems) when your pages are read from a web server.

In this part of the lesson, we will give you a chance to experiment with three different background images. The HTML format for adding a background image file is:

```
<body background="bgfile.gif">
```

where **bgfile.gif** is the name of the image file (this can be a full URL or a relative file path -- see lesson 8a).

Below we list the names of three background files. You can download each one (if you do not know how to download graphics from a web page, please refer to our help sheet). You should put each graphic file in your **pictures** folder/directory in your web workspace:

Blue Tile [bg.gif]

A square repeating pattern:
HTML: `<body background="../pictures/bg.gif">`
Example file with the Blue Tile background

Volcano Text [vtext.gif]

Light grey large text:
HTML: `<body background="../pictures/vtext.gif">`
Example file with the Volcano Text background

Legal Paper [paper.gif]

Long strip of notebook paper
HTML: `<body background="../pictures/paper.gif">`
Example file with the Legal Paper background

You can also modify the text colors for your page as we did in the above example. For example, if we want **RED** text for the Legal Paper background, we might write this HTML:

```
<body background="../pictures/paper.gif" text=#CC0000>
```

which gives us red text on yellow paper.

> **NOTE: Many web browsers have the ability to change the default text colors -- sometimes a user may have the preferences set for colors that will interfere with the ones you have selected. Therefore, we suggest when using any background tags (solid color or texture file) that you include the "normal" colors -- black for text, blue for hypertext links, and purple for recent links: `<BODY TEXT=#000000 VLINK=#660099 LINK=#0000FF>`**

If you are looking for some examples of background texture files, see the list of links from Yahoo

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. We are going to keep the sample files with the solid black colors that we added in the early part of this lesson.

# Review

Review topics for this lesson:

1. How do you add a solid color background to your web page?
2. Why are the color codes written in cryptic code like **#FF66CC** ?
3. How do you color the text of a web page?
4. What is the difference between

```
<body bgcolor=#FFFFFF>
```

and

```
<body background="tiles.gif">
```

?

# More Information

If you would like to know that the colors you choose for your web pages will look the same on other computers, consider the hexadecimal codes. With the different combinations of letters and numbers, there are literally millions of colors to choose from, e.g. **#FD6A2C**, **#06E293** or **#55A92B**. Yet, not all of these colors will be the same on all computers. Moreover, if your visitor does not have a cutting edge computer capable of displaying "millions" of colors, the web browser will make a closest "guess" to match the colors, with perhaps undesirable results.

Fortunately, you can do something about this... only use hex colors that are included in the NetScape 216 color palette. "What is that?" you ask. It is a set of 216 unique colors that are common to the system colors of both Macintosh and Windows operating systems. Therefore, these colors can be displayed on almost any computer.

If you are choosing these magical colors, you just need to choose ones that are triplet combinations of the following color codes: **00 33 66 99 CC FF**. For examples, these colors are all part of the cross-platform color set: **#FF6600**, **#00FF66** or **#669933**.

# Independent Practice

Add a solid color background or a texture file background to your web page(s). Ask some other people if they find that the text is suitably readable with the background elements you have chosen.

---

# Coming Next....

Two of the most hideous and **obnoxious** HTML tag evers created...

**GO TO.... | Lesson Index | previous: "Standard vs Enhanced HTML" | next: "Don't Blink, Don't Marquee" |**

---

**Writing HTML:** Lesson 16: Colorful and Textured Backgrounds
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut16.html

# 17. Don't Blink, Don't Marquee!

Are you trying to attract attention to your web pages? Rely on compelling **content** rather than **cheap** attention grabbers...

## Objectives

After this lesson you will (hopefully):

- Never use the **<blink>** tag
- Never use the **<marquee>** tag
- Understand why you should use HTML that is part of the HTML standard

---

## Lesson

When Netscape first unveiled their web browser they added a unique tag that would ostensibly draw attention to an important word or phrase -- by causing it to flash on and off in the web page. Rather than working through the international W3 consortium and developing an accepted standard for this features, they just added the functionality to their web browser.

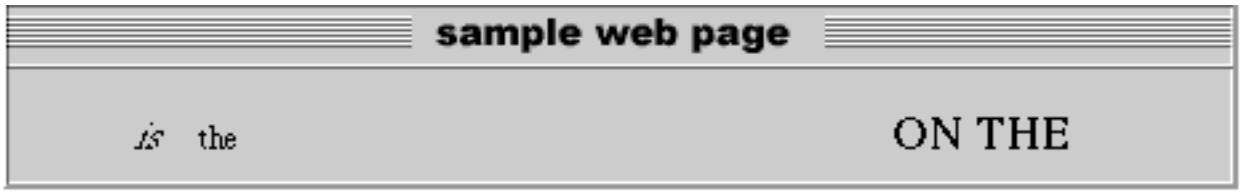
The dreaded blink tag:

```
<blink>Wow</blink>
```

is a sign that well-travelled web surfers take as **"avoid this page -- the person that wrote it just learned HTML from a bubble gum wrapper."**

Short of an extremely urgent reason to use it, avoid using this tag. Besides being obnoxious, it does not work on all web browsers! If you are viewing this page in NetScape, the colored text example below will "blink" visible. If you are viewing this page with Internet Explorer, you will not see any blinking text! (too bad, eh?).

If you are viewing this page with Internet Explorer, this animated GIF shows what NetScape viewers see (they get double blinked on this page, aren't they special?):




But not to be outdone, Microsoft created its own special, non-standard tag to work only in Internet Explorer

```
<marquee>Wow</marquee>
```

which takes the text inside and displays it like a ticker tape (one letter added at a time) across the page. So if are viewing this page in Internet Explorer, you would see this text march across the screen:

but if you are using a NetScape browser, nothing moves in the example above, so this animated GIF shows what Internet Explorer viewers see (they get double marqueed on this page, all is fair, eh?):

And who knows what you will see if you are using yet another web browser?

Unless you are building the **sleaze** row of web pages, avoid HTML tags like **<blink>...</blink>** and **<marquee>...</marquee>** that only work on specific browsers.

Stay with the standard HTML endorsed by the W3 Constortium.

Yes, this is our editorial stance. Go ahead and argue.

# Coming Next....

Let's get out of this drab, monotonous. black text and LIVEN it up a little! You know, take it to the $n^{th}$ degree!

**GO TO.... | Lesson Index | previous: "Backgrounds" | next: "Spiffing Up text" |**

---

**Writing HTML:** Lesson 17: Don't Blink
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut17.html

# 18. Spiffing Up Text

Not only can you color the background but you can Color, SIZE, and even change the font typeface for specific portions of text! How much would you pay?

**But WAIT!** You can now write superscripts and subscripts for fun things like chemistry and math:

$CO_2 + SO_4^{-2}$

$3x^2 - 2y^{-1/2} = 4z$

## Objectives

After this lesson you will be able to:

- Change the size of specific portions of text in a web page
- Change the color of specific portions of text in a web page
- Create superscripts and subscripts for text in a web page
- Specify the font for portions of text on a web page

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

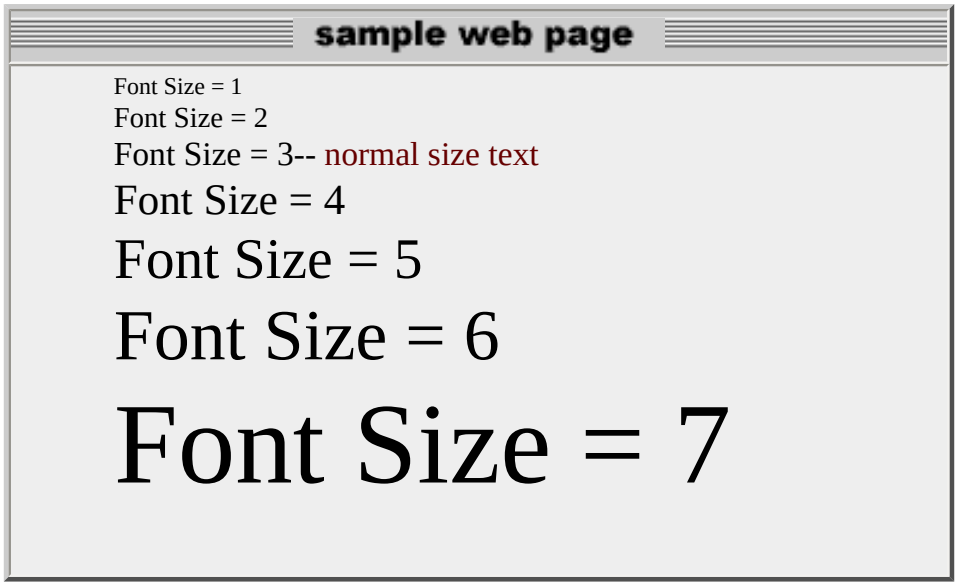
With the HTML introduced by Netscape and HTML 3.2, you have some more options for formatting your text. Specifically, you can create different sized text as well as different colored text. When used judiciously, the text formatting can enhance the layout of a web page. When used with reckless abandon, it can produce web pages that look like noise.

You can also now create superscript and subscripts used in mathematical expressions, chemical formulas, or as footnote markers.

In this lesson we will introduce you to these text features with some examples that you will use to modify your *Volcano Web* pages. You may want to first refer to the font test page to determine if your web browser supports the tags used in this lesson.

## Font Size

The **<font>..</font>** tag introduced by Netscape may be used to set the size of the font from 1 (smallest) to 7 (largest) with a size of 3 being the normal text size:




Remember that the actual size will depend on the computer font the user has selected for their web browser -- you are adjusting the size relative to the default font they have chosen.

The format for the **font** size tag is:

```
<font size=N>blah blah blah</font>
```

where N=1 to 7. The font tag may be used in conjunction with other style tags (lesson 5) or inside header tags (lesson 3).

The other method for using the **font** tag is to provide a **relative** size change:

```
<font size=+1>blah blah blah</font>
<font size=-2>blah blah blah</font>
```

i.e. where the **+N** or **-N** value provides the **offset** from the current font size. This is used with another tag:

```
<basefont size=5>
```

which changes the **base font size** from its default value of 3 to some other value. You might do this in a web page that will mostly have text of a larger or smaller than standard font size. That way, if you need to adjust small portions of the page, you can use the relative font size tags shown above.

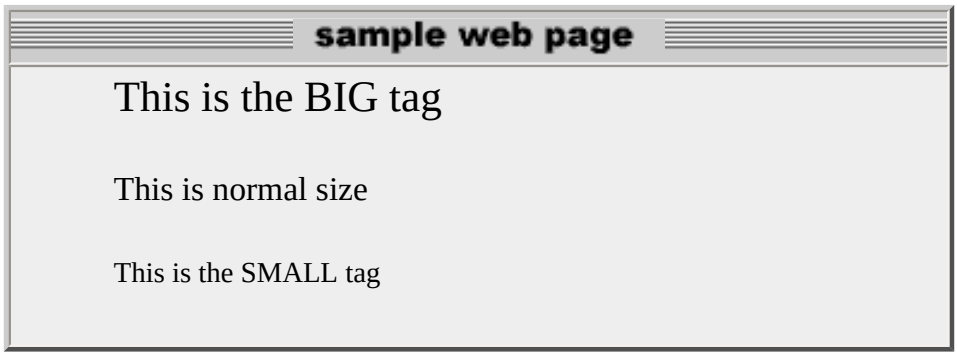
The power of using relative font size tags (e.g. **size=+2**) over absolute size tags (e.g. **size=5**) is that we could easily shift the size of ALL text in that page by changing one instance of the **<basefont>** tag.

> **Note: The <basefont> tag has no closing tag -- it continues to be the base font size until another <basefont> tag occurs.**

If a web browser does not support the <font> tags, you may want to try the HTML 3.0 tags:

```
<BIG>...</bIG>
<SMALL>...</SMALL>
```

which gives you less font sizes to work with but can still be useful for those browsers.




We will first use the **<font size=X>** tag to modify the title in our opening page:

1. Open the **index.html** file in your text editor
2. Previously we used an **<h1>..</h1>** header tag to format the title of our page. We will now use some font size tags instead to create a mixed size title.

   Change the line that reads:

   ```
   <h1>Volcano Web</h1>
   ```

   to:

   ```
   <p>
   <b><font size=+4>V</font><font size=+3>OLCANO
   WEB</font></b>
   ```

   Look carefully at what we have done -- the first V is now increased in size by 4 units above the base value, and the other letters (now capitalized) are increased 3 units above base value. This provides the layout style of SMALL CAPS. Also note that we have added a **...**tag to make the title stand out. And finally, because we are not using a header tag that carries a line break by default, we had to add a **<p>**tag above the title to force it to appear on a new line (we do not need one after because the next HTML is a <BLOCKQUOTE>that carries its own line break -- see lesson 13).
3. Next, we would like to make the quote from Pliny stand out a bit more, so we will raise it one font size:

   ```
   <b><I><font size=+1>"Nature raves savagely,
   threatening the lands"</font></I></b><br>
   ```

4. **Save** and **Load** into your web browser

Before going on, we will use the font size tags to modify two other web pages in our work area. the page for **Volcanic Places in the USA** (file **`usa.html`**) and **Research Project** (file **`proj.html`**) both use **`<h2>...</h2>`** tags for the main title and **`<h3>...</h3>`** tags for sub section headings. Go into your text editor and change every occurrent of tags:

```
<h3>blah blah blah</h3>
```

with:

```
<p>
<font size=+1><b>blah blah blah</b></font><br>
```

Note that the **`<p>`** tag forces a one line break with preceding text above and the <br> tag forces a line break (no white space) to subsequent text.

Can you see the difference between using the **`<font>`** tags and the **`<hN>`** tags for section titles? The formatting differences may be subtle, but it does offer you, the HTML author, alternatives for your web page design. Just remember that if a reader's browser does not support font size tags, they will see **all** the text as the standard size.

## Two Methods for Creating Headings

| **`<h3>...</h3>`** | **`<p><b><font size=+1> ...</font></b><br>`** |
|---|---|
| **sample web page**<br>...before the next great innovation that rocked the world of rotten milk.<br>**History of Longhorn Cheese**<br>Longhorn cheese was first discovered by Alister Longhorn in 1754 when he... | **sample web page**<br>...before the next great innovation that rocked the world of rotten milk.<br>**History of Longhorn Cheese**<br>Longhorn cheese was first discovered by Alister Longhorn in 1754 when he... |

# Font Color

In lesson 16, we introduced the tags for coloring text of the entire web page as well as the proper format for representing color values in HTML. You can also add an attribute to the tag to color a selected portion of text using the hexideciaml codes or the 16 colors recognized by their names:

```
<font color=red>...</font>
<font color=#993459>...</font>
<font color=lime>...</font>
```

```
<font color=#002200>...</font>
  <font color=navy>...</font>
<font color=#193467>...</font>
```

We will not go crazy with the color tags, but as an example, we will change the color of the **VOLCANO WEB** text we worked on above for the cover page. If you recall, we set the text color for the text of this page to yellow and we will override that color just for the title text:

1. Open the **index.html** file in your text editor.
2. Edit the line that contains the text of VOLCANO WEB to read:

```
<b><font size=+4 color=#FF66FF>V</font>
<font size=+3 color=#DD0055>OLCANO WEB</font></b>
```

3. **Save** and **Reload** in your web browser

**NOTE: The size and color attributes can reside together in a <font> tag. The effect here makes the "V" a brighter violet color and one size larger than the other letters.**

# Superscripts and Subscripts

Until HTML 3.0 you were out of luck if you needed to write mathematical expressions, chemical formulas, or other expressions with superscripts and/or subscripts. These new tags raise/lower the "scripted" text one half line and sizes it one size smaller.

The HTML format for these tags is:

**Superscripts / Subscripts**

| HTML | Result |
|---|---|
| `<sup>...</sup>` | super$^{script}$ |
| `<sub>...</sub>` | sub$_{script}$ |

We will now use these tags for our **Introduction** page:

1. Open the **intro.html** file in your text editor.
2. First we will use subscripts to write some chemical formulas. After the last sentence in paragraph 2 ("Compare the history of human beings..."), add this sentence that uses subscripts:

```
Volcanoes were important contributors to
the early earth atmosphere by releasing
```

```
gases such as nitrogen (N<sub>2</sub>),
carbon dioxide (CO<sub>2</sub>), and
ammonia (NH<sub>4</sub>).
```

3. Now we will use superscripts to denote a cubic volume. Below the table we created with the **<pre>...</pre>** tags, add this sentence:

```
Note that volcanic eruptions that occurred
before historic times were several orders
of magnitude larger (more than 1000 km<sup>3</sup>
in erupted volume) than ones observed by
humans.
```

4. If you notice the third column of our table, when we first wrote it we had to use **"km^3"** to indicate "km$^3$". Although this text is **inside** the preformat tag, we can still use the superscript tag. To do this change:

```
Volume in km^3
```

to

```
Volume in km<sup>3</sup>
```

# Font Face

The HTML 3.2 standards included the **<font>** tag to specify a particular screen font for text display. This may not work in all web browsers, so you may first want to take a try the font test.

The HTML for specifying a font face is:

```
<font FACE="font1,font2">some text
```

If a viewer's web browser supports the **font FACE** attribute and they have one of the listed fonts installed on their computer, then the text will be displayed with the specified font. Otherwise, it will use the same font as the rest of the web page.

If you choose to use a font face, you should select a face that is standard or be sure that the intended computers have any "exotic" fonts installed.

We will now modify the **<font>** tag for our title page so that the words "Volcano Web" appear in a different font:

1. Open the **index.html** file in your text editor.
2. Edit the line that contains the text of VOLCANO WEB to read:

```
<b><font face="Arial,Helvetica" size=+4 color=#FF66FF>V</font>
<font face="Arial,Helvetica" size=+3 color=#DD0055>OLCANO WEB</font></b>
```

3. **Save** and **Reload** in your web browser

**NOTE: We've included specifiers for the browser to choose Arial for Windows Helvetica for Macintosh and/or computers that do not have the Arial font installed.**

**Use this HTML with discretion! There is an art of use versus abuse of too many font styles!**

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. How do you change the size and color of a portion of text in a web page?
2. What HTML can you use if your browser does not support the **<font>...</font>** tags?
3. How do you create subscripts in HTML?
4. How can you create a block of text to display with a specific font?

# More Information

Here are two more styles that are now available in HTML 3.2 that may work in your web browser.

| Style tags | |
|---|---|
| **HTML** | **Result** |
| `<u>This is Underline...</u>` | This is Underline |
| `<strike>This is Strike-through...</strike>` | ~~This is Strike-through~~ |

And finally, here is a subtle point about text coloring that you may find useful someday. In lesson 16, we learned how to use the **<body>** tag to color the background, text, and link colors. If we were to color a block of text with the **<font>** tag, it would only affect the body text, and not the color of the hypertext links -- these keep the colors that are either the default blue or whatever is designated in the **<body>** tag.

## Default Link Colors HTML Result

```
<font color=red>
It was a long time after the sad death of
```

```
<a href="http://www.longhorn.org/sir/">
Sir Longhorn</a>
that someone was able to recreate
his formula.
```

It was a long time after the sad death of Sir Longhorn that someone was able to recreate his formula. We can change the color of the hypertext link by placing the tags *inside* the anchor link. Note that this will work only for unvisited links; once you have seen the page that corresponds to the link, it will be colored by whatever color designates visited links, e.g. the default purple Modified Link Colors

```
<font color=red>
It was a long time after the sad death of
<a href="http://www.longhorn.org/sir/">
<font color=#228800>Sir Longhorn</font></a>
that someone was able to recreate
his formula.
```

It was a long time after the sad death of Sir Longhorn that someone was able to recreate his formula.

# Independent Practice

Try experimenting with the **<font>...</font>** tags in your own web pages. See how they can work inside of the **<hN>...</hN>** tags too. Experiment with using different fonts, even the wild ones!

Look for places where you think you might need superscripts or subscripts. One example might be for footnotes -- you could number or character code them, and then each footnote could act as a hypertext link to a footnote (or jump to a separate page for end notes):

**sample web page**

... and after Linberger and Gordon's 1963 study[12] on the effects of temperature on cheese maturation, Gange and Walters (1964)[13] as well as Colby (1969)[14] reached the same conclusion.

blah blah blah

# Coming Next....

Huge Rulers? Hefty Rulers? Handy Rulers? More options for **<hr>**

,

**GO TO.... | Lesson Index | previous: "Don't Blink, Don't Marquee" | next: "Easy Horizontal Rules" |**

---

**Writing HTML:** Lesson 18: Spiffing Up Text
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut18.html

# 19. Easy Horizontal Rules

Confused? This is just one more option for creating fancy

dividers to mark the different sections of a web page.

## Objectives

After this lesson you will be able to:

- Create horizontal rule lines with different thicknesses
- Create horizontal rule lines with different widths
- Create horizontal rule lines without 3D shading

---

## Lesson

**Note: We will not be modifying our web pages in this lesson -- so you can just review the information and then decide if you want to experiment with it. You may want to first look at the test page to see if your browser supports the tags used in this lesson.**

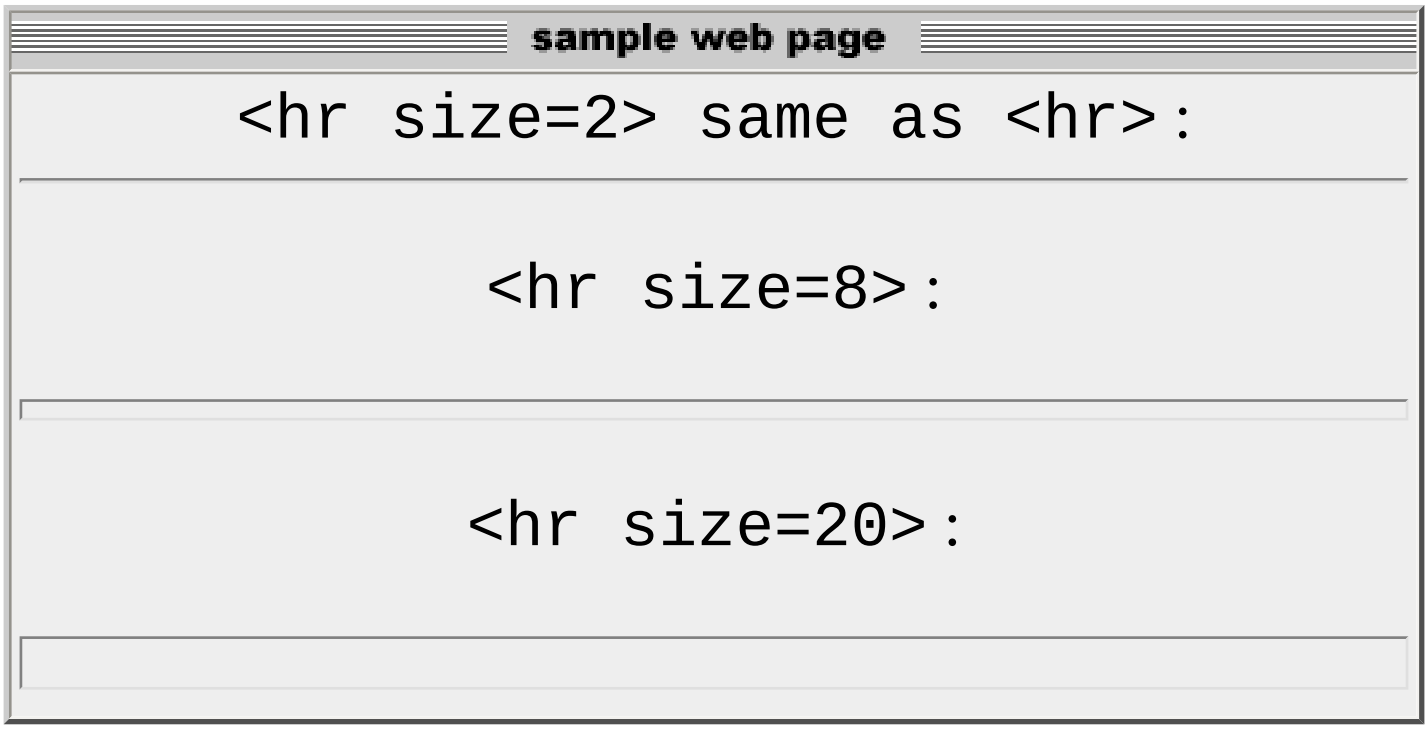
Netscape's first HTML "extensions" (or "deviations form standards") offered some more formatting options for the **<hr>** tag (**H**orizontal **R**ule) -- see lesson 4). By default, the Netscape browser displayed the solid line separator with a three-dimensional, shaded look rather then the solid line as implemented in previous web browsers.

### Line Thickness

The first option is to allow for lines of different thicknesses by using the option:

```
<hr size=N>
```

where **N** is the thickness of the line in pixels. Let's look at some examples to show the effect.



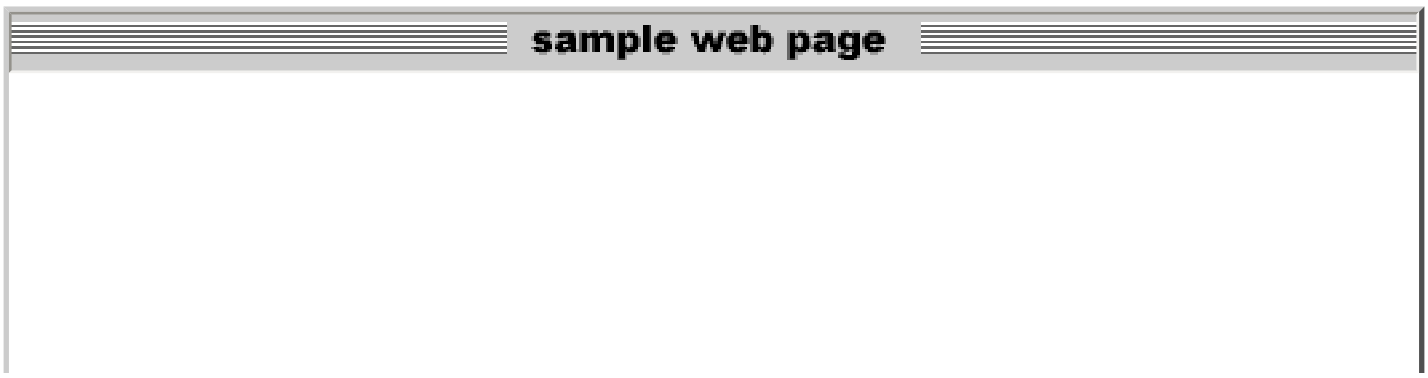

## Line Width

Another formatting option for the **<hr>** tag allows you to adjust the width of the line -- it does not necessarily have to stretch across the whole page. You can do this by using this format:

```
<hr width=X>
<hr width=Z%>
```

where **X** is a number of pixels for the width and **Z** is the percentage of the current web page. In general we recommend using the percentage as it will adjust itself to the width of the browser window used by the person reading your pages (some out there are using 2-page high resolution monitors while others are squinting into their 13" screen).

Here are some examples (see how we include the size tag as well):



<hr width=80 size=10>:

<hr width=80% size=10>:

**NOTE: Try stretching and/or shrinking the width of your web browser window to see the difference between specifying the width in absolute pixels `(width=80)` compared to specifying the width in percentage of the web page `(width=80%)`**

**sample web page**

<hr width=40% size=10>:

<hr width=5% size=10>:

## No Shading

Finally, there may be a case where you do not want the 3D shading on the **`<hr>`** tag. It is as simple as:

**sample web page**

<hr width=60% size=6 noshade>:

<hr width=60% size=1 noshade>:

# Review

Review topics for this lesson:

1. How do you change the thickness of a horizontal rule?
2. How can you change the width of a horizontal rule?
3. What is the effect of the **noshade** attribute inside the **<hr>** tag?

# Independent Practice

Experiment with some of the **<hr>** options in your own web pages.

---

# Coming Next....

Move it to the left... no! to the right... no! to the center...

**GO TO.... | Lesson Index | previous: "Spiffing up text" | next: "Extra Alignment" |**

---

**Writing HTML:** Lesson 19: Easy Horizontal Rules
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut19.html

# 20. Extra Alignment

Why just stay over to the **left**?

... when you can move to the **right**?

or just hang out in the middle?

## Objectives

After this lesson you will be able to:

- Create text that is aligned to center of the page
- Align images and blocks of text
- Add specified spacing around images
- Create text that is right justified

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now. Also, you may want to first look at the test page to see if your browser supports the tags used in this lesson.**

### Text Alignment

As the web grew and grew and grew, there was demand for more control over page layout. One desire was to format text that was aligned to the center of a page, rather then just justified to the left margin.

This gets into one of those sticky Netscape versus "standard" HTML situations. Netscape introduced an HTML extensions -- ** ... ** tag -- it aligns everything inside the tags with the center of the web page. Seems to make sense? Well, if you talk to a mark-up language purist, alignment is an **attribute**. The more general HTML (and "standard") for center alignment is a variation on the ** ** tag:

```
<p align=center>
blah blah blah
<img src="file.gif"><br>
blah blah
</p>
```

Note that now there is a closing ** ** tag. This becomes important later when we get into Style Sheets and other

design features of HTML 4.0. Also, if you are center aligning several paragraphs, each one will have to be marked with a separate **`<p align=center>...</p>`**

Most other browsers and the HTML 3.2 standards now support Netscape's **`<center>...</center>`** tags. Remember that if a browser does not support a tag, it will just ignore it.

For the effect of center alignment, compare:

sample web page

## Cheese In History

**Cheese has been there for many major events**

When Magellan sailed through the narrow Straits, he was eating cheese. When the Allied troops landed in Europe, cheese was there. When Neil Armstrong stepped onto the moon, he had just had finished a cheese snack.

return to the cheese home page

with this example that center aligns the content:

sample web page

## Cheese In History

**Cheese has been there for many major events**

When Magellan sailed through the narrow Straits, he was eating cheese. When the Allied troops landed in Europe, cheese was there. When Neil Armstrong stepped onto the moon, he had just had finished a cheese snack.

return to the cheese home page

Now we will modify our cover page (**`index.html`**) to use the **`<center>...</center>`** tag on the list of lesson sections. We will also make some other changes to improve its appearance.

1. Open up the **`index.html`** file in your text editor.
2. Find the section that looks like:

```
<b>In this Lesson...</b>
<ul>
<i>
<li><a href="intro.html">Introduction</a>
<li><a href="term.html">Volcano Terminology</a>
<li><a href="usa.html">Volcanic Places in the USA</a>
<li><a href="mars.html">Volcanic Places on Mars</a>
<li><a href="proj.html">Research Project</a>
</i>
</ul>
```

and replace it with the following HTML:

```
<p align=center>
<font size=+1>
<i>
<a href="intro.html">Introduction</a><br>
<a href="term.html">Volcano Terminology</a><br>
<a href="usa.html">Volcanic Places in the USA</a><br>
<a href="mars.html">Volcanic Places on Mars</a><br>
<a href="proj.html">Research Project</a><br>
</i>
</font>
</p>
```

3. **Save** and **Load** into your web browser.

   **NOTE: Look closely at the changes we have made. This whole section is enclosed in the `<p align=center>..</p>` tags for alignment. The unordered list we built in lesson 6 would look odd centered -- the bullet marks would be staggered. So we have removed the `<ul><li>...<li>...</ul>` structure. The `<br>` tags at the end of each line will force a line break.**

   **And finally, we have added a `<font>...</font>` tag to increase the text size. (Note, this may not work on all web browsers; reports are that the font will not be resized in Internet Explorer version 4).**

   **If you wish you could also use the `<center>...</center>` tags in place of `<p align=center>...</p>`**

You may want to compare your HTML file to an example of how it should look at this point.

Any header **`<hN>...</hN>`** tag may be center aligned by adding the attribute:

```
<hN align=center>blah blah blah</hN>
```

We will now use this attribute to center the title of each web page:

1. Open all of your other HTML files in your text editor.
2. For each one, edit the text that appears near the top, in **<h1>** or **<h2>** tags, following this example from file **intro.html**:

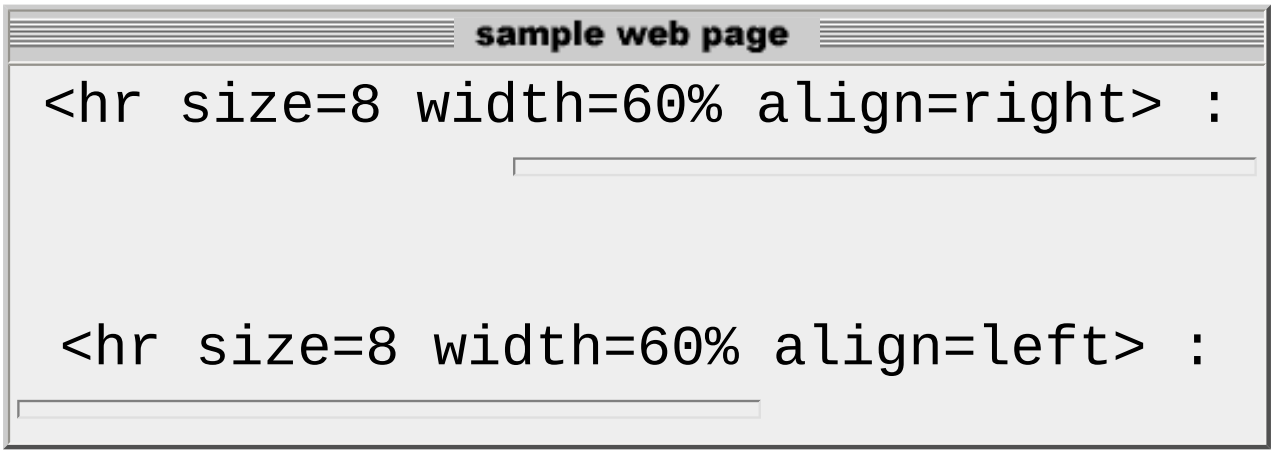
   ```
   <h2>Introduction</h2>
   ```

   to

   ```
   <h2 align=center>Introduction</h2>
   ```

3. **Save** and **Reload** into your web browser.

   **NOTE: There is also an align attribute for the <hr> tag that is relevant to the other options we reviewed in lesson 19 "Easy Horizontal Rules". When you specify a shorter width for a horizontal rule, you can also specify that it be aligned to the left or the right (the default is to center align horizontal rule lines):**




# Aligning Images and Text

In lesson 7a we learned how to put images in our pages and saw that we could have one line of text align with the top, middle, or bottom of an image. However, up to now, we could not have a block of text sitting side by side with an inline image.

With the **align** attribute in the **<img>** tag we can specify to have the image itself aligned to the right or left margin of the page. But more so, we can have any HTML following it "fill" in around the empty space. The HTML to do this is:

```
<img src="filename.gif" align=right>
<img src="filename.gif" align=left>
```

Compare these two examples:

## no align attribute

**`align=bottom` (default if none specified)**

sample web page

### Cheese In History

**Cheese has been there for many major events**
When Magellan sailed through the narrow Straits, he was eating cheese. When the Allied troops landed in Europe, cheese was there. When Neil Armstrong stepped onto the moon, he had just had finished a cheese snack.

## text flow

**`align=left`**

sample web page

### Cheese In History

**Cheese has been there for many major events**
When Magellan sailed through the narrow Straits, he was eating cheese. When the Allied troops landed in Europe, cheese was there. When Neil Armstrong stepped onto the moon, he had just had finished a cheese snack.

Shrink and stretch your web browser window to see how it affects your page with aligned images/text.

One more note. There will be times when the text or other items that are aligned with the image are rather short and you want to force the next text to jump down below the image. There is an attribute for the **`<br>`** tag to clear the alignment:

```
<br clear=left>
<br clear=right>
<br clear=all>
```

that will "clear" (or disable) any preceding alignments set up in the **`<img>`** tags. We suggest that you always use these tags because the alignment will vary depending on the font specified in the reader's web browser and the width of their browser window.

# Image Space Padding

Sometimes when you use the right or left alignment with an image, the text wraps a bit too close to the image. There is an option you can add to the **<img>** tags to add more padding or space around your image:

```
<img scr="my_image.gif" alt="My Image" width=79 height=142 vspace=8 hspace=12>
```

The **vspace** and **hspace** options put the amount of pixels specied of space above and below (vspace) or on the left and the right (hspace) of the image. So we could modify our example above to look like:

**text flow with no spacing**
**align=left**

sample web page

**Cheese In History**

**Cheese has been there for many major events**
When Magellan sailed through the narrow Straits, he was eating cheese. When the Allied troops landed in Europe, cheese was there. When Neil Armstrong stepped onto the moon, he had just had finished a cheese snack.

**text flow with spacing**
**align=left, vspace=10 hspace=18**

sample web page

**Cheese In History**

**Cheese has been there for many major events**
When Magellan sailed through the narrow Straits, he was eating cheese. When the Allied troops landed in Europe, cheese was there. When Neil Armstrong stepped onto the moon, he had just had finished a cheese snack.

# Adding Image Alignment to our Volcano Site

We will now return to our **index.html** page to re-format the opening screen. The picture may look nice, but it takes more vertical space to have the picture followed by the text below. We will use the image/text alignment to put some of the text adjacent to the image.

1. Open the **index.html** file in your text editor.

2. Find the portion that reads:

```
<img alt="A Lesson on:"   src="../pictures/lava.gif">
<p>
<b><font size=+4 color=#FF66FF>V</font>
<font size=+3 color=#DD0055>OLCANO WEB</font></b>
<BLOCKQUOTE>
<b><I><font size=+1>"Nature raves savagely,
threatening the lands"</font></I></b><br>
-- <a href="http://magic.geol.ucsb.edu/~fisher/pliny.htm">
Pliny the Elder</a>, who died of asphyxiation after
observing the destruction of Pompeii by the
79 A.D. eruption of Mount Vesuvius.
</bLOCKQUOTE>

In this lesson you will use the Internet
to research information on volcanoes and then
write a report on your
results.
```

and replace it with:

```
<img alt="A Lesson on:" src="../pictures/lava.gif"   align=left>
<b><I><font size=+1>
"Nature raves savagely, threatening the lands"
</font></I></b><br>
-- <a href="http://magic.geol.ucsb.edu/~fisher/pliny.htm">
Pliny the Elder</a>, who died of asphyxiation after
observing the destruction of Pompeii by the
79 A.D. eruption of Mount Vesuvius.
<p>
<b><font size=+4 color=#FF66FF>V</font>
<font size=+3 color=#DD0055>OLCANO WEB</font></b>
<p>
In this lesson you will use the Internet
to research information on volcanoes and then write a report on your
results.
<br clear=left>
```

3. **Save** and **Reload** in your web browser.

**NOTE: Everything between the `<img... align=left>` tag and the `<br clear=left>` tag will be placed adjacent to the picture -- the image is aligned left and the other HTML fills the empty space.**

**We also put the quotation at the top to attract attention. With this new layout, the `<blockquote>` tag is not effective, so it was removed.**

You may want to compare your HTML file to an example of how it should look at this point.

# Text Justification/alignment

We will add in one more alignment tag, the "divisions" tag ** ... **, introduced as part of HTML 3.0. All text within the tag is justified according to the align attribute:

```
<div align=left>...</div>
<div align=right>...</div>
<div align=center>...</div>
```

Note that the center attribute accomplishes the same effect as the Netscape ** ... ** tag.

We will now use this tag to make the opening quotation on our cover page have its text aligned to the **right** margin of the page:

1. Open your **index.html** file in your text editor.
2. Add the ** ** and ** ** tags as shown below:

```
<div align=right>
<b><I><font size=+1>
"Nature raves savagely, threatening the lands"
</font></I></b><br>
-- <a href="http://magic.geol.ucsb.edu/~fisher/pliny.htm">
Pliny the Elder</a>, who died of asphyxiation after
observing the destruction of Pompeii by the
79 A.D. eruption of Mount Vesuvius.
<p>
<b><font face="Arial,Helvetica" size=+4 color=#FF66FF>V</font>
<font face="Arial,Helvetica" size=+3 color=#DD0055>OLCANO WEB</font></b>
</div>
```

3. **Save** and then **Reload** in your web browser.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. How do you make certain parts of your web page aligned to the center of the page?
2. What is the correct way to make an inline image aligned to the right side of the web page?

3. How can you insert 10 pixels of space between a left side aligned image and the text that wraps it?
4. How can you create text that is right justified?

# Independent Practice

Try making some of your text centered- aligned or right margin aligned.

---

# Coming Next....

**Let's set up** **a pretty table...**

**GO TO.... | Lesson Index | previous: "Easy Horizontal Rules" | next: "Tables" |**

---

**Writing HTML:** Lesson 20: Extra Alignment
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

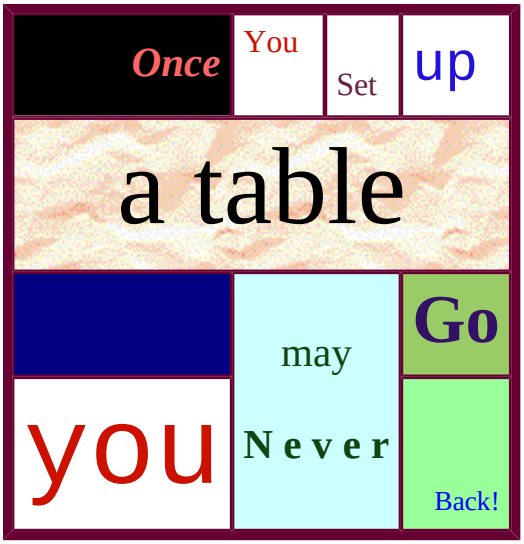
The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut20.html

# 21. Tables

Let's set the table...

...and completely **revolutionize** ordinary web pages




## Objectives

After this lesson you will be able to:

- Design a web page table with rows and columns of text in a gridded display
- Write the HTML for integrated layout schemes of text and pictures
- Write the HTML for an invisible table that creates side-by-side columns
- Create a table that has different colored cells
- Create a table that uses images as backgrounds in table cells

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now. Also, you may want to first look at the test page to see of your browser supports the tags used in this lesson.**

Tables were introduced into HTML 3.0 and further enhanced by Netscape to add another dimension to web page design. They provide a structure for organizing other HTML elements into "grids" on the page. One of the more obvious uses of tables is when you have to format... columnar tables of text! But, tables also open up the door for many other page layout options.

The HTML for tables can look very complex -- but we will start simple and build up some tables for our *Volcano Web* lesson.

For starters, keep in mind this concept:

**Tables start being built from the top left, then across the columns of the first row, then to the second row, across the columns of the second row...**

```
-->  -->  -->  -->  -->  -->
_____________________________|
|
-->  -->  -->  -->  -->
```

We will call each grid in a table a **cell**.

# Basic Table Tags

The HTML for the basic table structure is shown below:

**HTML**

```
<table border=1>
  <tr>
  <td>Row 1 col 1</td>
  <td>Row 1 col 2</td>
  <td>Row 1 col 3</td>
  </tr>

  <tr>
  <td>Row 2 col 1</td>
  <td>Row 2 col 2</td>
  <td>Row 2 col 3</td>
  </tr>

  <tr>
  <td>Row 3 col 1</td>
  <td>Row 3 col 2</td>
  <td>Row 3 col 3</td>
  </tr>

</table>
```

**Result**

| | | |
|---|---|---|
| Row 1 col 1 | Row 1 col 2 | Row 1 col 3 |
| Row 2 col 1 | Row 2 col 2 | Row 2 col 3 |
| Row 3 col 1 | Row 3 col 2 | Row 3 col 3 |

The **`border=1`** attribute in the **`<table>`** tag instructs the browser to draw a line around the table with a thickness of 1 pixel. Note how each row is defined by **T**able **R**ow tags **`<tr>...</tr>`** and then cells in each row are defined by **T**able **D**ata **`<td>...</td>`** tags. Each `<td>...</td>` tag can contain any type of HTML tag we have used in this tutorial -- headers, styled text, hypertext links, inline graphics, etc. Within this tag you can uses several attributes to control the alignment of items in a single cell:

**Horizontal Alignment** **Vertical Alignment**

- `<td align=left>` aligns all elements to the left side of the cell (this is the default setting)
- `<td align=right>` aligns all elements to the right side of the cell
- `<td align=center>` aligns all elements to center of the cell
- `<td valign=top>` aligns all elements to the top of the cell
- `<td valign=bottom>` aligns all elements to the bottom of the cell
- `<td valign=middle>` aligns all elements to the middle of the cell (this is the default setting)

You can combine these attributes:

```
<td align=left valign=bottom>
```

This HTML will produce a cell with items aligned along the left and bottom of the cell.

# Rows and Columns

The table shown above is pretty simple and square -- three rows by three columns. Look what we can do if using the **`colspan`** and **`rowspan`** attributes in the **`<td>...</td>`** tags.

**HTML**

```
<table border>
  <tr>
  <td>Row 1 col 1</td>
  <td align=center colspan=2>
   Row 1 col 2-3</td>
  </tr>

  <tr>
  <td>Row 2 col 1</td>
  <td>Row 2 col 2</td>
  <td>Row 2 col 3</td>
  </tr>

  <tr>
  <td>Row 3 col 1</td>
  <td>Row 3 col 2</td>
  <td>Row 3 col 3</td>
  </tr>
</table>
```

**Result**

** Note the attribute for the second cell of the first row -- it **spans** 2 columns. We have also aligned the text in the center of this cell.

| | | |
|---|---|---|
| Row 1 col 1 | Row 1 col 2-3 | |
| Row 2 col 1 | Row 2 col 2 | Row 2 col 3 |
| Row 3 col 1 | Row 3 col 2 | Row 3 col 3 |

Okay, now that we have had a cell span two columns -- let's make a cell that spans two rows. Remember to follow the HTML as it builds from the top left across, then down, then across...

**HTML** **Result**

```
<table border=1>
  <tr>
  <td>Row 1 col 1</td>
  <td align=center colspan=2>
   Row 1 col 2-3</td>
  </tr>

  <tr>
  <td>Row 2 col 1</td>
  <td valign=top rowspan=2>
  Row 2 col 2</td>
  <td>Row 2 col 3</td>
  </tr>

  <tr>
  <td>Row 3 col 1</td>
  <td>Row 3 col 3</td>
  </tr>
</table>
```

<table border=1>
<tr><td>Row 1 col 1</td><td align=center colspan=2>Row 1 col 2-3</td></tr>
<tr><td>Row 2 col 1</td><td valign=top rowspan=2>Row 2 col 2</td><td>Row 2 col 3</td></tr>
<tr><td>Row 3 col 1</td><td>Row 3 col 3</td></tr>
</table>

There is still quite a bit more to cover, but let's stop looking at these boring examples and work on our web page...

# A Data Table

Our Introduction page contains a table ("Volumes of Some Well Known Eruptions") that we first created in lesson 9 using the preformat tags **`<pre>...</pre>`**. We will now enhance that chart using a table tags.

1. Open up the **`intro.html`** file in your text editor.
2. Delete everything inside and including the **`<pre>...</pre>`** tags.
3. In the same place put:

```
  <table border>
  <tr>
  <th>Eruption</th>
  <th>Date</th>
  <th>Volume in km<sup>3</sup></th>
  </tr>

  <tr>
  <td>Paricutin, Mexico</td>
  <td align=center>1943</td>
  <td align=center>1.3</td>
  </tr>

  <tr>
  <td>Mt. Vesuvius, Italy</td>
  <td align=center>79 A.D.</td>
```

```
<td align=center>3</td>
</tr>

<tr>
<td>Mount St. Helens,<br>Washington</td>
<td align=center>1980</td>
<td align=center>4</td>
</tr>

<tr>
<td>Krakatoa, Indonesia</td>
<td align=center>1883</td>
<td align=center>18</td>
</tr>

<tr>
<td>Long Valley, California</td>
<td align=center>pre-historic</td>
<td align=center>&gt;450 &amp; &lt;700</td>
</tr>

<tr>
<td>Yellowstone, Wyoming</td>
<td align=center>pre-historic</td>
<td align=center>400</td>
</tr>
</table>
```

**NOTE: Look at the HTML for the first row. The Table Header tags `<th>...</th>` function exactly like the `<td>...</td>` tags except that any text is automatically made bold and all items are automatically centered.**

**Also see that in the cells for "Long Valley" you must use the HTML for the special characters to display the symbols for "<", ">", and "&" (See lesson 10)**

4. **Save** and **Load** into your web browser. You can compare your attempt with this sample of how the table should look at this point.

   **NOTE: The table may not be completely distinct as it sits on a solid black background.**

Now let's add some more things to our table.

Some browsers allow you to specify other settings for the lines that make up the table. These are the attributes for the table tag:

```
<table border=X cellpadding=Y cellspacing=Z>
```

where X is the width (in pixels) of the outer border of the table. The attribute **`cellpadding`** specifies how much empty "space" exists between items in the cells and the walls of the cells -- high values for Y will make the cells larger ("padding" the cell). The attribute **`cellspacing`** specifies (in pixels) the width between the inner lines that divide the cells. To see

what effect these attributes have see the examples on the table test page.

Modify your **<table>** tag to read:

```
<table border=3 cellpadding=4 cellspacing=8>
```

1. The **<caption>** tag places a string of text (centered to the width of the table) as a title/caption for the table. Modify the lines of your table to read:

   ```
   <table border=3 cellpadding=4 cellspacing=8>
   <caption><b><font size=+1>
   Volumes of Some Well-Known
   Volcanic Eruptions</font></b></caption>
   ```

   You can include and HTML inside the **<caption>** tag; just makes sure that it is **within** the **<table>...</table>** tags.
2. And just for fun, let's color the text in the **<th>...</th>** tags an orange color (for more on coloring text, see lesson 19). Modify the HTML for the first row to look like:

   ```
   <tr>
   <th><font color=#EE8844>Eruption</font></th>
   <th><font color=#EE8844>Date</font></th>
   <th><font color=#EE8844>Volume in km<sup>3</sup></font></th>
   </tr>
   ```

3. And let's move that table to the center of the page. If your web browser supports the ** ... ** tag, then just surround the table with these tags. For more on text alignment, see lesson 20.
4. **Save** and **Reload** into your web browser. You can compare your attempt with this sample of how the table should look at this point. Pretty good, eh?
5. Finally, we will add a column to the left side -- we want to show the grouping of historic volcanic eruptions (the first four rows) and the pre-historic ones (the last two rows). We now add an empty cell by adding **<th></th>** to the first row -- the reason why we did this should be come clear as we build this new column in the next few steps.

   ```
   <tr>
   <th></th>
   <th><font color=#EE8844>Eruption</font></th>
   <th><font color=#EE8844>Date</font></th>
   <th><font color=#EE8844>Volume in km<sup>3</sup></font></th>
   </tr>
   ```

6. We go to the first row, and add a first cell that spans the next 4 rows:

   ```
   <tr>
   <td rowspan=4>
   <font color=#EE8844>
   <i>eruptions<br>
   observed<br>
   by humans</i>
   </font></td>
   ```

```
<td>Paricutin, Mexico</td>
<td align=center>1943</td>
<td align=center>1.3</td>
</tr>
```

> **NOTE: We have added some <br> tags so that this first column does not become too wide. You might want to investigate for yourselves the effect of omitting these tags.**

7. And in the fifth row, we add a cell that spans the next 2 rows (note again the use of special character codes to represent "**>450 & <700**":

```
<tr>
<td rowspan=2>
<font color=#EE8844>
<i>inferred<br>
by study<br>
of deposits</i>
</font></td>
<td>Long Valley, California</td>
<td align=center>pre-historic</td>
<td align=center>&gt;450 & &lt;700</td>
</tr>
```

8. **Save** and **Reload** again into your web browser. You can compare your attempt with this sample of how the table should look at this point. This is all we will add to this table.

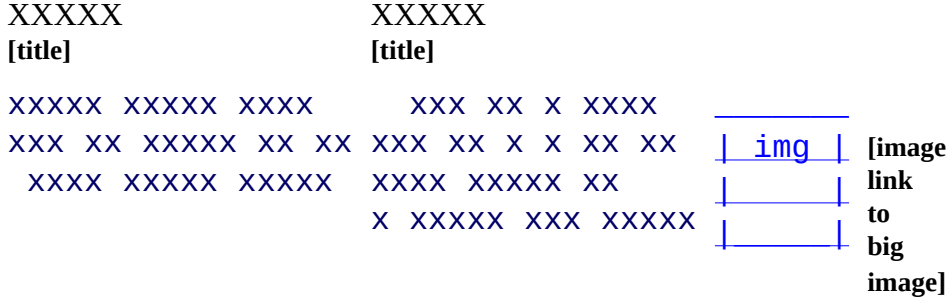
# Invisible or "Phantom" Tables

A table with borders is useful for charts and data but other times we want to create a grid-like layout that does not have the borders. We like to call these "phantom" tables because to the reader it may not be obvious that they are looking at a table!

A phantom table is built in the same manner as the table we built above except the **<table>** tag looks like:

```
<table border=0>
```

If you look at the very top of this page, the top most part is actually a Phantom table that contains in one of its cells a second table with borders! However, we are jumping ahead. For our example, we will reformat the file **usa.html** (Volcanoes in the USA) into a two column format. Rather than having page-wide paragraphs stacked vertically on the page, we will put them side by side like this sketch:



xxx xx xx
**[hypertext link to big image]**

Note that the right hand column also has an adjacent picture that is a hyperlink to a full screen image (see lesson 8e).

1. Open up the file **usa.html** in your text editor.
2. Find the section that looks like:

```
<font size=+1><b>Mount St Helens</b></font><br>
On May 18, 1980, after a long period of rest,
this quiet mountain in Washington provided
<a href="msh.html">detailed observations</a> on
the mechanics of highly explosive eruptions.

<p>
<font size=+1><b>Long Valley</b></font><br>
This field seismometer measures earthquakes
associated with subsurface volcanic forces and
may help to predict future events. It sits on a
plateau known as the "Volcanic Tableland" formed
by a major eruption  600,000 years ago.<p>
<a href="../pictures/seismo.jpg">
<img src="../pictures/stamp.gif">
-- [full size image] --</a>
```

and replace it with the following HTML:

```
<table border=0 cellpadding=6 cellspacing=2>
<tr>
<td><font size=+1><b>Mount St Helens</b></font></td>
<td colspan=2><font size=+1><b>Long Valley</b></font></td>
</tr>

<tr>
<td valign=top>On May 18, 1980, after a
long period of rest, this quiet mountain
in Washington provided <a href="msh.html">
detailed observations</a> on
the mechanics of highly explosive eruptions.
</td>

<td valign=top>
This field seismometer measures earthquakes
associated with subsurface volcanic forces
and may help to predict future events. It
sits on a plateau known as the
"Volcanic Tableland" formed by a major
eruption  600,000 years ago.
</td>
<td valign=top><a href="../pictures/seismo.jpg">
```

```
<img src="../pictures/stamp.gif"></a>
</td>
</tr>

<tr>
<td colspan=3 align=right>
<a href="../pictures/seismo.jpg">
-- [full size image] --</td>
</tr>
</table>
```

**NOTE: Look carefully at the HTML here. We actually are using a 3 column table -- the first paragraph (Mount St Helens) becomes the left column. The right column includes one column of text and another column for a small image. A bottom row, aligned right and spanning 3 columns, is used to hold the hypertext link that leads to the same graphic as the thumbnail image**

3. **Save** and **Reload** again into your web browser.

# Splitting a Long List

Another handy use for invisible tables is to transform a long list of items (created with the **`list`** tags, see lesson 6). Lists grow downward on a page, and can waste valuable real estate on the right side of the page.

The effect is to transform a list:

**Long Linear List**

```
<ul>
<li> xxxxxx
<li> xxxx xxxx
<li> xxx  x xxxx
<li> xxx xxxxx
<li> xx x  xxxxx
<li> xxx xx
<li> xxxx x
<li> xxx  x xxx
</ul>
```

**to**

**Column 1**

```
<ul>
<li> xxxxxx
<li> xxxx xxxx
<li> xxx  x xxxx
<li> xxx xxxxx
</ul>
```

**Column 2**

```
<ul>
<li> xxx xx
<li> xxxx x
<li> xxx  x xxx
</ul>
```

We will now break up the list of resources on our Resource Projects page (file **`proj.html`**).

1. Open your **`proj.html`** file in your text editor.
2. Look for the **`<ul>...</ul>`** list under the **References** heading and replace all of it with:

```
<table border=0 cellpadding=2 cellspacing=2>
<tr>
<td valign=top>
```

```
<ul>
<li><a href="http://www.avo.alaska.edu/">
Alaska Volcano Observatory</a>
<li><a href="http://vulcan.wr.usgs.gov/home.html">
Cascades Volcano Observatory</a>
<li><a href="http://www.dartmouth.edu/~volcano/">
The Electronic Volcano</a>
<li><a href="http://www.geo.mtu.edu/volcanoes/">
Michigan Tech Volcanoes Page</a>
<li><a href="http://volcano2.pgd.hawaii.edu/eos/">
NASA Earth Observing System (EOS) IDS Volcanology Team</a>
<li><a href="http://www.geol.ucsb.edu/~fisher/">
Volcano Information Center</a>
</ul>
</td>
<td valign=top>
<ul>
<li><a href="http://vulcan.wr.usgs.gov/Servers/earth_servers.html">
Volcano/Earth Science-Oriented Servers</a>
<li><a href="http://volcanoes.usgs.gov/">
US Geological Survey Volcanic Hazards Program</a>
<li><a href="http://www.nmnh.si.edu/gvp/">
Global Volcanism Program (GVP) </a>
<li><a href= "http://hvo.wr.usgs.gov/volcanowatch/">
Volcano Watch Newsletter</a>
<li><a href="http://library.advanced.org/17457/">
Volcanoes Online</a>
<li><a href="http://volcano.und.edu/">
VolcanoWorld</a>
</ul>
</td>
</tr>
</table>
```

**NOTE: We simply have taken one `<ul>...</ul>` list and broke it into two complete lists, each in the cell of an invisible table with one row and two columns.**

3. **Save** and **Reload** in your web browser. You can compare your attempt with this sample of how the table should look at this point.

# Coloring Tables

Most web browsers now support HTML to color tables, rows, and individual table cells. In fact, we have used it throughout this tutorial -- on the About the Tutorial page, the lesson index, and throughout the lessons when we display HTML examples.

This is an effective means to add graphic/color elements to a web page without attaching a lot of bandwidth consuming images.

**NOTE: If your computer display is running at less than 24-bit "True Color" settings, e.g. at 256 colors, then you may not be able to see all of the colors used in this section. Most computers sold since 1999 are capable of displaying at 24-bit color.**

You might want to look at the last example on the test page to see if your browser supports coloring of tables. The example on "colored tables" uses the colors used in this lesson.

We will uses the hex color codes that we used in lesson 16 to color the background of web pages and in lesson 18 to color text.

It is as simple as inserting **`bgcolor=#FF0000`** as an attribute to the different **`<table>`** tags:

| Table Part | HTML |
|---|---|
| **table**<br>color the area behind the entire table | `<table bgcolor=#880000>` |
| **row**<br>color the area behind a single row | `<tr bgcolor=#880000>` |
| **cell**<br>color the area behind a single cell | `<td bgcolor=#880000>` |

Now we will add some color to the data table you created for the Introduction page. We will not add great gobs of colors (but feel free to experiment on your own pages). In our case, we will simply add the HTML to make the cells that hold the row and column headings a lighter shade of gray than the background black.

1. Open your **`intro.html`** file in your text editor.
2. First find all of the **`<th>`** tags that hold the column headings:

```
<th><font color=#EE8844>Eruption</font></th>
<th><font color=#EE8844>Date</font></th>
<th><font color=#EE8844>Volume in km<sup>3</sup></font></th>
```

   and add the attribute to color those cells grey (#222222):

```
<th bgcolor=#222222><font color=#EE8844>Eruption</font></th>
<th bgcolor=#222222><font color=#EE8844>Date</font></th>
<th bgcolor=#222222><font color=#EE8844>Volume in km<sup>3</sup></font></th>
```

3. Now, find the two tags that format the row labels and add the same color tag so that they read:

```
<td  bgcolor=#222222 rowspan=4>
<font color=#EE8844>
<i>eruptions<br>
observed<br>
by humans</i>
</font></td>
   :
   :
```

```
   :
<td  bgcolor=#222222 rowspan=2>
<font color=#EE8844>
<i>inferred<br>
by study<br>
of deposits</i>
</font></td>
```

4. Save and Reload in your web browser. You can compare your attempt with this sample of how the table should look at this point.

And you can even do more than use solid color backgrounds for table cells. In lesson 16 we could use a tiled image (a graphic image that can repeatedly fill a space) as a background for an entire page via the **BODY** tag. You can use a similar format to designate that table cells are filled with repeating patterns. You should be aware of the differences in how NetScape and Internet Explorer tags handle this modifier:

| Table Part | HTML | Notes |
|---|---|---|
| **table**<br>fill all cells with the same pattern | `<table background="image.gif">` | In NetScape browsers, this will fill all cells in the table; in Internet Explorer, the entire table area (including cell walls) will be covered with the pattern. |
| **row**<br>fill all cells in one row | `<tr background="image.gif">` | Will not work in Internet Explorer |
| **cell**<br>fill a single cell a same pattern | `<td background="image.gif">` | works the same in NetScape and Internet Explorer. |

If you look at the table at the top of this lesson page, you can see that some cells have solid color backgrounds, but the cell that spans the second row uses a graphic to fill the text behind the word **table** with a pattern or crumpled paper:

We will now modify the table you made before to apply a graphic background to the headings in our chart of volcanos.

1. Go to the Lesson 21 Image Studio to get a copy of the small (<1k) image that we will use as a background tile image for part of our table. Save it inside your **pictures** directory/folder as a file named **pattern.gif**.
2. Open your **intro.html** file in your text editor.
3. Edit the lines for the first table cell row that read:

```
<tr>
```

```
<th></th>
<th bgcolor=#222222><font color=#EE8844>Eruption</font></th>
<th bgcolor=#222222><font color=#EE8844>Date</font></th>
<th bgcolor=#222222><font color=#EE8844>Volume in km<sup>3</sup></font></th>
</tr>
```

We will now change the background solid color in the cells of the top row to use the **pattern.gif** file:

```
<tr>
<th></th>
<th background="../pictures/pattern.gif">
  <font color=#EE8844>Eruption</font></th>
<th background="../pictures/pattern.gif">
  <font color=#EE8844>Date</font></th>
<th background="../pictures/pattern.gif">
  <font color=#EE8844>Volume in km<sup>3</sup></font></th>
</tr>
```

4. Save and Reload in your web browser. You can compare your attempt with this sample of how the table should look at this point.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Tables quickly become complicated, so double-check that you have defined the rows and columns correctly.

# Review

Review topics for this lesson:

1. What is the order in which a web browser interprets the elements of a table tag?
2. What is the difference between **<td>...</td>** and **<th>...</th>** tags?
3. Where does the **colspan=X** attribute go? What does it do?
4. How do you create a table that has no visible margins?
5. How do you color a single row of a table?
6. How can you use a pattern as a background in the last row of your table? for all cells in your table?

# Independent Practice

Look at your own web pages and find a place where a table may give you a better page layout. Or, add a chart or list of data to your web pages and use a table to format it. Try creating a table with colored cells, or like we have done in this tutorial, use the colors on an invisible table to color blocks of areas on a web page.

# More Information

Tables are pretty much the norm these days for web page design. In fact, you can peek at the web source of many sites and see tables inside of tables inside of... There is a drawback to pages that are all or heavily designed by tables... the entire table structure from the first **<table>** to the last **</table>** must all be downloaded before any content will appear. Therefore, if you have a lot of things inside of your table, a viewer accessing your page may stare a long time (especially if they are connecting via a slower modem) at a blank page while everything comes in. We have seen web sites that take more than 2 minutes to display any content because of complicated and large tables.

What can you do? Always test your pages on a slower connection or older computer than your own! If your entire page is layed out in tables, make sure that all **<img...>** tags have their **width=** and **height=** dimensions included (this helps the browser present the page quicker if it knows the dimensions of the images. If possible, try to put at least one line of text **before** a complicated table, so at least something appears qucikly when a viewer first reaches your page.

Now how would you like to learn an extra credit table trick? You can use tables to set up a page so the content is always centered, no matter the size of the browser window. If you do not believe us, try looking an example. To better understand what the tables are doing, look at the "revealed" version of this example that has the borders big and thick so you can see the tables.

How is it done? The "trick" is that with tables you can use relative sizes for both it's width and height. Here is the HTML to produce the example:

```
<html>
<head>
<title>Always Centered</title>
</head>
<body bgcolor=#000000 text=#FFFFFF link="#CC9966" vlink="#9999FF">
<table border=0 width=100% height=100%>
<tr>
<td align=center valign=middle>
<img src="../pictures/lava.gif" alt="lava!" width="300" height="259">
<p>
No matter How much you shrink or stretch, I stay centered!
</td>
</tr>
</table>
</body>
</html>
```

The **table** tag contains options to size the table to the full width and height of the browser window. It's single cell **<td>..</td>** is set to be centered horizontally and vertically. Try it yourself and experiment!

Do you want more? See another example of how you can also use table sizing to make a title that always stretches across the screen. In this case, we make a table with a width of 100%. Each table cell contains one letter, and we divide the number of letters into 100 to get the proportional width of each cell:

```
<html>
<head>
<title>Stretch Text</title>
</head>
<body bgcolor=#000000 text=#FFFFFF link="#CC9966" vlink="#9999FF">
```

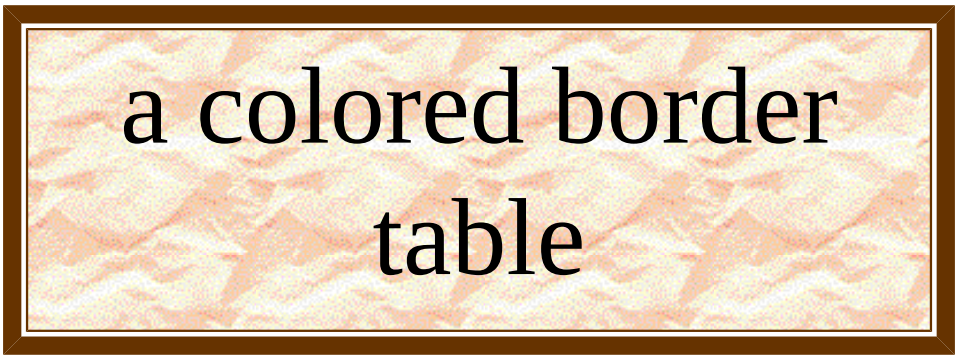
```
<table border=0 width=100%>
<tr>
<td align=center width=12%><font size=+3>V</font></td>
<td align=center width=12%><font size=+3>O</font></td>
<td align=center width=12%><font size=+3>L</font></td>
<td align=center width=12%><font size=+3>C</font></td>
<td align=center width=12%><font size=+3>A</font></td>
<td align=center width=12%><font size=+3>N</font></td>
<td align=center width=12%><font size=+3>O</font></td>
<td align=center width=12%><font size=+3>!</font></td>
</tr>
</table>
</body>
</html>
```

Again, you can look at our "revealed" version with big thick borders so you can see what the tables are doing.

We found these tips from the glassdog design-o-rama site. Play with it and see what else you can do!

But wait! Here is more! You can also add the **`bordercolor`** attribute to your **`<table>`** tag to shade the colors of a table with visible borders. Compare:

to one with the borders colored:




(This may not work on all web browsers). It creates a color shaded 3D look by having the top and left sides of the table lighter than the right and bottom sides. Creating this is as simple as modifyng our **`<table>`** tag to read:

```
<table border=8 cellpadding=8 cellspacing=2 bordercolor="#663300">
```

For more resources on HTML tables, look at the sites listed in our references section.

---

# Coming Next....

Even **more** things to throw into your lists and toss around your images.

**GO TO.... | Lesson Index | previous: "Extra Alignment" | next: "More with Lists and Images" |**

---

**Writing HTML:** Lesson 21: Tables
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut21.html

# 22. More for Images and Lists

Vaporize those annoying boxes around graphic buttons,

or add extra borders plus:

- fiddle
  - with
    - bullets

A. plus
b. change the
III. **number**
iv. **style**
109. and **value**
FFFFF. of list items

## Objectives

After this lesson you will be able to:

- Create a hyperlinked-graphic without a bounding border
- Place a framing border around an inline image
- Write the HTML for an un-ordered lists that uses different bullet styles
- Write an ordered list that numbers items via capital and small alphabetical letters or large or small Roman numerals
- Modify an ordered list to start counting from any positive value

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now. Also, you may want to first look at the test page to see of your browser supports the tags used in this lesson.**

## No Hypertext Borders on Hyperlinked Images

In lesson 8e we learned how to make a small graphic image act as a hyperlink to some other web page or a larger size copy of a picture. We noticed that the web browser places a bounding box around the graphic, to identify it as being

"hyper" like normal hypertext items:

However, the box is sometimes distracting, especially if we have an image that has non-rectangular borders. The user can typically determine if a picture is "hyper" simply by noting a change in the cursor as they move a mouse over the image (it usually changes to a "hand" when it is over an active link).

You can hide the box by adding an attribute to the **`<img...>`** tag:

```
<a href="bigpict.gif"><img src="lilpict.gif" border=0>
```

In this example, the inline image **`lilpict.gif`** acts as a hyperlink to the bigger image **`bigpict.gif`**. The **`border=0`** attribute has no meaning if the **`<img..>`** tag is not inside of an **`<a href=...</a>`** tag.

We have two places in our *Volcano Web* web site where we have hyperlinked inline images -- do you remember where?

The first is in the **`usa.html`** file, where a small image of a seismometer links to a larger sized image. The second is the arrow button on the **`msh.html`** page that leads back to the lesson.

1. Open both the **`usa.html`** and the **`msh.html`** in your text editor.
2. Locate the place where we have a small image that links to something else.
3. Modify each **`<img src=...>`** tag that is a hyperlink so that it contains a **`border=0`** attribute. For example, in the **`msh.html`** file it should look like:

   ```
   <a href="usa.html">
   <img src="../pictures/left.gif" alt="** " border=0>
   Return to <i>Volcano Web</i></a>
   ```

4. **Save** and **Load** the **`usa.html`** and **`msh.html`** files in your web browser.
5. If the small images have lost their borders but still connect to their intended target when clicked, your job is done.

# Putting Borders around Images

And now we show you how to do the opposite of taking away borders; adding BIG THICK borders! You can use the same **`border=X`** option to **add** a border around an image. The number you use for X determines the thickness, in pixels, of the border placed around an image:

**8 pixel border on an inline image**

```
<IMG SRC="pictures/disk.gif" WIDTH=48 HEIGHT=40 border=8>
```

Note that the border color is the color defined as the **TEXT** color in the **BODY** tag. (see lesson 16)

You can also use this on an image that is acting as a hyperlink:

**8 pixel border on an inline image, hyperlinked to another page**

```
<a href="page.html">
  <IMG SRC="pictures/disk.gif" WIDTH=48 HEIGHT=40 border=8></a>
```

Note that the border color is the color defined as the **LINK** color in the **BODY** tag. (see lesson 16)

# Bullets for Un-Ordered Lists

In lesson 6 we first created unordered **<ul>...</ul>** lists. The web browser automatically puts a bullet mark in front of each item -- and the bullets change if we create a list inside of a list. With some web browsers, you can specify in your HTML any of three bullet characters by adding an attribute to the **<ul>** tag:

```
<ul type=xxxx>
```

where **xxxx** can be:

- **type=circle**
- **type=square**
- **type=disc** [default bullet for top level list]

And here is even more! You can change the type *within* a list by putting the type attribute in the **<li>** tag:

| HTML | How it Looks |
|---|---|
| `<ul type=square>`<br>`<li>this is item 1`<br>`<li>this is item 2`<br>`<li>this is item 3`<br>`<li type=circle>`<br>`  Hey! how about these circle bullets?`<br>`<li type=circle>this is another item`<br>`<li type=circle>and one more`<br>`<li type=disc>`<br>`  Hey! how about this disc bullet?`<br>`</ul>` | **sample web page**<br>▪ this is item 1<br>▪ this is item 2<br>▪ this is item 3<br>○ Hey! how about these circle bullets?<br>○ this is another item<br>○ and one more<br>• Hey! how about this disc bullet? |

In NetScape browsers, the type specified in the **<li type=xxxx>** tag is used by all succeeding **<li>** tags that do

not have a **type** attribute until another bullet type is established. However, in Internet Explorer web browsers, list items without a **type** attribute will revert to the type set in the intitial **<ul>** tag. Therefore, our recommendation is that if you wish to change the bullet styles within a list is that you set it for each **<li>** tag that requires a different bullet.

Differences between the web browsers sure make life complicated! And this is another reason you should consider checking your site in more environments that just your own computer.

We will now change the bullet list style used in our Research Projects page (file **proj.html**).

1. Open your **proj.html** file in your text editor.
2. The first list in this file is an Ordered List **<ol>...</ol>** but we will now change it to an unordered list with **circle** bullets. Edit your HTML for this first list so that it looks like:

```
<ul type=circle>
<li>Type of volcano
<li>Geographic location
<li>Name, distance, and population of nearest major city
<li>Date of most recent eruption and date of most destructive
    eruption
<li>Other events associated with the last eruption (earthquakes,
    floods, mudslides, etc)
</ul>
```

3. **Save** and **Load** your file in your web browser. Compare your page with this sample of how your list should like at this point.

# Styles and Values for Ordered Lists

When we first created an ordered list **<ol>...</ol>** in lesson 6, we saw how the web browser automatically numbers the items, one for each **<li>** tag. What if we do not always want to use arabic numerals (1,2,3...)? Well, here are the answers, a **type=x** attribute for the **<ol>** and **<li>** tags inside:

| Arabic | Capital Letters | Small Letters | Large Roman | Small Roman |
|---|---|---|---|---|
| **<ol type=1>** | **<ol type=A>** | **<ol type=a>** | **<ol type=I>** | **<ol type=i>** |
| 1. I am first! | A. I am first! | a. I am first! | I. I am first! | i. I am first! |
| 2. I am second! | B. I am second! | b. I am second! | II. I am second! | ii. I am second! |
| 3. I am third! | C. I am third! | c. I am third! | III. I am third! | iii. I am third! |
| 4. I am fourth! | D. I am fourth! | d. I am fourth! | IV. I am fourth! | iv. I am fourth! |
| 5. I am fifth! | E. I am fifth! | e. I am fifth! | V. I am fifth! | v. I am fifth! |

We presented an example of using ordered lists within ordered lists to create outlines -- with the **type** attribute we can have pages with standard format:

I. Cheese in Pre-Historic time
    A. Africa
        1. Afar Triangle
        2. East Coast
    B. Asia
    C. Europe
        1. France
            a. Cave paintings depicting cheese harvesting
            b. Burial rituals inferred from dried cheese remnants
        2. British Isles
    D. North America
II. Cheese in the Middle Ages
    A. King Arthur's Longhorn
    B. Sharp Cheddar for the Crusades
III. Cheese in the Space Age

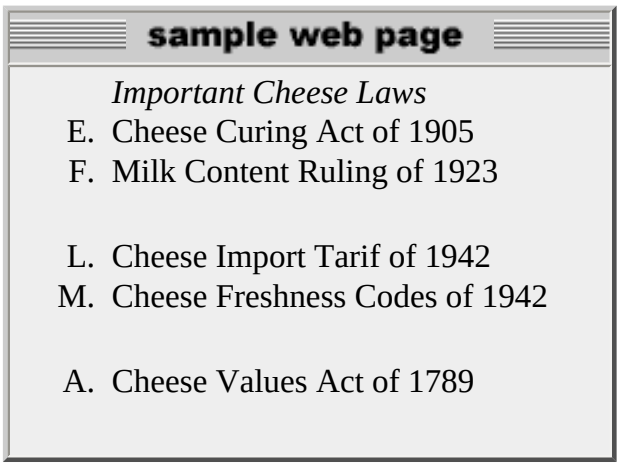
Another thing we can do with ordered lists is to have them start counting from some other value than 1. To do this, we add the **`start=y`** attribute to the **`<ol>`** tag. Note that even if we have some other **`type=x`** attribute, we still specify the starting value y as "2,3,10,100, etc". Look at some of these examples:

| Arabic | Capital Letters | Small Letters | Large Roman | Small Roman |
|---|---|---|---|---|
| `<ol type=1 start=11>` | `<ol type=A start=11>` | `<ol type=a start=11>` | `<ol type=I start=11>` | `<ol type=i start=11>` |
| 11. I am eleventh!<br>12. I am twelfth!<br>13. I am thirteenth!<br>14. I am fourteenth!<br>15. I am fifteenth! | K. I am eleventh!<br>L. I am twelfth!<br>M. I am thirteenth!<br>N. I am fourteenth!<br>O. I am fifteenth! | k. I am eleventh!<br>l. I am twelfth!<br>m. I am thirteenth!<br>n. I am fourteenth!<br>o. I am fifteenth! | XI. I am eleventh!<br>XII. I am twelfth!<br>XIII. I am thirteenth!<br>XIV. I am fourteenth!<br>XV. I am fifteenth! | xi. I am eleventh!<br>xii. I am twelfth!<br>xiii. I am thirteenth!<br>xiv. I am fourteenth!<br>xv. I am fifteenth! |

And finally you can change the numbering sequence **within** a list by adding a **`value=z`** attribute to the **`<li>`** tag. Look at this example:

| HTML | How it Looks |
|---|---|

```
<ol type=A start=5><i>
Important Cheese Laws</i>
<li>Cheese Curing Act of 1905
<li>Milk Content Ruling of 1923
<p>
<li value=12>Cheese Import Tariff of 1942
<li>Cheese Freshness Codes of 1942
<p>
<li value=1>Cheese Values Act of 1789
```




It may not be exactly clear (especially from this silly example!) when you might use these tags -- just keep them in your mind as potential tools in your web writing. We will demonstrate again on our Research Projects page (file **`proj.html`**). If you recall, in our lessons on tables we split the unordered list of reference sites into a two column list. Let's change this to an ordered list and use the **`type`** attribute to list them using small letters. Because we have actually **two** individual lists, see if you can determine why we would use the **`start`** attribute as well.

1. Open your **`proj.html`** file in your text editor.
2. Look at the table we created under the **References** section. Change both **`<ul>`** and **`</ul>`** tags to **`<ol type=a>`** and **`</ol>`**.
3. Now, we have 6 items in the first list so we want the second list to start numbering at 7. Modify the **`<ol>`** tag now to read:

   ```
   <ol type=a start=7>
   ```

4. **Save** and **Reload** your file in your web browser. Compare your page with this sample of how your list should like at this point. The first column should list items from "a" to "f" and the second column from "g" to "l".

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. How do you get rid of the hypertext box that surrounds a graphic that is hyperlinked to another item?
2. How can you put a 20 pixel border around an inline image? What color will it be?
3. How can you change the bullet markers for an unordered list?
4. Do you have to use the same bullet markers for an entire list? If not, how do you change the markers mid-list?
5. How can you create a list that marks items by Roman Numerals?

# Independent Practice

Experiment with different bullet markers and numbering styles in the lists of your own web pages. Can you think of some unique ways to use these added features? Can you make a complex outline with "standard" formatting? All of those

```
<ol>...</ol>
```

tags gets to be pretty complicated!

---

# Coming Next....

Map your way from an image... **Click n' go** hyperlinked sections within a graphic...

**GO TO.... | Lesson Index | previous: "Tables" | next: "Image Maps" |**

---

**Writing HTML:** Lesson 22: More for Images and Lists
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut22.html

# 23. Clickable Image Maps

Make different **parts** of an inline image hyper-active with Client-Side Image Maps...

## Objectives

After this lesson you will be able to:

- Understand the difference between server-side and client-side processes
- Create an inline image that has different portions hyperlinked to other web pages, pictures, and other sites on the Internet

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now. Also, you may want to first look at the test page to see of your browser supports the tags used in this lesson**.

In lesson 7a we learned how to write the HTML to place an inline image in our page and in lesson 8e we saw how we could make the entire picture act as a hyperlink to some other related item. Since the early days of the web, there has been a way to have different parts of an inline image be hyperlinked--known as a "clickable image map".

**But** until recently, the clickable image maps required that some things be processed from the web server. This is how it works:

1. Viewer sees page with clickable image map and... clicks on the part of it in the lower right hand corner.
2. Web browser says, "Hey! I got a mouseclick in this image! Gotta send a message to the web server- Someone clicked at these coordinates of this image."
3. The web server says."Hmmm. I got these coordinates for this image- The HTML file it was called from says look up the coordinates in this file... Okay, the file says, if the coordinates are in this rectangular region, then send the viewer to this URL" The server then returns this information to the web browser.
4. Web browser says,"Okay! The server said go to this URL- Let's go!"

This is a **server-side** process; one that was pretty efficient and a web server could handle it in a split second. But it meant that to use clickable image

maps one always had to have access to a web server.

Spyglass was the first browser built with the capability to handle the calculations and transmit the proper URL based upon coordinates that were built inside the HTML for a page. This is a **client-side** process. That entire conversation above now takes place just between the browser and itself! For more information about image maps, see the Imagemap Help Page (IHiP).

The HTML needed for a client-side clickable image map is:

```
<img src="image.gif" usemap="#map_name">
```

where **`image.gif`** is the file name of the image and **`map_name`** is an anchor link (see lesson 8d) elsewhere in the same HTML file:

```
<map name="map_name">
<area shape="rect" coords="x1,y1,x2,y2" hrEF=URL1>
<area shape="rect" coords="x1,y1,x2,y2" hrEF=URL2>
       :
       :
 </map>
```

Each line in the **`<map>...</map>`** tag identifies a different "hot" region, specified by the coordinates, **`coords=`**. **`x1,y1`** are the horizontal and vertical pixel locations for the upper left corner (relative to the upper left corner of the entire image), while **`x2,y2`** are the horizontal and vertical pixel locations for the lower right corner. **`URL1, URL2, ...`** are the URLs (or file names for local files) that the region should correspond to when clicked.

> **NOTE: To identify the coordinates for the hot region you will have to use some sort of graphics program. The Image Map Help Page or Yahoo lists several other utility programs that make this easy to do. For this lesson, we will be providing the exact coordinates for you.**

In this lesson we are going to add to our Volcano Terminology page (file **`term.html`**) a picture showing different kinds of volcanic eruptions. Each one will be hyperlinked to an external web site. In addition we will create two more links to our own (local) files.

1. First, you will need to download a copy of the image file we are using from the Lesson 23 Image Studio. Save this file as **`volc.jpg`** and make sure that you store it in your **`pictures`** folder/subdirectory.
2. Open your **`term.html`** file in your text editor.
3. Below the last paragraph ("...a historically active volcano on the island of Martinique. <p>"), add this HTML:

```
There are many different types of volcanic eruptions and landforms.
They can be classified according to the
<a href="explode.html">degree of "explosiveness"</a>
```

```
and the <a href="height.html">height</a> of the eruption column:
<p>
<center>
<font size=+2>
Investigate each type by clicking on a picture
</font><br>
<img src="../pictures/volc.jpg" usemap="#volcmap" border=0>
<p>
</center>
```

**NOTE: We have inserted an inline image that will reference a set of coordinates associated with the name "volcmap". The `<center>...</center>` tag aligns the image in the middle of the page (see lesson 20). The `border=0` attribute inside the `<img>` tag suppresses the display of a hypertext box around the image.**

**We also have built in links to two new HTML documents that we will create below.**

4. Next we will add the HTML for the map coordinates. This can be placed anywhere within the HTML document- it is HTML that is **not** displayed in the web browser. We suggest just adding it below the HTML you added in the previous step:

```
<map name="volcmap">
<area shape="rect"
 href="http://volcano.und.edu/vwdocs/frequent_questions/grp7/europe/question308.html"
   coords="48,46,204,153">
<area shape="rect" href="explode.html"
   coords="0,66,26,227">
<area shape="rect" href="height.html"
   coords="95,283,378,309">
<area shape="rect" href="http://www.geo.mtu.edu/volcanoes/pinatubo/"
   coords="321,154,468,261">
<area shape="rect" href="http://stromboli.net/"
   coords="172,155,318,274">
<area shape="rect" href="http://hvo.wr.usgs.gov/volcanowatch/"
   coords="36,155,168,276">
<area shape="rect" href="http://www.geo.mtu.edu/volcanoes/santamaria/"
   coords="205,3,343,123">
</map>
```

**NOTE: You should see that 5 of the 7 defined areas connect to remote internet sites. The two others are local documents that we will create in the next step.**

5. Save your **`term.html`** file.
6. We now have to create two more HTML files that will be linked from the image map. Create a new document in your text editor and place in it:

```
<html>
<head>
<title>Explosiveness</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#EEEEBB
LINK=#44DDFF VLINK=#00FF88>
<h5>Volcano Web /
<a href="index.html">Index</a> /
<a href="term.html">Volcano Terminology</a> /
</h5>

<h1 align=center>Explosiveness</h1>
The <b>explosiveness</b> of an observed volcanic
eruption is estimated by the calculated force of
the eruption. Experiments have shown that when water
comes in contact with hot magma, the eruption is
much stronger- this is known as <b>hydro-volcanism</b>.
<p>
For pre-historic eruptions, the explosiveness is estimated
by the degree of fragmentation of small volcanic particles.
A more explosive eruption will "shatter" volcanic tephra
much more than a less explosive eruption.
<p>
<a href="term.html">
<img src="../pictures/left.gif" alt="** " border=0>
Return to <i>Volcano Web</i></a>

<hr>
<address>
<b><a href="index.html">Volcano Web</a> :
<a href="term.html">Volcano Terminology</a> :
Explosiveness</b> <p>

created by Lorrie Lava,
<a href="mailto:lava@pele.bigu.edu">
lava@pele.bigu.edu</a><br>
```

```
Volcanic Studies, <a href="http://www.bigu.edu/">
Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
<tt>URL: http://www.bigu.edu/web/explode.html</tt>
<p>
</body>
</html>
```

7. Save this document in the same directory as your other HTML files and call it **`explode.html`**
8. Now create a second HTML document in your text editor and in it enter:

```
<html>
<head>
<title>Height of Eruption Column</title>
</head>

<BODY BGCOLOR=#000000 TEXT=#EEEEBB
LINK=#44DDFF VLINK=#00FF88>
<h5>Volcano Web /
<a href="index.html">Index</a> /
<a href="term.html">Volcano Terminology</a> /
</h5>

<h1 align=center>Height of Eruption Column</h1>
The <b>height</b> of a volcanic eruption cloud
(in kilometers) is taken from direct observation
or from estimates based upon historic accounts.
<p>
For pre-historic eruptions, this scale is calculated
based upon the <b>dispersal</b> of the volcanic products-
i.e. how far and wide they are scattered. Volcanic
eruptions that have very tall columns will spread
tephra far and wide.
<p>
<a href="term.html">
<img src="../pictures/left.gif" alt="** " border=0>
Return to <i>Volcano Web</i></a>
```

```
<hr>
<address>
<b><a href="index.html">Volcano Web</a> :
<a href="term.html">Volcano Terminology</a> :
Height of Eruption Column</b> <p>

created by Lorrie Lava,
<a href="mailto:lava@pele.bigu.edu">
lava@pele.bigu.edu</a><br>
Volcanic Studies,
<a href="http://www.bigu.edu/">
Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
<tt>URL: http://www.bigu.edu/web/height.html</tt>
<p>
</body>
</html>
```

9. Save this document in the same directory as your other HTML files and call it **`height.html`**
10. Now open the **`term.html`** file in your web browser. The image of the different volcanoes should appear and as you move the mouse over the different regions of the image, the cursor should change to a "hand" and somewhere in the bottom of the web browser window, there should be an indicator of the URL the "hot" region will link to when clicked.

You should compare your web page to this sample of how your clickable image page should look.

# Covering Your Bases

A few years ago client-side image maps was a new development in HTML, and you had to allow for web browsers that did not support them. This is less the case now. Netscape's documentation provides some examples for handling this situation. If you have access to server-side image mapping, you can set it up so that if the browser does not support the client-side interpretation, it will then refer to the server.

All in all, it is better to use client-side image maps because it avoids extra communications from web browser to web server and back.

For our case we can set up a special page that has a message for web browsers that do not support client-side image maps:

1. First we will need to create a new HTML file called **`nomap.html`**. Open a new file in your text editor and in it put:

```
<html>
<head>
<title>No Image Map Available</title>
</head>
<BODY BGCOLOR = #000000 TEXT=#EEEEBB
LINK=#44DDFF VLINK=#00FF88>
<h5>Volcano Web /
<a href="index.html">Index</a> /
<a href="term.html">Volcano Terminology</a> /
</h5>

<h1 align=center>Sorry!</h1>
Apparently your web browser does not support
client-side image maps. You can still reach the
information by following these links:
<ul>
<li><a href="explode.html">Explosiveness</a>
<li><a href="height.html">Height of Eruption Column</a>
<p>
<li><a
href="http://volcano.und.edu/vwdocs/frequent_questions/grp7/europe/question308.html">Surtseyan</a>
<li><a href="http://www.geo.mtu.edu/volcanoes/santamaria/">Phreato-Plinian</a>
<li><a href="http://hvo.wr.usgs.gov/volcanowatch/">Hawaiian</a>
<li><a href="http://stromboli.net/">Strombolian</a>
<li><a href="http://www.geo.mtu.edu/volcanoes/pinatubo/">Plinian</a>
</ul>
<p>
<a href="term.html">
<img src="../pictures/left.gif" alt="** " border=0>
Return to<i>Volcano Web</i></a>

<hr>
<address>
<b><a href="index.html">Volcano Web</a> :
<a href="term.html">Volcano Terminology</a> :
No Image Map</b> <p>

created by Lorrie Lava,
<a href="mailto:lava@pele.bigu.edu">
```

```
lava@pele.bigu.edu</a><br>
Volcanic Studies,
<a href="http://www.bigu.edu/">Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
<tt>URL: http://www.bigu.edu/web/nomap.html</tt>
<p>
</body>
</html>
```

2. Save this file as **nomap.html**

   **NOTE: See how we have offered the viewer access to the same information that was available from the clickable image map. A good web page does not restrict someone from content simply because they are using a different web browser.**

3. Now open up the **term.html** file in your text editor.
4. Find the line that looks like:

   ```
   <img src="../pictures/volc.jpg" usemap="#volcmap" border=0>
   ```

   and replace it with

   ```
   <a href="nomap.html">
   <img src="../pictures/volc.jpg" usemap="#volcmap" border=0>
   </a>
   ```

   **NOTE: You should be able to dissect this HTML- if the web browser can understand client-side image maps, it does so; otherwise, the whole image is hyperlinked to the nomap.html page.**

5. **Save** this file and then **Reload** it into your web browser.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. What is the difference between client-side image mapping and server-side image-mapping?
2. What does the **<map>...</map>** tag do?
3. How can you handle cases where a web browser does not understand client-side image mapping?

# More Information

For more information, we suggest you visit the Image Map Help Page or Imagemap Authoring Guide and Tutorial Sites

# Independent Practice

Identify a place in your web pages where a clickable image map would add to the interactivity or navigation of your design. Don't just throw one in for the sake of having one in there! You will have to do some work to identify the "hot" regions (you could guess if you are really desperate). Following the example in this lesson, write the HTML for the clickable image.

---

# Coming Next....

Adding **META** tags to the **HEAD** for your pages... Why? Stay tuned!

**GO TO.... | Lesson Index | previous: "More for Images and Lists" | next: "META tags" |**

---


©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

# 24. META tags

*"Hey! What's the <META> in Your <HEAD>?"*

**"A slicker opening... and I look better on Search Engines!"**

## Objectives

After this lesson you will be able to:

- Write the HTML to have one web page automatically advance to another page
- Add embedded information that will help others find your site using Internet searches

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In this lesson we will review two uses of **<META>** tags. This HTML syntax was created to allow "meta" or "extra" information to be embedded in a web page, in the part that is not displayed by the web browsers. There are many other types of META tags that you may come across, but we have provided examples on the two that are most useful.

The place that these tags go are within the **<HEAD> ... </HEAD>** part of your HTML document that we learned about way back in lesson 1.

## META REFRESH for auto advancing of web pages

The first type of **META** tag allows you to write HTML that will display your web page, wait an arbitrary number of seconds, and then *automatically* advance to another page or URL. Why would you even want to do such a thing?

- **Pages that have moved.** As a web site designer, you should strive to avoid your visitors striking the dreaded **Error 404 Site Not Found** message. Often you may create a web site available on the web, and then because you re-design the site, the file has been moved to a different folder/directory (see Swiss Web Knife), or moved to another web server (see M2C), or is just no longer made available (see Math Software Evaluations). It is nice to leave a page that offers a link to the newly named or moved site, but with the META tag, you can have it jump there automatically.
- **Timed Slide Shows** You may create a series of web pages that each sit for say 10 seconds before automatically advancing to another web page, like a kiosk (see 1996 League of Innovation Conference Web Display)
- **Opening Sequences** Many web sites now are designed to load a quick opening page, perhaps with a small logo, that pauses a few seconds before jumping automatically to a more comprehensive page. This presents a more eye-pleasing introduction, especially if the main page is "busy", like the opening credits to a movie.

We will show you how to do the third example above, adding an opening screen to the **Volcano Web** lesson. The **<META>** tag resides inside the **<HEAD>..</HEAD>** tags:

```
<head>
<title>My page title</title>
```

```
<META HTTP-EQUIV="REFRESH" CONTENT="X; URL=newpage_or_URL.html">
</head>
```

where **X**is the number of seconds that this page will be displayed before advancing to the HTML file or URL listed after **URL=**. Note that the whole string after **CONTENT=** must be in quotes, and there must be a semi-colon after the value for the number of seconds.

For this lesson we will be creating a cover page, that replaces the entry point to the lesson, **index.html**.

Before we do anything, we have to adjust the name of the page and all link references to it.

1. First, you will need to get a copy of a colorful Volcano graphic image from the Lesson 24 Image Studio. Put this image inside of your **pictures** folder/directory with your other image files.
2. Change the name of the HTML file **index.html** to **index1.html**
3. Using your text editor, open all of your HTML files, and change every link to file **index.html** to read **index1.html**. This occurs twice in every page, at the top of the page

   ```
   <a href="index.html">Index</a> /
   ```

   which should be changed to read:

   ```
   <a href="index1.html">Index</a> /
   ```

   The second place is at the bottom of every page,

   ```
   <address>
   <b><a href="index.html">Volcano Web</a> :
   ```

   which should be changed to read:

   ```
   <address>
   <b><a href="index1.html">Volcano Web</a> :
   ```

4. Now we will create a new page to replace the **index.html** page we had before. We will first construct it and verify the HTML before inserting the META tag (it is difficult to test your pages if it keeps moving every time you load it!). Using your text editor, create a new HTML file that looks like:

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
   <html>
   <head>
   <title>Volcano Web</title>
   </head>
   <BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
   <!-- first table centers all content on any browser window size -->
   <table border=0 width=100% height=100%>
   <tr>
   <td valign=middle align=center>

     <!-- Second table stretches the word "volcano!" across the width of the screen --
>
     <table border=0 width=100%>
     <tr>
     <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>v</b></font></td>
   ```

```
    <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>o</b></font></td>
    <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>l</b></font></td>
    <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>c</b></font></td>
    <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>a</b></font></td>
    <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>n</b></font></td>
    <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>o</b></font></td>
    <td width=12% align=center><font face="verdana,helvetica,arial"
size=+4><b>!</b></font></td>
    </tr>
    </table>

  <p>
  <a href="index1.html">
  <IMG SRC="../pictures/cover.gif" ALT="volcano!" WIDTH="206" HEIGHT="186"
BORDER="0"></a>
  <p>
  <font face="verdana,helvetica,arial" size=2><a href="index1.html">enter</a></font>
  </td>
  </tr>
  </table>
  </body>
  </html>
```

**NOTE: We are using some table tricks mentioned at the end of lesson 21. All of the content on this page is inside a table sized with a width and height of 100%, so it remains perfectly centered no matter what the dimensions are of the browser window. Then inside this table, we include text, pictures, and *another* table that is set to be 100% the width of the browser window, evenly stretching the 8 letters "`v o l c a n o !`" across the screen (by putting each letter in a table cell that is set to be 1/8 or 12% the width of the screen). Try it, and see what happens if you resize your web browser window!**

**We also have used `<font face=..>` styles that may not be on every person's computer; if they do not have the first one, the browser looks for the second, then the third...**

**Although we will be adding the `<META>` tag for automatically advancing the web page, it is good design to provide the hyperlinks as well; we have made the image as well as some text links to our "main" page, `index1.html`**

5. **Save** and **Reload** in your web browser. Test all of your links to make sure they lead back to the `index1.html` file.
6. Now open your newly created `index.html` file in your text editor, and after the line that reads:

```
<title>Volcano Web</title>
```

add the META tag line:

```
<META HTTP-EQUIV="REFRESH" CONTENT="3; URL=index1.html">
```

**NOTE: The META tag instructs the browser to pause 3 seconds and then automatically advance to the web page at `index1.html`**

7. **Save** and **Reload** **`index.html`** in your web browser. See if it makes the auto jump to the second web page.

# META Descriptor tags

You are doing all of this hard work to learn HTML and create web pages. Ultimately, you probably want people to be able to find them among the other 10 gizillion pages out there. **`META`** tags allow you to add extra information that will help identify your web page when people use Internet searches.

We are not going to go into great detail about web search engines (but we invite you to visit our friendly companion tutorial, "How to be a WebHound"). Basically, some clever programmers have written code that tries to follow as many web links as it can, and return a bit of information about each page it "walks" to a centralized database. They then offer a single web page where you can enter some descriptive words, and their site returns a list from their database that might match what you are looking for.

If you are not familiar, experiment with using the search features from sites such as Altavista, HotBot, Excite, Lycos, etc.

At many of these web sites, the results that are returned display the text of the first few lines of the displayed web page. Unless the very first words on your page are very descriptive, the results will not provide a clear understanding about what is on the page. For example, for our Writing HTML tutorial main page at `http://www.mcli.dist.maricopa.edu/tut/`, without META tags, a search engine might return something that looks like:

**sample web page**

Writing HTML
From: http://www.mcli.dist.maricopa.edu/tut/
Writing HTML | About | FAQ | Alumni | References | Tags | Lessons | / August, 1998 / version 3.0.4 / version history / About this Tutorial WRITING HTML WAS CREATED to help teachers create learning...

This is not too bad; we can get an idea about what this site is about, but we are seeing the text from the links at the top of the tutorial pages. However, by inserting a **`META`** tag, we can make it so that the results appear like:

**sample web page**

Writing HTML
From: http://www.mcli.dist.maricopa.edu/tut/
More than just an HTML reference, this is a structured approach for learning how to create web pages, designed by specialists in learning at the Maricopa Center for Learning & Instruction.

The HTML format for adding this kind of **`META`** tag is

```
<META name="description" content="The string of text is a really
      good description of this web page. Perhaps a second sentence
      would help as well.">
```

If you are tempted to write a long descriptive narrative, keep in mind that maximum length of a **`META`** tag, from the first **`<`** to the last **`>`** is 1024 characters, leaving 998 characters between the beginning of the tag

```
<META name="description" content="
```

and the closing

```
">
```

leaving room for perhaps 100 words.

Another **META** tag that is useful in shaping how your site fares on web searches allows you to add the important words that you think a visitor might enter when they are searching for a site such as your own. For example, for our Writing HTML tutorial main page at `http://www.mcli.dist.maricopa.edu/tut/` we provided these keywords using the format for our **META** tag:

```
<META  name="keywords" content="HTML, tutorial, learn,
   make, create, design, web page, home page, education,
   maricopa, mcli, writing, form, tables, frames,
   javascript">
```

Now we will add the two descriptor **META** tags to the main entry page for your Volcano Web site. For your own work, you may add these to each page so that it has unique content and keywords.

1. Open the **index.html** file in your text editor.
2. After the portion of the **HEAD** tag that reads:

   ```
   <META HTTP-EQUIV="REFRESH" CONTENT="2; URL=index1.html">
   ```

   add these two additional **META** tags:

   ```
   <META  name="description" content="A mini lesson about the wide world of
       volcanoes as an example for the Writing HTML tutorial">
   <META  name="keywords" content="HTML, volcano, learn, web page, earth,
        mars, Mount St, Helens, Long Valley, Pliny">
   ```

3. **Save** the HTML file and **Reload** in your Web browser.

   **NOTE: The addition of the two META tags will not make your web page *look* any different. (Why?) but once they are placed on a web server, they will enhance how your web page is "indexed" by the different web search sites.**

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. How can you make one web page automatically jump to another?
2. If you edit the **META DESCRIPTION=...** tags, why do you not see anything different when you view your web page?
3. How would you use **META REFRESH** tags to create a 10 screen, auto-advancing, slide show? How would you make it cycle endlessly through all 10 slides?
4. How can you use **META** tags to increase the chances of your web page being found by web search engines?

# Independent Practice

Can you develop a one or two sentence that would define your site to a person who had not seen it before? Add some **META** tags for descriptors and keywords to your own page.

Experiment with a front end page that introduces your web site and use the **META REFRESH** tags to make it jump to your main page. Can you make the opening page compelling enough to "invite" someone into your site? Do you think this page should contain a lot of graphics or very few? Why or why not?

# More Information

To learn more about search engines and how they work, see Search Engine Watch, and their special section on How to Use Meta Tags.

If you want to learn more about how to search techniques, try our Web Hound tutorial (it is as much or more fun than this one!)

---

# Coming Next....

Making hyper links that pop up in another browser window.

**GO TO.... | Lesson Index | previous: "Clickable Image Maps" | next: "Target That Window" |**

---

**Writing HTML:** Lesson 24: META tags
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut24.html

# 25. Target That Window

## *"Add a TARGET to your hrEF"*

or

## "How to Make HyperLinks that Do Not Leave Your Site Behind"

## Objectives

After this lesson you will be able to:

- Write the HTML to make a hyperlink open a URL or web page in a second web browser window
- Write HTML so that all links in a page load in a specified browser window
- Understand the limitations of opening multiple browser windows

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

For all of the pages we've worked on so far, we have written documents that have hyperlinks, so that clicking the hyperlink loads a new web page, replacing the content that was there. Yes, a viewer can return to a web page using the browser's **back** buttons or the **Go** / **History** features.

But perhaps there is an application where you might wish a link to open the new content in a second browser window, leaving the originating window in tact. One case where you might do that is where the first page contains a list of links to say 50 different photograph images or definitions in a glossary, It might be handier if the list of links were to remain in view so the users can click a link, view the content, and easily return to the list of choices to select another item.

Or perhaps the first page is part of an educational lesson where the students must visit 5 different web sites and answer questions about each one-- it would help them remember the assignment if the starting page remained in view.

In this lesson we will see how to modify the **`<a href=...>`** tag to specify that the link be loaded in a new browser window. In this tutorial we have been creating new windows with our web browser-- now we can see how to provide this functionality via HTML.

Before we begin, there are issues to consider. If your web pages open up too many different web browser windows, it may be confusing to the viewer which one they started with. Also, on some computers, the new browser window completely obscures other ones, so they may not even know that it is a new browser window. So use this feature when it makes the most sense for the content.

# The TARGET Attribute

The HTML code we use to specify a target browser (the window to display the content in) looks like:

```
<a href="some_url.html" target="window_name">
```

where as we know from earlier lessons **`href`** indicates the URL, either a web address or another one of our own web pages. The new part of the tag is the name we provide for **`target=`**.

We can provide any name for the target; it is an internal identifier for the web browser. What happens when you click a link like:

```
<a href="http://www.mcli.dist.maricopa.edu/tut/likethis.html" target="tutorial">
```

is that your browser says, "Hmmm. I must go fetch the HTML for the URL `http://www.mcli.dist.maricopa.edu/tut/likethis.html`, and place it in a window named **`tutorial`**. Ahhh, I do not have any such window, so I will create a new one."

Try it now.

> **NOTE: The first time you try the link, it should display a new web page in a window that sits on top of this lesson page. If you close that window, and try the test link above, it will behave the same way.**
>
> **Try it again, but this time simply shrink the new window, move it to the side, or simply activate the lesson window to the front. In many web browsers, if you click the link again, it will reload the window, but it will not bring it to the front. In fact, you can keep clicking that link until next year, and it will seem like the link is broken, when it is just hanging around in the back.**

The name you provide in the **`target=`** attribute can be almost anything. We will see later that there are 4 names that have special meaning:

1. **`_top`**
2. **`_self`**
3. **`_blank`**
4. **`_parent`**

We suggest that you use short but descriptive window names.

## Adding a Targeted Window Links to Your Pages

We will now use this new code in our volcano lesson. From our previous work, the page we created describing Volcanoes in the United States contained the small graphic of the seismograph, that when it was clicked, linked to a larger image. We will now adjust the HTML so that the large image opens in a new browser window.

1. Open the **`usa.html`** file in your text editor.
2. Find both instances where we have links to the **`seismo.jpg`** image file that read:

```
<a href="../pictures/seismo.jpg">
```

and edit them to read:

```
<a href="../pictures/seismo.jpg" target="photo">
```

3. **Save** and **Reload** in your web browser.

Now, both the links from the smaller image as well as it's hypertext caption should load the larger JPEG image in a new browser window.

We now will show you a way to set the **TARGET** attribute so that the link is forced to open in a new browser window and be in front... by using the special window target name "_blank". The disadvantage with this approach is that if you have 20 links with this window target name, it is possible to then have your single web page spawn 20 different browser windows. Because additional browser windows require more computer memory, this might be a recipe for a computer crash!

But let's look at a case were we can use this feature. In our Volcano Web site, the page for terminology contains the image of different volcano types, each hyperlinked to an external web site. Although the **href** links are written differently because they are embedded in the code we created to make the clickable image map (in lesson 23), we could modify them to read:

```
<map name="volcmap">
  <area shape="rect"

href="http://volcano.und.edu/vwdocs/frequent_questions/grp7/europe/question308.html"
     target="_blank" coords="48,46,204,153">
  <area shape="rect" href="explode.html"
     target="_blank" coords="0,66,26,227">
  <area shape="rect" href="height.html"
     target="_blank" coords="95,283,378,309">
  <area shape="rect" href="http://www.geo.mtu.edu/volcanoes/pinatubo/"
     target="_blank" coords="321,154,468,261">
  <area shape="rect" href="http://stromboli.net/"
     target="_blank" coords="172,155,318,274">
  <area shape="rect" href="http://hvo.wr.usgs.gov/volcanowatch/"
     target="_blank" coords="36,155,168,276">
  <area shape="rect" href="http://www.geo.mtu.edu/volcanoes/santamaria/"
     target="_blank" coords="90,3,343,123">
  </map>
```

If we wrote our code this way, any hyperlink from the image map would load the linked site in a new web browser window. But since we have a series of links all with the same window target, we can use a different tag to write HTML that will do the same thing:

```
  <base target="window_name">
```

which means that **all** hypertext links after this tag will be opened in the window specified. So now we will update our working files to use this new tag.

1. Open the **term.html** file in your text editor.
2. After the line that reads:

```
<font size=+2>Investigate each type by clicking on a picture</font><br>
```

add this line:

```
<font size=-1>each link will open its own browser window</font><br>
```

This provides a message for the user so they understand what will happen when they follow the links.

3. Now let's specify the window. On the line **above** the HTML code that reads:

```
<map name="volcmap">
```

insert this HTML:

```
<base target="_blank">
```

4. **Save** and **Reload** in your web browser.

> **NOTE: If you click on any of the hyperlinks from the image map, the links to the different volcano sites should open in new browser window, in front of the lesson window. Note what happens if you click the part labeled SURTSEYAN, leave the new window open, return to the first page, and repeat the click-- now you have two windows open with the same URL!**

But there is another subtle problem; below our image map we have more hypertext links to other documents and web sites that we do **not** want to load into a new window! We can take care of this situation by using another one of the special window target names, "_self" which means load the new url/page into the same browser window.

1. Open the **`term.html`** file in your text editor.
2. After the end of the image map code:

```
</map>
```

add this line:

```
<base target="_self">
```

3. **Save** and **Reload** in your web browser.

> **NOTE: This now over-rides the previous <base target=..."> tag, so that all links following this will load into the current browser window.**

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. How can you modify links to force them to open a URL in a different browser window?
2. What are some problems in opening new browser windows? What is an approach you can take to lessen the confusion for the viewer?
3. How can you make it so all links after a certain point open in a new window? How can you make this effect stop after a certain point?

# Independent Practice

In your own web page, look for places where it would make sense to open new browser windows. Experiment with adding the **TARGET=...** tag to some of your links. Does it make sense to other people? Do they understand what has happened?

---

# Coming Next....

Create useful 'frame' sets of web pages... chopping up a single page into useful pieces.

**GO TO.... | Lesson Index | previous: "META tags" | next: "Framed Web Sites" |**

---

**Writing HTML:** Lesson 25: Targeting Windows
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut25.html

# 26. Framed Web Pages

**Cop:** "Web Page, You've Been Framed!"

**Web Page:** "That's Ok! Now I can really control my content! Just keep your eye on the top of this screen as you move down with the right side scroll bar.."

## Objectives

After this lesson you will be able to:

- Create a web page that consists of multiple frames
- Write hypertext links to load content into a specified area of a framed web page
- Write hypertext links that will load content into a page replacing the framed web page
- Modify the border attributes of a framed web page

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

You likely have seen web pages that use **frames**. If you have not already figured it out, this very page uses frames-- if you scroll down through the page for this lesson, the links in the pale orange area at the top of the page stay fixed. It is in a separate frame from the bottom portion.

Frames make this web page two different HTML documents-- one document defines the layout for the top portion with the navigation links, and the other contains the remainder of this lesson. Each frame is independent of each other.

The advantage is clear for web sites that contain navigation links to many other web pages. For another example, see our Multimedia Authoring Web, a searchable database that keeps all of the navigation and control elements in the left frame while content is displayed in the right side.

Hyperlinks have special uses in framed web pages. Sometimes a hyperlink in one frame will replace the content in that frame with new content. Other times a hyperlink will load new content in another frame. And you can have links that will completely replace all of the frames with a new page. This is actually the same kind of link "targeting" we learned

in lesson 25.

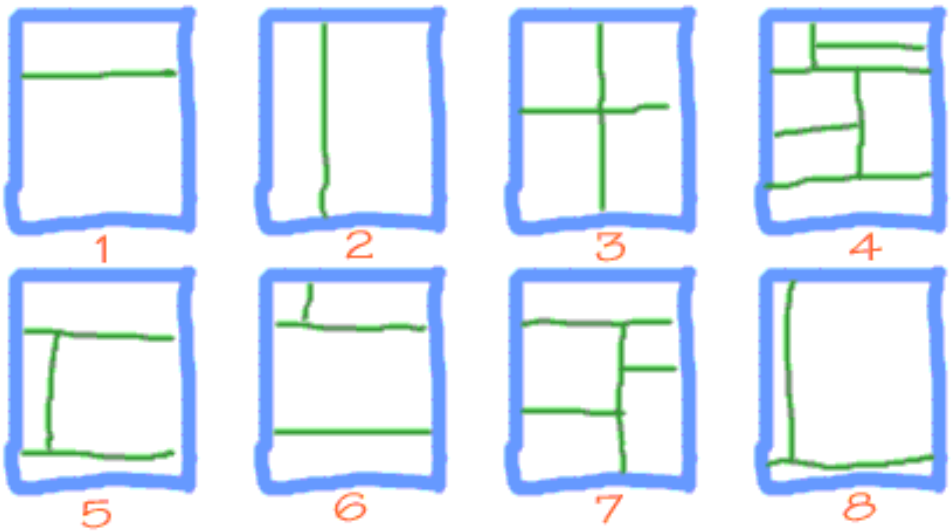
**What are some disadvantages of frames?** As a web designer, you must keep track of more HTML documents. When converted to a framed design, one single HTML file might end up as 3,4, or maybe 12 HTML files. For the viewer, a framed page can take longer to load and display. Poorly designed, framed web pages look crowded and sometimes amateurish. Frames also make it difficult to print paper copies of the entire page. Finally, you may be restricting some users from your site if they have a web browser that does not support frames (most browsers since NetScape 2.0 and Internet Explorer 3.0 display frames).

**When should you use frames?** The content should tell you. If there is a need to keep some elements on a web page visible at times while changing the content of other areas, frames can be effective. You can get a better sense by examining other web sites and see how they use frames.

## Frame Basics

A web page that uses frames consists of a "master" HTML document, that we'll call the "blueprint" for the layout, that defines the **framesets**, or the arrangement of the framed areas on the page. This is the document that loads the frame structure and the one that represents the URL for the framed page.

You can devise many ways to slice and dice the web page area:




Each of the sub-divided areas will be associated with an HTML file that defines what goes into that particular box. Therefore, the first step is to sketch out how the page should be divided up and how much relative space each area needs.

Once you have a visual idea, you must define it in terms of **rows** and **columns**, similar to the approach for designing tables (see lesson 21). Start working from the upper left to the lower right.

Looking at the sketches above, **example 1** is made of 2 rows and **example 2** contains 2 columns. **Example 3** can be seen either as two rows, each containing two columns, or 2 columns, each of which contains two rows.

Are you still with us? **Example 5** can be seen as three rows, where the middle row has two columns. So is **Example 6** except that first row is the one with two columns.

Now, look carefully at **example 7**. It is divided into 2 rows. The lower row contains 2 columns, each of which contains two rows of differing proportions.

Each collection of rows and columns makes up an HTML **frameset** and the HTML "blueprint" document for the framed page can have one or more framesets. The HTML format is slightly different from the ones we've created so far-- it lacks a set of **<body>...</body>** tags. This make sense from what we learned way back in lesson 1 since none of the frameset definitions actually specify the content that appears on the web page (normally everything inside the **<body>...</body>** tags,) but rather the structure of how they are arranged.

Below is a generalized format for a "blueprint" HTML document:

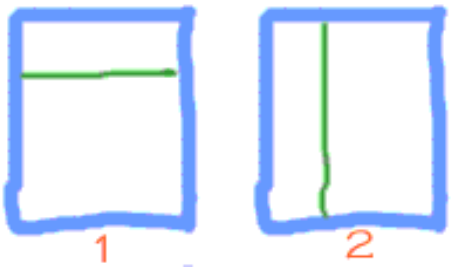
```
<!doctype html public "-//W3C//DTD HTML 3.2 Final//EN">
  <html>
  <head>
    <title>Title of this Whole Page</title>
   </head>

    <FRAMESET ROWS/COLS="X,Y,...Z">
      <FRAME SRC="frame_source1.html">
      <FRAME SRC="frame_source2.html">
          :
          :
      <FRAME SRC="frame_sourceN.html">
    </frameset>
    <NOFRAMES>
    This is what someone would see who does not have a web
    browser that can display frames.
    </NOFRAMES>
</html>
```

**NOTE: Each frameset defines either a set of rows or columns (either <FRAMESET ROWS=...> or <FRAMESET COLS=...>). The values of X, Y, and Z indicate the amount of screen area each row/column will occupy, either in percentages of the browser window size, or an absolute number of pixels. The number of items in this list defines the number of rows or columns. For each row/column specified, this dimension is associated with the HTML document specified in the subsequent list of <FRAME SRC=...> tags.**

**A browser that cannot display frames will ignore everything between the <FRAMESET> ... </frameset> tags and display what is inside of the <NOFRAMES> ... </NOFRAMES> tags. On the other hand, web browsers that can display framed content will ignore what is inside the <NOFRAMES> ... </NOFRAMES> tags.**

The most challenging part of designing framed web site is developing the layout structure of this main document. To repeat, the numbers that you provide in the **<FRAMESET...>** tag, indicate the number of and the dimensions of a set of rows or columns in the page. You can use either percentages (i.e. **ROWS=10%,30%,60%**) or absolute numbers of screen pixels (i.e. **COLS=100,300,80,200**). The choice of this depends on whether the design requires a particular frame to always be the same size (use absolute pixels) or if it can scale to the proportions of the viewer's browser window (use precentages). You will see more examples as we go along.

So let's see how some examples might look. Referring again to our example sketches,




we might write the page for **example 1** as:

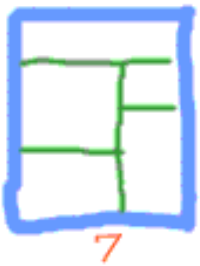
```
<!doctype html public "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
  <title>A Two Row Framed Page</title>
 </head>

  <FRAMESET ROWS="15%,85%">
    <FRAME SRC="frame_source1.html">
    <FRAME SRC="frame_source2.html">
  </frameset>
  <NOFRAMES>
  This is what someone would see who does not have a web
  browser that can display frames
  </NOFRAMES>
</html>
```

We could easily modify this document to create the layout of **example 2** by changing **ROWS=** to **COLS=**. In either of these cases, if the viewer shrinks or expands their web browser window, the areas of the frames will adjust according to the percentages provided. If you wanted to fix a frame at an absolute width, say in **example 1** to have the top row to always be 100 pixels high, you would change the tag to read:

```
<FRAMESET ROWS="100,*">
```

**NOTE: The "*" or "wildcard" value for the dimension tells the web browser to use what ever space is left for the bottom row.**

Now let's look at a more complex frame document, the one that would define **example 7**:



```
<!doctype html public "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
  <title>A More Complex Framed Page</title>
 </head>
  <!-- two main rows -->
  <FRAMESET ROWS="120,*>

    <!-- row 1 is a single doc -->
    <FRAME SRC="row1.html">

    <!-- two main columns in row 2 -->
    <FRAMESET COLS="75%,25%>
            <!-- two rows in first column -->
            <FRAMESET ROWS="60%,40%>
                    <FRAME SRC="row2col1row1.html">
                    <FRAME SRC="row2col1row2.html">
            </frameset>

            <!-- two rows in second column -->
            <FRAMESET ROWS="100,*>
                    <FRAME SRC="row2col21row1.html">
                    <FRAME SRC="row2col21row2.html">
            </frameset>
    </frameset>
  </frameset>
  <NOFRAMES>
  This is what someone would see who does not have a web
  browser that can display frames
  </NOFRAMES>
</html>
```

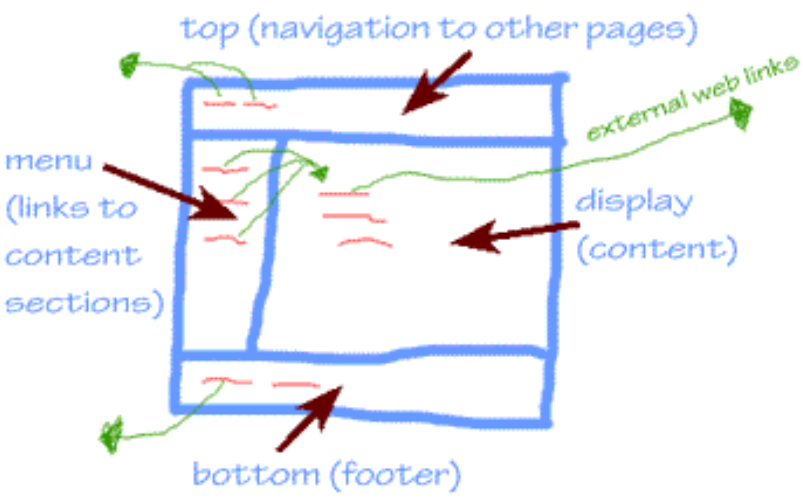
**NOTE: This web page requires 6 different HTML files to display; one blueprint document with the HTML code above plus 5 more HTML documents that make up the content as defined by the <FRAME SRC...> tags. Carefully compare the diagram with the "nesting" of the multiple <FRAMESET ... > tags.**

Yes, it gets complicated! And... you will learn even more things you can do with the HTML code for frames as you work through the examples below.

Now let's work on our Volcano Web site. The page we had created so far for the Research project page, proj.html has three different sections:

1. a description of the project;
2. a list of web site references; and
3. a bibliography

We are now going to use **frames** to convert this single page to a framed version shown in the diagram below.




The **top** frame will contain the navigation links to the other pages of our site. Links from this page must replace the entire frame page. The **bottom** frame will contain our standard page footer, and any links from this area must also replace the entire frame page.

The **left frame in the middle row** will contain hyperlinks to the three different parts of the original project page. The hyperlinks from here should load content into the frame to the right. The content for the three sections will be displayed in the **right frame in the middle row**. Any hyperlinks from this area will jump to external web sites that we will load into a new browser window.

This design allows us to break up the project page into more discrete chunks; for now they are small chunks, but perhaps you could see how this might be useful if we had much more content for each section.

Here are the steps to make the framed version of the projects page.

1. Change the name of the **`proj.html`** file to **`proj-noframes.html`**. This will be the page we will send viewers to if they have a browser that cannot display frames.
2. Create a new file in your text editor and save it as **`proj.html`**. This will be the frame layout page. Enter the following HTML into this file:

```
<!doctype html public "-//W3C//DTD HTML 3.2 Final//EN">
<html><head>
<title>Project</title></head>
<FRAMESET ROWS="45,*,150">
        <FRAME NAME="top" SRC="proj_nav.html" scrolling="no">

        <FRAMESET COLS="24%,*">
                <FRAME NAME="menu" SRC="proj_menu.html">
                <FRAME NAME="display" SRC="proj_descrip.html">
        </FRAMESET>

        <FRAME NAME="bottom" SRC="proj_footer.html" scrolling="no">
</FRAMESET>

<NOFRAMES>
<h2 align=center>NOTE: This site uses frames, but apparently your
browser does not support this feature. Try the
<a href="proj-noframes.html">alternative</a> page.</h2>
</NOFRAMES>
</html>
```

**NOTE: This frameset is set up so the top row is fixed at 45 pixels high, the bottom at 150 pixels, and the middle row will use whatever screen space remains. The `<FRAME NAME="top" SRC...>` tags in the top and bottom frames also have the attribute added for**

**`scrolling=no` which tells the browser not to add any scroll bars to the frame (these frames are small and should not need to have scrolling content). Without this attribute, the browser assumes a setting of `scrolling=auto` which means scroll bars will appear only if the content is to large to display in the frame area available. You can also force a frame to have scroll bars with a setting of `scrolling=yes`.**

**Each of the `<FRAME NAME="top" SRC...>` also contains an assigned `NAME=` attribute which we will use shortly when we set up how the frames link to each other.**

**Finally, note the `<NOFRAMES>` area that will provide the frame-disabled viewer to jump to another web page that they can use. You can make this more transparent to these users by putting inside the `<NOFRAMES>` area the HTML code (including everything between and including the `<BODY>` and the `</bODY>` tags) for the alternative page. We prefer not to do this because the number of users with browsers that do not display frames is decreasing, and this `<NOFRAMES>` code becomes extra baggage in your HTML files. Either way, you will have to make sure that if you change the content of your framed pages, you also update the frames-alternative pages. You will find many sites that do not even bother offering the alternative, but we suggest that as a courtesy you at least put a message in the `<NOFRAMES>` area.**

We have set up the frame layout, but now we must create the HTML documents that fills in the content.

3. **top frame**: Create a new HTML file in your text editor and save it as **`proj_nav.html`**. Enter the following HTML into this file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
<base target="_top">
<h5>Volcano Web /
<a href="index1.html">Index</a> /
<a href="mars.html">back</a> /
</h5>
</body>
</html>
```

**NOTE: The use of the special `"_top"` name in the `<base target=...>` tag tells the browser that any of the hypertext links that follow should load their links in a web page that replaces all of the framed content. (See lesson 25 for more on targets). Also for a frame document, we do not need to specify a `<TITLE>..</TITLE>` tag in the `<HEAD>` area (It would not cause a problem if you did , but it has no meaning here; the `<TITLE>..</TITLE>` tag in the `proj.html` file serves as the title for the entire framed page).**

4. **bottom frame**: Create a new HTML file in your text editor and save it as **`proj_footer.html`**. Enter the following HTML into this file:

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

```
<html>
<head>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
<base target="_top">
<hr>
<address><b><a href="index1.html">Volcano Web</a> : Research Project</b> <p>
created by Lorrie Lava, <a
href="mailto:lava@pele.bigu.edu">lava@pele.bigu.edu</a><br>
Volcanic Studies, <a href="http://www.bigu.edu/">Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
<tt>URL: http://www.bigu.edu/web/proj.html</tt>
<p>
</body>
</html>
```

5. **menu frame**: Create a new HTML file in your text editor and save it as **`proj_menu.html`**. Enter the following HTML into this file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
<base target="display">
<a href="proj_descrip.html">
  <font size=+2 face="arial,helvetica">
  D</font>ESCRIPTION...</a><br>
of your project.
<p>
<a href="proj_ref.html">
  <font size=+2 face="arial,helvetica">
  R</font>EFERENCES..</a><br>
web sites to research
<p>
<a href="proj_bib.html">
  <font size=+2 face="arial,helvetica">
  B</font>IBLIOGRAPHY...</a><br>
other print resources
</font>
</body>
</html>
```

**NOTE: This page provides links to the three different parts of the projects content. Note that the `<base target="display">` tag in this file will make all of three hypertext links load their HTML into the frame we have called "display".**

**Also, we have used some `<font size... face...>` tags to format the text links.**

6. **display frame (description)**: Create a new HTML file in your text editor and save it as **proj_descrip.html**. Enter the following HTML into this file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head></head>
<BODY BGCOLOR=#FFFFCC TEXT=#333333 LINK="#0000CC" VLINK=#FF6666>
<h2 align=center>Research Project</h2>
Your mission is to find information and report on a volcano, other
than the ones listed above, that has erupted in the last 100 years.
Your reports must include:
<ul type=circle>
<li>Type of volcano
<li>Geographic location
<li>Name, distance, and population of nearest major city
<li>Date of most recent eruption and date of most destructive
eruption
<li>Other events associated with the last eruption (earthquakes,
floods, mudslides, etc)
</ul>
<p>
Then, attach a one page description on the major hazards to humans in
the vicinity of this volcano. Speculate on what you would do if you
were in charge of minimizing the risk to the population.
</body>
</html>
```

> **NOTE: This page that describes the research project is the one that is loaded when the frame page is first assembled. It would also be displayed when the viewer clicks the link in the "menu" frame labelled "DESCRIPTION". This content is simply copied from our original proj.html file, but we have turned it's background to a pale yellow for the new framed page.**

7. **display frame (reference)**: Create a new HTML file in your text editor and save it as **proj_ref.html**. Enter the following HTML into this file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
</head>
<BODY BGCOLOR=#FFFFCC TEXT=#333333 LINK="#0000CC" VLINK=#FF6666>
<base target="_blank">
<h2 align=center>References</h2>
Use these references to start your research:
<p>
<table>
<tr>
<td valign=top>
```

```
<ol type=a>
<li><a href="http://www.avo.alaska.edu/">
Alaska Volcano Observatory</a>
<li><a href="http://vulcan.wr.usgs.gov/home.html">
Cascades Volcano Observatory</a>
<li><a href="http://www.dartmouth.edu/~volcano/">
The Electronic Volcano</a>
<li><a href="http://www.geo.mtu.edu/volcanoes/">
Michigan Tech Volcanoes Page</a>
<li><a href="http://volcano2.pgd.hawaii.edu/eos/">
NASA Earth Observing System (EOS) IDS Volcanology Team</a>
<li><a href="http://www.geol.ucsb.edu/~fisher/">
Volcano Information Center</a>
</ol>
</td>
<td valign=top>
<ol type=a start=7>
<li><a href="http://vulcan.wr.usgs.gov/Servers/earth_servers.html">
Volcano/Earth Science-Oriented Servers</a>
<li><a href="http://volcanoes.usgs.gov/">
US Geological Survey Volcanic Hazards Program</a>
<li><a href="http://www.nmnh.si.edu/gvp/">
Global Volcanism Program (GVP) </a>
<li><a href= "http://hvo.wr.usgs.gov/volcanowatch/">
Volcano Watch Newsletter</a>
<li><a href="http://library.advanced.org/17457/">
Volcanoes Online</a>
<li><a href="http://volcano.und.edu/">
VolcanoWorld</a>
</ol>
</td>
</tr>
</table>
</body>
</html>
```

**NOTE: This page that provides web reference links is the one that is loaded when the viewer clicks the link in the "menu" frame labelled "REFERENCES". This content is simply copied from our original `proj.html` file, but we have turned it's background to a pale yellow for the new framed page.**

**The `<base target="_blank">` tag makes all of the hyperlinks open in a new empty browser window, so links to external site will not displace our content. Without this tag, all of the hyperlinks would load content into the "display" frame.**

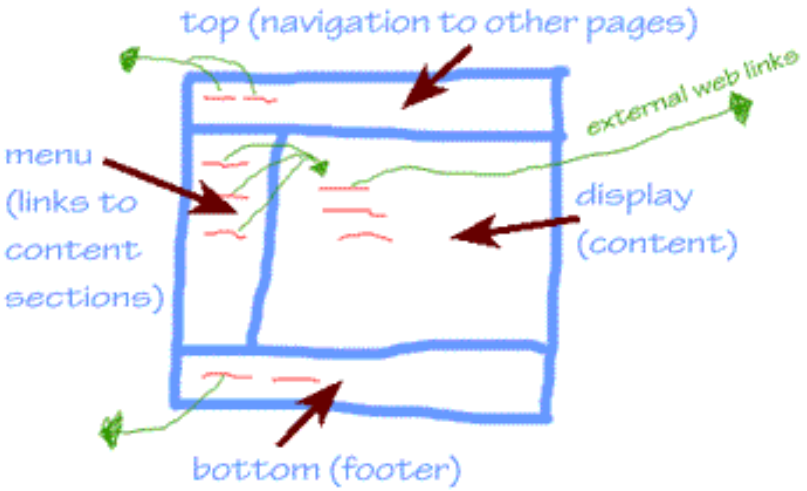
8. **display frame (bibliography)**: Create a new HTML file in your text editor and save it as **`proj_bib.html`**. Enter the following HTML into this file:

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

```
<head>
</head>
<BODY BGCOLOR=#FFFFCC TEXT=#333333 LINK="#0000CC" VLINK=#FF6666>
<h2 align=center>Bibliography</h2>
Check your library for these books:
<dl>
<dt>Cas, R.A.F. and Wright, J. V. (1987).
<dd><I>Volcanic Successions: Modern and Ancient.</I>
London: Allen & Unwin.

<dt>La Croix, A. (1904)
<dd><I>La Montagna Pel&eacute;e et ses &Eacute;ruptions.</I>
Paris: Masson

<dt>Lipman, P.W. and Mullineaux (eds). (1981)
<dd><I>The 1980 Eruptions of Mount St. Helens, Washington.</I>
U.S. Geological Survey Professional Paper 1250.
</dl>
</body>
</html>
```

By this point you should have created seven new html files that correspond to the frame design:

1. `proj.html` (defines frameset structure)
2. `proj_nav.html` (content for **top** frame, navigation links)
3. `proj_footer.html` (content for **bottom** frame, footer)
4. `proj_menu.html` (content for **menu** frame, left side links to portions of projects content)
5. `proj_descrip.html` (content for **display** frame, project description content)
6. `proj_ref.html` (content for **display** frame, project reference content)
7. `proj_bib.html` (content for **display** frame, project bibliography content)




9. Now it is time to test all of the pages we have just made. **Load** the **`proj.html`** file in your web browser. If all went well, it should look like this example. If the web screen is blank, then there is probably a mistake in the page that defines the frame structure, **`proj.html`**. If only some of the frames load, then the file names in the **`proj.html`** file may not match the files that you have created.

   Also test the links in your framed page. All hyperlinks from the "top" and "bottom" frames should replace the framed content. Links in the left side "menu" frame should load content to the right side "display" frame.

## A Few More Frame Options

We will introduce a few more things you can do with frames. But first, keep in mind that the implementation of many

frame options will not be the same on NetScape and Microsoft browsers (thanks for sticking to "standards"!)-- so we will focus on the ones that we have found to be most browser brand independent.

You may have noticed that your web browser places gray frame dividers between the frames. You can set the width of these dividers and even their color.

1. Open the **proj.html** file in your text editor and edit the first **<FRAMESET...>** tag to read:

   ```
   <FRAMESET ROWS="45,*,150" BORDER=10 frameborder="1" BORDERCOLOR="#66CCFF">
   ```

   Setting the **frameborder** parameter to 1 turns on the features for modifying the border features. The **BORDER=...** option specifies the width of the frame dividers in pixels. And the **BORDERCOLOR=** attribute lets you choose a color for the frame dividers (see lesson 16 for more on the color codes).

   **NOTE: In our experience the first frameset tag that sets these options controls the following framesets. Other HTML references suggest that each frameset tag can specify different divider widths and colors, but we've not been able to achieve that result.**

2. **Save** and **Reload** in your web browser. Compare it to our colored border example.
3. In our opinion, framed pages with thick (or thin) dividers, look rather "clunky", like a patchwork of squares. We prefer a less obtuse look, where we display the content with no visible borders. We will modify the first **<FRAMESET...>** tag in our **proj.html** file as follows:

   ```
   <FRAMESET ROWS="45,*,150"  BORDER=0 frameborder="0">
   ```

   **NOTE: These tags may look redundant, but it is what seems to make most browsers behave the same way.**

4. **Save** and **Reload** in your web browser. Compare it to our invisible frame border example.

Frames are a place where all browsers do not behave the same. NetScape browsers will "pad" the margins of frames with about 4 pixels of blank space, while Internet Explorer places no space padding around the frames. This means viewed in Internet Explorer, the yellow colored "display" frames will have text that almost touches the black "menu" frame to the left. This is hard to read.

Several HTML references list HTML options for setting this margin space (**MARGINWIDTH=X** and **MARGNHEIGHT=Z** in the **<FRAME...>** tag), but we've been unable to obtain consistent results on enough browsers. But as clever HTML coders, we can find a way to insert a "shim" of blank space between the two frames in the middle row.

1. Create a new HTML file called **yellow.html** and enter in this file:

   ```
   <html><head></head><BODY BGCOLOR=#FFFFCC></body></html>
   ```

   Yes, this is a pretty puny HTML document! It is just a blank yellow page that matches the yellow background of the "display" frame.
2. Edit the **proj.html** file by changing the lines that read:

```
<FRAMESET COLS="24%,*">
        <FRAME NAME="menu" SRC="proj_menu.html">
        <FRAME NAME="display" SRC="proj_descrip.html">
</frameset>
```

to read:

```
<FRAMESET COLS="24%,10,*">
        <FRAME NAME="menu" SRC="proj_menu.html">
        <FRAME NAME="spacer" SRC="yellow.html">
        <FRAME NAME="display" SRC="proj_descrip.html">
</frameset>
```

We have inserted a middle column of 10 pixels width in the **<FRAMESET COLS=...>** tag that points to the blank yellow page we created in the previous step. (See how you can mix frame dimensions that are percentages, pixels, and wildcards!)

3. **Save** and **Reload** in your web browser.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. What are the purposes of the **<FRAMESET..>** and **<FRAME...>** tags?
2. How many HTML documents would it take to make a framed web page that had 3 rows, each with 2 columns?
3. How do you make a hyperlink from one frame load information into another frame?
4. How do you make a hyperlink from one frame go to a URl that replaces the complete framed structure?
5. How do you make red thick dividers between frames? How do you make invisible dividers?

# Independent Practice

Set up the frameset pages for examples 4 and 8 from the lesson above.

Review your own web pages and try to identify whether frames would offer you any advantages.

# More Information

If your framed page has visible dividers/borders, the user can click and drag that border with the mouse to resize the frame dimensions. Try this on the blue frame border at the top of this page. If you would like to prevent this from occurring, add the attribute NORESIZE to any **<FRAME ...>** tag:

```
<FRAME NAME="myfixedframe" border=8 frameborder="1" SRC="fixed.html" NORESIZE>
```

---

# Coming Next....

Adding interactivity with JavaScript.

**GO TO.... | Lesson Index | previous: "Target That Windows" | next: "JavaScript (some)" |**

---

**Writing HTML:** Lesson 26: Frames
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut26.html

# 27. A Wee Dose of JavaScript

## Just a spoonful of JavaScript...

## It won't hurt...

## It may make your web feel better...

## Be careful! Don't gulp it!

# Objectives

After this lesson you will be able to:

- Describe the difference between **Java** and **JavaScript**
- Understand the relationship of JavaScript to other elements of an HTML document
- Write the general code for a set of JavaScript instructions
- Write a command for a JavaScript "alert" message
- Describe the JavaScript object model

---

# Lesson

**JavaScript** offers you, the web page designer, a way to further enhance the design and functionality of your web pages. Unfortunately a complete tutorial on JavaScript is beyond the scope of this tutorial (but see some of the other references we recommend), but what we will do is give you a taste of what JavaScript can do.

Please be aware of the tremendous differences between **Java**, a programming langauge, and **JavaScript**, a scripting language. Too often people use them interchangably. Java was developed by Sun

Microsystems as a computer platform independent programming language for creating small applications, or "applets" that could be part of a web page as well as being a stand alone desktop program. Java applets are like small, self-contained programs, that you can use without seeing or even caring about what is inside of them (we will in a later lesson show you how to find and use Java applets).

The downside of Java is that to create your own applets, you must learn a pretty complex programming language or try to use some of the newer software tools that make the coding less difficult. The other downside is that many Java applets can take a disturbing amount of time to download and run, essentially stalling your web page until it "loads" (in our opinion, waiting to load a scrolling banner is not too many notches above the **<blink>** tag!).

Originally named "LiveScript", **JavaScript** was developed by NetScape as something quite different. It was renamed because of structural similarities to the Java programming language. To create JavaScript you simply type its commands interspersed with your HTML and the browser follows the commands as it tries to format the web page.

Hopefully by now you have a sense that as a web browser reads in the HTML code for a page, it starts assembling and displaying the page layout from the top down, so that as a page may partially display even as the browser is reading in the later parts of the HTML document. As each HTML code is read in the browser reacts and formats, without asking any questions.

When the web browser encounters some JavaScript code, it starts interpreting it line by line. But the JavaScript code can instruct the browser to do different things under different situations, or build in some functionality that is not set in motion until the user does something on a page. JavaScript can even create HTML content on the fly, so you can have it do something like print a different HTML message depending on what day of the week it is or changing the background to a random color every time you reload the web page.

So think about JavaScript as a way to add a little bit more brains to your web page. While not as difficult to learn as a pure programming langauge like Java, to use JavaScript is to take a step in complexity up from HTML formatting.

In this lesson we will learn a few small doses of JavaScript that you can use right away in web pages. In another lesson, we will see how it can be used to process your web page forms.

# Putting JavaScript in its Place

Where Javascript code goes depends on what it needs to do. As we will see, sometimes we will place JavaScript code inside the **<HEAD>..</HEAD>** of your HTML file. Other times it sits inside your HTML content. And in other instances, it is added to other HTML tags to initiate "events" triggered by the person interacting with your web page.

The basic approach for writing most "doses" of JavaScript reads like:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- hide from JavaScript impaired browsers

// This is a JavaScript comment. It is not interpreted

  JavaScript statement1;
  JavaScript statement2;
  JavaScript statement3;

// done hiding -->
</SCRIPT>

<NOSCRIPT>
  Content for browsers that cannot deal with
  JavaScript
</NOSCRIPT>
```

This is the most reliable way we have found to set up JavaScript so that it works well in most environments. All of the functionality is defined by "statements" between the **<SCRIPT>...</SCRIPT>** tags. The lines shown in red that are inside of these tags protect the code from displaying if the viewer's browser cannot understand JavaScript. Remember that if a browser does not know what the tag **<SCRIPT>** means, it will ignore it. The lines in red:

```
<!-- hide from JavaScript impaired browsers
   :
   :
// done hiding -->
```

enclose the JavaScript statements inside an HTML comment tag so they will not be displayed. The browser would march on, ignoring the **<NOSCRIPT>** and **</NOSCRIPT>** tags (again, ignorance is bliss) but it will display the content in between these last two tags.

Now if the browser knows JavaScript, it begins interpreting the code line by line. In JavaScript, lines that begin with either **<!--** or **//** are interpreted as comments, and ignored. The browser looks at all of the other statements, which are step by step instructions, and does as it is told to do.

> **NOTE: Each JavaScript statement must end with a semi-colon (;) which is how the browser knows it is time to do whatever that line told it to do.**

# JavaScript says Hello

Probably the simplest JavaScript command is one that displays an **alert** message-- text that appears in a "dialog" box in the middle of the screen that typically causes the computer to beep and requires you to click an **OK** button to return to what you were doing. For example, see what happens when you click the button below:

Tell me something

The command to make this happen looks like:

```
alert('JavaScript here, boss! What do you want?');
```

Now if we simply inserted this into our HTML file like:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- hide from JavaScript impaired browsers
   alert('JavaScript here, boss! What do you want?');
// done hiding -->
 </SCRIPT>
```

The alert message would pop up immediately as the browser read the code, likely not what we want. To see this in action, try this test page.

More typically, as with the button above, you want this JavaScript command to happen **when it is triggered by an "event"** such as the viewer clicking the mouse or even moving the mouse over different parts of the screen. We'll learn more about events as we go.

# Objects

Now we will throw some more programming jargon at you! Don't recoil in horror!

JavaScript references an **object model** for the web environment. What does this mean? Think of it as a family tree structure that as you read it moves from left to right but really references things structure from most global to more specific.

Huh?

The big "parent" is the "navigator"- it has many different "properties" that describe more or less the web browser you are using, i.e. what kind (NetScape, Microsoft, etc) and version number. Below is the

"window" object that describes the properties of one web window, with its own special properties. Next down is the "document" object, that describes information about a particular web page, say its URL, the time it was last changed, how many links there are in the page. And within the document object are many more objects that we will see soon.

So in JavaScript we often have to refer to things by where they are in the object family tree, like:

```
window.document.form[3].choices.options[2]
```

which would refer to some property of "options" contained within something else called "choices", which is part of one "form" inside the document of a window. So from left to right this object model goes from biggest to smallest objects, each one separated by a period. The things in square brackets ([x]) represent **arrays**, or collections of similar objects, so that the example above, the document has at least 4 forms since we are referring to the 3rd one (it gets confusing because JavaScript starts counting many objects starting with 0 rather than 1!).

Sometimes we can use this type of structure to "test" or get some value from our web page environment, known as **properties**. Other times, we use this structure to change these values or properties.

# Caution Flag

As we continue with these advanced lessons, your task becomes more difficult. Some web browsers, especially older ones, do not support JavaScript, but we have chosen examples of code that should work on a wide range of environments.

Also, JavaScript code is pretty picky! When you are copying the code examples, it is very important to **not** have extra return characters within a single line of JavaScript code. You will see what we mean in the next lesson.

# Check Your Work

This is an introduction, so we have not made any changes to our project.

# Review

Review topics for this lesson:

1. What are some differences between Java and JavaScript?
2. How are JavaScript statements hidden inside HTML for browsers that cannot understand

JavaScript?
3. What are two ways of writing comments inside JavaScript commands?
4. How do you write a JavaScript command to display an "alert" message?
5. What is the importance of the order in which JavaScript objects are written left to right?

# Independent Practice

See if you can insert the JavaScript **`alert`** command into different parts of your HTML document. What happens if you place two different ones in different parts of your document?

---

# Coming Next....

Doctor JavaScript prescribes your first dose... alert messages that are triggered by mouse clicks plus mouse rollover displays.

**GO TO.... | Lesson Index | previous: "Framed Web Sites" | next: "JavaScript: Alerts and MouseOvers" |**

---

**Writing HTML:** Lesson 27: A Wee Dose of JavaScript
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut27.html

# 27a. JavaScript : Alerts and Rollovers

## You've tasted it...

## let's make alert messages...

## generate mouseover messages...

## good!

## Objectives

After this lesson you will be able to:

- Write the JavaScript code to generate an alert message when a link is clicked
- Write the JavaScript code to display messages in the status bar when the mouse is moved over a link.

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

Our first dose of JavaScript is to write a command inside a hyperlink tag to do something other than jump to a URL when it is clicked. This allows our links to do more than just transfer us to another document:

```
<a href="alt.html" onClick="JavaScript Statement; return false">link text</a>
```

> **NOTE: The listed JavaScript statement will be processed when the viewer clicks the hypertext link. The `onClick="..."` event is triggered by a mouse click. Note the "C" must be capitalized!**
>
> **The addition of the JavaScript statement `return false` tells the web browser to ignore the URL in the `href=` part of this tag. However, if the person viewing this page does not have a JavaScript enabled web browser or has JavaScript disabled, the `onClick="..."` code is ignored and the viewer is transferred to the URL `alt.html`. This enables us to support a wide range of audiences by providing an alternative HTML page to the JavaScript. if you want to verify that this works, check your web browser's Preference/Options settings and look for a checkbox where you can de-activate JavaScript (do not forget to turn it back on when you are done!). This is well worth doing as you add more JavaScript features to your pages.**

The first thing we will do is to modify our terminology page of your *Volcano Web* project, so that the list of words in the first paragraph display an **`alert`** message defining their meaning.

1. Open the **`term.html`** file in your text editor.

2. Modify the first **<ul>..</ul>** list to read:

```
<li><a href="term_1.html" onClick="alert('A caldera is a circular shaped
landform depression caused by the eruption of a large, near surface
body of magma.'); return false">caldera</a>
<li><a href="term_2.html" onClick="alert('Vesicularity is a measure how much
of a rock volume consists of air chambers.'); return false">vesicularity</a>
<li><a href="term_3.html" onClick="alert('Pahoehoe is a type of basaltic lava
flow texture that comes from the Hawaiian word for smooth and
ropy.'); return false">pahoehoe</a>
<li><a href="term_4.html" onClick="alert('Rheology is the study of how
materials deform.'); return false">rheology</a>
<li><a href="term_5.html" onClick='alert("A lahar is a mudslide generated
from the flanks of a volcano. Some say it comes from the
phrase \"Look Out!\" in the Indonesian language."); return false'>lahar</a>
```

**NOTES: We have created a JavaScript command that over-rides the hyperlink tag with an instruction to display a string of text in an "alert" box on the screen.**

**The JavaScript code will generate error messages if there are any RETURN characters inside the command! So each <li> line must have only one RETURN at the end of the line. If you would like to copy and paste this code, use this sample of HTML that will load in a new browser window.**

**If there are errors in your JavaScript code, the browser will generate error messages as it tries to interpret the code. To see how this works take the JavaScript Error test.**

Look carefully at the way quote characters are used in the **onClick** part of the tag. The entire JavaScript command must be enclosed in a set of quotes. And the **alert** command itself must include another string that is in quotes. We use single quotes and double quotes for this purpose, and it does not matter which one we use:

```
<a href="#" onClick="alert('You are the boss!')">tell me something</a>
```

will act exactly like:

```
<a href="#" onClick='alert("You are the boss!")'>tell me something</a>
```

Why bother? Let's say that the message you want to display in the JavaScript triggered alert needs to contain a quote as a character. JavaScript needs to know that the quote means just a quote and not part of the JavaScript instruction. We can do this by putting the forward slash (\) in front of the quote character' known in techie terms as "escaping" the character.

In our example above, we want the words **Look Out!** in the last list item to have quotes around them. If we want these to be single quotes, we would use:

```
<li><a href="#" onClick="alert('A lahar
is a mudslide generated from the flanks of a volcano. Some
say it comes from the phrase \'Look Out!\' in the Indonesian
language.')">lahar</a>
```

but since we wanted double quotes, we swapped our uses of single and double quotes like:

```
<li><a href="#" onClick='alert("A lahar
is a mudslide generated from the flanks of a volcano. Some
say it comes from the phrase \"Look Out!\" in
```

```
the Indonesian language.")'>lahar</a>
```

Finally, we need to create 5 new (but small) HTML pages that will be used if the person viewing our page cannot use the JavaScript command.

1. Open your text editor and create a new HTML document named **term_1.html**
2. Insert the following HTML into this new file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>Volcano Terminology</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFF00 LINK=#33CCFF VLINK=#FF6666>
<h2 align=center>
A <font color="#FF0000">caldera</font> is a circular shaped
landform depression caused by the eruption of a large, near
surface body of magma.</h2>
<center>
<a href="term.html">return</a>
</center>
</body>
</html>
```

3. Open your text editor and create a new HTML document named **term_2.html**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>Volcano Terminology</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFF00 LINK=#33CCFF VLINK=#FF6666>
<h2 align=center>
<font color="#FF0000">Vesicularity</font> is a measure how
much of a rock volume consists of air chambers.
</h2>
<center>
<a href="term.html">return</a>
</center>
</body>
</html>
```

4. Open your text editor and create a new HTML document named **term_3.html**
5. Insert the following HTML into this new file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>Volcano Terminology</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFF00 LINK=#33CCFF VLINK=#FF6666>
<h2 align=center>
<font color="#FF0000">Pahoehoe</font> is a type of basaltic
lava flow texture that comes from the Hawaiian word for
```

```
smooth and ropy.
</h2>
<center>
<a href="term.html">return</a>
</center>
</body>
</html>
```

6. Open your text editor and create a new HTML document named **term_4.html**
7. Insert the following HTML into this new file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>Volcano Terminology</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFF00 LINK=#33CCFF VLINK=#FF6666>
<h2 align=center>
<font color="#FF0000">Rheology</font> is the study of how
materials deform.
</h2>
<center>
<a href="term.html">return</a>
</center>
</body>
</html>
```

8. Open your text editor and create a new HTML document named **term_5.html**
9. Insert the following HTML into this new file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>Volcano Terminology</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFF00 LINK=#33CCFF VLINK=#FF6666>
<h2 align=center>
A <font color="#FF0000">lahar</font> is a mudslide generated
from the flanks of a volcano. Some say it comes from the phrase "Look Out!"
in the Indonesian language.
</h2>
<center>
<a href="term.html">return</a>
</center>
</body>
</html>
```

10. Save all of these HTML files and Reload **term.html** in your web browser. Compare to this example of how the terminology page should appear at this point.

## Some MouseOver Action!

So far we have learned how to use JavaScript to do something when the viewer clicks a hypertext link. We can add another feature to our hyperlinks to do something just when the viewer moves their mouse over the linking text (without clicking the link).

The normal behavior for this action is that the web browser displays in its **status bar** (the area usually at the bottom left corner of the browser window) the URL that the link leads to, like what you see if you move your mouse over, but do not click on, this link to our tutorial.

With JavaScript, we can create a custom message that is displayed instead of the link's URL, like this second link to our tutorial. Clicking either of this links will do the expected; transfer you to the web site specified in the hyperlink's URL.

The JavaScript format for adding this feature is:

```
<a href="xxxx.html"
    onMouseOver="window.status='text of custom message';
    return true">linking text</a>
```

The JavaScript **event** that triggers the custom message is **`onMouseOver="..."`** meaning the hypertext link says "when the mouse is over me, do this". There are two different JavaScript function statements here, separated by a semi-colon (;).

1. `window.status='text of custom message';`

   This is the JavaScript command that says, "Display in the status bar everything between the quotes".

2. `return true`

   This is a "message" that must be sent back to the browser to let it know we are done. We cannot explain why exactly this is needed, but we can tell you that it will not work unless you include this!

Now, we will add a mouseOver message to the links we modified for the terminology web page that produce the alert messages.

1. Open the **`term.html`** file in your text editor.
2. Modify the first **`<ul>..</ul>`** list to read:

```
<ul>
<li><a href="term_1.html"
  onClick="alert('A caldera is a circular shaped landform
  depression caused by the eruption of a large, near surface
  body of magma.'); return false"
  onMouseOver="window.status='what is a caldera?';
  return true">caldera</a>
<li><a href="term_2.html"
  onClick="alert('Vesicularity is a measure how much of a
  rock volume consists of air chambers.'); return false"
  onMouseOver="window.status='what is vesicularity?';
  return true">vesicularity</a>
<li><a href="term_3.html"
  onClick="alert('Pahoehoe is a type of basaltic lava
  flow texture that comes from the Hawaiian word for
  smooth and ropy.'); return false"
  onMouseOver="window.status='what is pahoehoe?';
  return true">pahoehoe</a>
<li><a href="term_4.html"
  onClick="alert('Rheology is the study of how materials
  deform.'); return false"
```

```
  onMouseOver="window.status='what is rheology?';
   return true">rheology</a>
<li><a href="term_5.html"
  onClick='alert("A lahar is a mudslide generated from
  the flanks of a volcano. Some say it comes from the
  phrase \"Look Out!\" in the Indonesian language.");
  return false'
  onMouseOver="window.status='what is a lahar?';
  return true">lahar</a>
</ul>
```

**NOTE: If you would like to copy and paste this code, use this sample of HTML that will load in a new browser window.**

3. While we are adding this feature, it would help add MouseOver messages to the "hot area" of the clickable image map we created for this page in lesson 23. So modify the HTML between the **<map>...</map>** tags to read:

```
<map name="volcmap">
<area shape="rect"

href="http://volcano.und.edu/vwdocs/frequent_questions/grp7/europe/question308.html"
  coords="48,46,204,153"
  onMouseOver="window.status='information about surtseyan type volcanos';
  return true">
<area shape="rect"
  href="explode.html"
  coords="0,66,26,227"
  onMouseOver="window.status='description of explosiveness scale';
  return true">
<area shape="rect"
  href="height.html"
  coords="95,283,378,309"
  onMouseOver="window.status='description of height scale';
  return true">
<area shape="rect"
  href="http://www.geo.mtu.edu/volcanoes/pinatubo/"
  coords="321,154,468,261"
  onMouseOver="window.status='information about plinian type volcanos';
  return true">
<area shape="rect"
  href="http://stromboli.net/"
  coords="172,155,318,274"
  onMouseOver="window.status='information about strombolian type volcanos';
  return true">
<area shape="rect"
  href="http://hvo.wr.usgs.gov/volcanowatch/"
  coords="36,155,168,276"
  onMouseOver="window.status='information about hawaiian type volcanos';
  return true">
<area shape="rect"
  href="http://www.geo.mtu.edu/volcanoes/santamaria/"
  coords="90,3,343,123"
  onMouseOver="window.status='information about phreato-plinian type volcanos';
  return true">
</map>
```

**NOTE: If you would like to copy and paste this code, use this sample of HTML that will load in a new browser window.**

If your JavaScript is working, as you move the mouse over the image on this page, the status bar will display the custom message that describe the link, rather than the URL.

**NOTE: As of April 1999, this technique of using `onMouseOver` inside the `<map>...</map>` tags does not work on Microsoft Internet Explorer 5.0 web browsers.**

Using the **`mouseOver`** technique can be an effective feature of your web site, but keep in mind that sometimes it is more useful for the person looking at your site to know the URL where the link leads to (perhaps so they can guess the organization that owns the web site from its URL)-- the **`onMouseOver`** message hides this information. Use it where it makes sense for your web site design and what you can determine are the needs of the visitors to your site.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Note that JavaScript is **very** sensitive to typographical mistakes -- one missing quote or semi-colon can ruin the page!

# Review

Review topics for this lesson:

1. How can you make a hypertext link display an alert message rather than connecting to another web page?
2. How do you write a JavaScript **`alert`** command with text that includes quote characters?
3. What is a **`mouseOver`** event?
4. How do you generate messages in the status bar when the viewer of a web page moves the mouse over different links?

# Independent Practice

Add some JavaScript **`alert`** and **`mouseOver`** code to some of the hyperlinks in your own web pages.

Extra Credit: Can you think of a way that a mouseOver can generate an alert message?

---

# Coming Next....

Get a second dose of web medicine from your JavaScript Pharmacy... JavaScript that creates custom content and dynamic footers.

**GO TO.... | Lesson Index | previous: "JavaScript: Intro" | next: "JavaScript: Dynamic Content" |**

---

**Writing HTML:** Lesson 27a: A Wee Dose of JavaScript : Alerts and Rollovers
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut27a.html

# 27b. JavaScript : Dynamic Content

## Let's Increase our Dosage...

## write dynamic content to a page...

## different messages for different conditions...

# Objectives

After this lesson you will be able to:

- Write the JavaScript code to dynamically write any HTML code
- Write the JavaScript code to display automatically the web page title, the URL and file modification dates
- Write the JavaScript code to print a formatted date
- Write the JavaScript code to determine the current web browser brand and version

---

# Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

We have reminded you before that as an HTML document loads into a web browser, it is interpreted as the browser reads in more data. For HTML, this means that as soon as the browser gets enough information to display something, it tries to do that, even as it reads in the remainder of the document.

When we refer to **dynamic** content that you can write in JavaScript, that means that the web browser can do more than just "read and display", "read and display"... as it reads in JavaScript code among your HTML, it can make decisions based upon the way the code is set up, test against some built in functions, and then write different HTML code according to its programmed instructions. It may even be programmed to do something randomly different every time the page loads.

It makes your web pages a bit more "intelligent" than just sitting there looking pretty!

We can use JavaScript to write any other kind of HTML content using the format:

```
document.write('xxxxxxxx xxxxxxxxxx');
document.write('xxxx xxxxxxx xxxxxxx xxxxx');
document.write('xx xx xxxxxxxxxxxx xx');
```

Each line of this code writes a chunk of HTML, that is everything inside of the single quote characters. This series of JavaScript statements:

```
document.write('<h1>Wide World of Cheese</h1>')
document.write('<b><i>Not everyone in this world likes cheese');
document.write('as much as I!<i></b><br> -- Colby Jack (1903)');
```

will produce the exact same display as this chunk of HTML:

```
<h1>Wide World of Cheese</h1>
<b><i>Not everyone in this world likes cheese
as much as I!<i></b><br> -- Colby Jack (1903)
```

Now if this was all there was to JavaScript, we would not even be trying to teach it to you! JavaScript provides other types of information that you can display that is not available via HTML. There are built in functions that can provide the current date and time, information about the user's web browser, and more as we will soon see. But even more than this, we can set up logical statements to test, so that we can do things like:

> If today is Tuesday or Wednesday and the time is after 12:00 PM then display this custom message morning greeting; but if it is before 12:00 PM, then display this different message for the afternoon. But if it is Friday, at any time, let's display a different type of message no matter what time of day it is. Any other day or time, just display a standard "Have a nice day" message.

JavaScript code gives us some flexibility to create **dynamic** content that can behave and display differently to the viewer.

The first thing we will do is to write a series of JavaScript statements that will create the footers of all of our documents-- but unlike the ones to date that we have written by hand, this same block of HTML/JavaScript can be cut and pasted into every document but will also dynamically print (with examples for this page shown in parentheses):

a. The title of the web page, from the text in the <TITLE>..</TITLE> tags
   (undefined)
b. An e-mail link that by default will insert the web page title as the subject line
   (email test with this page title as subject)
c. The date that the document was most recently modified, if valid
   (this feature not available from this host)
d. The actual URL of the file
   (

Not all web servers provide accurate document modification dates, item (c), and typically when you are testing documents on your desktop computer, you will not be able to get this information and it would print some erroneous date like January 1900. We will show you how to test for these conditions. Item (d) is extremely useful because if we were to move our web pages to different web servers or re-arrange our site, the URL would be updated automatically.

The template for our "standard" web page footer for the Volcano Web project looks like the code below. We'll present it first section by section with some explanations. It's not critical that you understand exactly how it works, since when you incorporate it into your working pages, it will be simply a cut and paste operation.

## JavaScript Footers

| HTML code | explanation |
|---|---|
| `<!-- start of Volcano Web standard footers -->`<br>`<SCRIPT LANGUAGE="JavaScript">`<br>`<!-- hide scripts from old browsers` | Marks the beginning of the footer area, with proper JavaScript tags |

```
document.write('<p><hr>
<FONT FACE="helvetica,arial" size=-1>
<i>Volcano Web:</i> <b>');
```

Horizontal rule, begin text font size and main web site title

```
document.write(document.title + '</b><BR>');
```

Write the current document title

```
document.write('created by Lorrie Lava, ');
document.write('<a href="mailto:lava@pele.bigu.edu?subject='
+ document.title + '">');
document.write('lava@pele.bigu.edu</a><br>');
```

Credits for page with an e-mail link that automatically inserts the page title into the subject line.

```
document.write('Volcanic Studies,
<a href="http://www.bigu.edu/">');
document.write('Big University</a><p>');</tt></font>
```

More static content.

```
// append a modification date only if
// server provides a valid date
if (Date.parse(document.lastModified) > 0) {
  document.write('<b>last modified: </b>'
  + document.lastModified + '<BR>');
}
```

This looks tricky. We have set up a **conditional** test on the value that is returned by **`document.lastModified`**, and if it is valid (a non zero return from a **function** we use called **`Date.parse`**, we write the modification date. If we do not get a valid date, this whole block is ignored.

```
document.write('<b>URL: </b>'
  + document.location.href + '</FONT><P>');
```

Write the URL of the current document given by the value returned by the JavaScript variable **`document.location.href`**

```
// done hiding from old browsers -->
</SCRIPT>
<!-- end of Volcano Web standard footers -->
```

End of the footer code block.

Most of the dynamic content in this examples comes from "properties" of different JavaScript "objects", namely in this case the **document** object. Each HTML document has built-in identifying pieces of information-- namely it's title, date of last modification, URL, etc. We can query the **document** object to get and then use this information.

Information we are writing to the page using **document.write** that is fixed content (strings of text in quotes) and information that is stored in JavaScript variables, like **document.title**, we join with the "+" sign:

```
'Here is some fixed <b>HTML</b> text for the page:' + document.title + '! Nice, eh?'
```

The "+" sign joins the text together into one longer string, so if we were using it in this lesson page, we would see:

Here is some fixed **HTML** text for the page: 27b. JavaScript- Dynamic Content! Nice, eh?

Our JavaScript footer also uses a "conditional" test ("if this then do that") for the modification date, that looks like:

```
if ( some condition to test ) {
  JavaScript statement1;
  JavaScript statement2;
        :
        :
  JavaScript statementN;
}
```

which means that if the condition test in the first line's parenthesis results in a TRUE value, we would follow the steps inside the "{" and "}" brackets; if the test is FALSE, we skip these statements. With the power of JavaScript, we can construct very complex conditional tests, but for now we will keep it simple.

This is how our JavaScript footer would look if it were used in this lesson page:

**sample web page**

*Volcano Web:* **undefined**
created by Lorrie Lava, lava@pele.bigu.edu
Volcanic Studies, Big University

Now we will insert the footer into our *Volcano Web* HTML documents.

1. First, open your main entry page, **index1.html** in your HTML editor. (we will not put the footer on our opening "splash" page, **index.html**).
2. Delete our old footer HTML content:

```
<hr>
<address><b>Volcano Web</b><br>
created by Lorrie Lava, <a href="mailto:lava@pele.bigu.edu">lava@pele.bigu.edu</a><p>
Volcanic Studies, <a href="http://www.bigu.edu/">Big University</a><p>
<tt>last modified: April 1, 1995</tt>
</address>
<p>
<tt>URL: http://www.bigu.edu/web/index.html</tt>
<p>
```

3. Replace it with our new JavaScript footer content:

```
<!-- start of Volcano Web standard footers -->
<SCRIPT LANGUAGE="JavaScript">
<!-- hide  scripts from old browsers

document.write('<p><hr><FONT FACE="helvetica,arial" size=-1><i>Volcano Web:</i>
<b>');
document.write(document.title + '</b><BR>');
document.write('created by Lorrie Lava, ');
document.write('<a href="mailto:lava@pele.bigu.edu?subject='
```

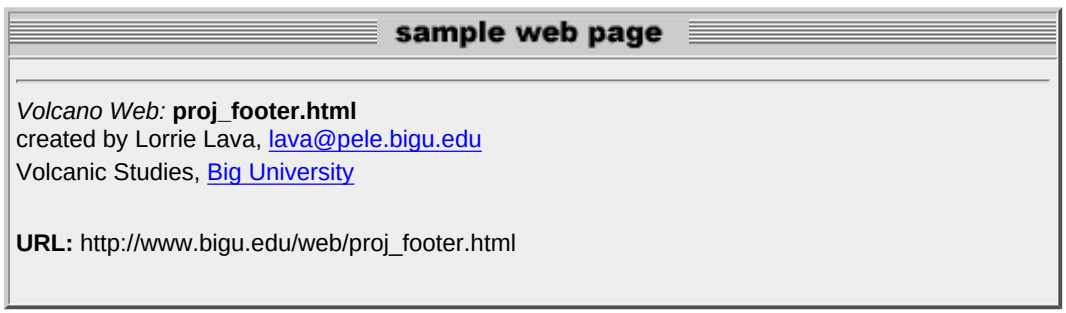
```
  + document.title + '">');
document.write('lava@pele.bigu.edu</a><br>');
document.write('Volcanic Studies, <a href="http://www.bigu.edu/">');
document.write('Big University</a><p>');

// append a modification date only if server provides a valid date
if (Date.parse(document.lastModified) > 0) {
        document.write('<b>last modified: </b>' +
               document.lastModified + '<BR>');
}
document.write('<b>URL: </b>' + document.location.href +
          '</FONT><P>');

// done hiding from old browsers -->
</SCRIPT>
<!-- end of Volcano Web standard footers -->
```

**NOTE: It is critical that all of your JavaScript statements are on a single line, starting from `document.write('...` and ending with `');`, If you have extra RETURN characters in the code, it will not work. Also, be sure the structure of the `if (...) {` block of statements is exactly as shown above.**

**JavaScript is even more picky than HTML about typos! And there is more room for mistakes with the ways quote characters are used.**

4. **Save** and **Reload** in your web browser. If there are no JavaScript errors, then see if the content in the footer appears similar to the example shown above (the URL should be different, of course, it will generate a URL that may include your local hard drive). If you do get JavaScript errors, compare your code in detail with the example. Finally, if nothing appears, try **reload** again. If this does not work, most likely your JavaScript code is missing a quote or a ">".
5. Now you are ready to cut and paste the new JavaScript footer into the same areas of these other files that are part of your *Volcano Web*:
   a. **explode.html**
   b. **height.html**
   c. **intro.html**
   d. **mars.html**
   e. **msh.html**
   f. **term.html**
   g. **usa.html**

We have not yet updated the footer that is part of our **project** page, which we broke into a framed set up in lesson 26. You may think that all we have to do is to paste it into the frame on that page for the footer document, **proj_footer.html**. But the problem here is that the JavaScript function **document.location.href** will not display the URL for the framed web page, "...project.html" but for the footer document. Also, many browsers return the file name rather than the page title for **document.title** of a framed web page. We would end up with something that looked like:



To get around this problem we could go back to a regular HTML footer... or use some more creative (and complex) JavaScript. We will take the second option, using JavaScript to "extract" the information we wish to display from the **document** object.

1. Open your **proj_footer.html** in your text editor
2. Delete the old footer content as we did in the above sections.
3. Replace it with this JavaScript code:

```
<!-- start of Volcano Web standard footers -->
<SCRIPT LANGUAGE="JavaScript">
<!-- hide  scripts from old browsers

document.write('<p><hr><FONT FACE="helvetica,arial" size=-1><i>Volcano Web:</i>
<b>');
document.write('Project</b><BR>');
document.write('created by Lorrie Lava, ');
document.write('<a href="mailto:lava@pele.bigu.edu?subject=Project">');
document.write('lava@pele.bigu.edu</a><br>');
document.write('Volcanic Studies, <a href="http://www.bigu.edu/">');
document.write('Big University</a><p>');

// append a modification date only if server provides a valid date
if (Date.parse(document.lastModified) > 0) {
        document.write('<b>last modified: </b>'
          + document.lastModified + '<BR>');
}

// extract proper URL from this file name, assuming this file
// is "proj_footer.html" or "proj_footer.htm"
// and the proper URL for the frameset is "proj.html" or "proj.htm"
myURL = document.location.href;

// Get the suffix on the file name (everything after "_footer")
myExt = myURL.substring(myURL.indexOf("_footer") + 7, myURL.length)

// Get the part of the URL that goes up to "proj"
myUrl = myURL.substring(0,myURL.indexOf("_footer"))

// Assemble the appropriate string
document.write('<b>URL: </b>' + myUrl + myExt + '</FONT><P>');

// done hiding from old browsers -->
</SCRIPT>
<!-- end of Volcano Web standard footers -->
```

4. **Save** this document and **Reload** the main frame web page, **project.html** in your web browser.

In this example, we have used some more advanced JavaScript functions to do things like extract portions of one string from another, and finding the location of a particular character in a string. Unfortunately, explaining it all is beyond what we can cover here; please see our references for recommended tutorials and resources.

## More Dynamic Content

Now we will show you even more things to do with JavaScript. Another built-in functionality is the ability to use the viewer's computer to get the date and time

(assuming they have it set correctly!). We can use this to "stamp" a greeting on the front page. We could write "Good Morning!", "Good Afternoon!", "Good Evening!", or "Isn't it Late to be At the Computer?" depending on what time of day the function returns to us through JavaScript.

The first thing you have to do is to create a **Date** object in JavaScript:

```
var today = new Date();
```

Now we have something called **today** that we can reference to get information about the date and time. JavaScript stores this information internally as something like the number of seconds since a reference date such as January 1, 1900. But the JavaScript **Date** object has properties that allow us to get the month, day, year, hour, minute, second, etc.

One of the easier object properties to use is **Date.toLocaleString()** which will display the date in accordance to the settings of the user's computer (since there are different conventions for displaying dates in other countries). So we could write code like:

```
var today = new Date();

document.write('According to my JavaScript watch, it is ' + today.toLocaleString()
)'
```

which would display like:

If you **reload** this lesson page, the time on the display will change. Note that the date format returned is dependent on the type of computer.

JavaScript offers other information about the web browser that the viewer is using, via the **navigator** object, so we can test which web browser it is (NetScape, Internet Explorer, etc) as well as which version. This is useful if you are using features in a web page that require certain web browsers-- you can use JavaScript to "test" the set up and display information specific for different browsers or versions.

We will now use JavaScript Date objects and navigator objects to display a customized greeting in our main page. we will do some extra work so that JavaScript displays the day of the week ("Monday, Tuesday", etc) and then write some message if the viewer has an older browser version. We briefly explain below what the code is doing, but if you just want to try it out, you may copy, paste, and test in your browser.

1. Open the **index1.html** document in your text editor.
2. Between the part that reads:

```
In this lesson you will use the Internet
to research information on volcanoes and then write a report on your
results.

and

<br clear=left>
<hr>
<p align=center>
```

insert this JavaScript code:

```
<p>
<SCRIPT LANGUAGE="JavaScript">
<!-- hide from old browsers

// get date object
var today = new Date();
var days = new
Array('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday');

// write the date
document.write('Welcome on this fine <b>' + days[today.getDay()]);
document.write('</b>, or more precisely,<br>');
document.write('<font color="#FFFF33" size=+2><tt>');
document.write(today.toLocaleString() + '</tt></font>');

// write a welcome string acknowledging the browser version
document.write('<p>We hope you enjoy your web experience with your version of ');
document.write(navigator.appName + ' ' + parseFloat(navigator.appVersion));

// write a warning for old web browsers
if ( parseInt( navigator.appVersion) < 3 ) {
        document.write('<font color=#FFFF33> Hmmm.. that\'s a pretty old version of
');
        document.write(navigator.appName + '! Perhaps it is time to consider an
upgrade?</font>');
}

// done hiding -->
</SCRIPT>
```

**NOTE: On the last section of this code, you may see something that looks out of place, a forward slash \ in the word that's. This is not a typo! You need this special marker so that JavaScript interprets the single quote as an apostrophe character and not the single quote that marks the end of the JavaScript text string. If you remove this forward slash, the code will generate an error.**

3. **Save** and **Reload**.

In this script we first create the **Date** object and put it in a variable we call **today**. We then create an "array", or a list of things, called **days** that contains names of the days of the week.

Arrays are very handy containers because we can refer to items in them like:

```
days[2]
```

where the number in the square brackets indicates the location in the array for the item we are looking for. But watch out! JavaScript starts counting arrays at 0, so **days[2]** actually returns the third item, or "Tuesday".

Our code writes a welcome text string and then uses the array to extract the correct name of the day. The **Date** object function **today.getDay()** returns a number from 0 to 6 that corresponds to which day of the week it is. So we can combine the **Date** object function and our array of names to print the correct day of the week.

Following this, we use the format provided from **today.toLocaleString()** to write the full date information.

The next piece prints the browser name ("NetScape", "Internet Explorer") followed by the version returned to us by the function **navigator.appVersion**, which actually returns a longer descriptive name. By putting that inside a function called **parseFloat**, it pulls out or "parses" the part of that string that corresponds to a whole number. As a comparison, see:

a. navigator.appName
   Microsoft Internet Explorer
b. navigator.appVersion
   4.0
c. parseFloat(navigator.appVersion)
   4
d. parseInt(navigator.appVersion)
   4
e. navigator.appName + ' ' + parseFloat(navigator.appVersion)
   Microsoft Internet Explorer 4

This last item provides a nicely formatted string to display the browser and version number. The code **parseInt(navigator.appVersion)** parses out the whole number part of this information. if you look at the last portion of our code above, we test the parseInt value and write an extra warning if it is less than our test value of 3. To see the difference, you should change the 3 to a number higher than your present browser version.

## Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Note that JavaScript is **very** sensitive to typographical mistakes -- one missing quote or semi-colon can ruin the page!

## Review

Review topics for this lesson:

1. Why does it matter where in your HTML document you insert JavaScript code for writing dynamic content?
2. What information can you get from the JavaScript **document** object? What is the format for printing that information to the web page?
3. How do you get the current date and time from JavaScript? the day of the week?
4. Using JavaScript, what information can you get about the web browser?

## Independent Practice

Copy the format for the JavaScript footer used here to your own documents and see if you can change the HTML format to match your design.

Can you think of a way to write JavaScript code that would display a different welcome message for every hour of the day? *Hint:* Use an array to create the text of all your messages, created a date object called **today** and use the **today.getHours()** function to determine the current hour.

---

# Coming Next....

Your JavaScript Doctor gives you the next dose of code... functions and code for opening browser windows of any size and configuration. Where you want them and with as many or as few browser buttons as you like.

**GO TO.... | Lesson Index | previous: "JavaScript: Alerts and MouseOvers" | next: "JavaScript: Custom Window Openers" |**

---

**Writing HTML:** Lesson 27b: A Wee Dose of JavaScript : Dynamic Content
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut27b.html

# 27c. JavaScript : Window Openers

## So you want to open Browser Windows?

## JavaScript Does Windows...

## Where you Want Them, How big You Want Them, and with What Browser Buttons You Choose...

# Objectives

After this lesson you will be able to:

- Write JavaScript code that opens another web browser window of any size and any combination of browser features
- Write JavaScript code that opens another web browser window in a specific screen location (for version 4.0 browsers)
- Write a JavaScript function in the **`<HEAD>...</HEAD>`** that can be used several times

---

# Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In lesson 25, we learned how to add a TARGET option to a hyperlink so that a mouse click would load the link in a new browser window. You may have noticed that this second window opens with the same web browser buttons and fields as the original window, and that it is usually the same size as that first window.

With JavaScript we can write code that creates a new window of arbitrary size, and we can instruct it not to display the URL field or the web browser navigation buttons. Furthermore, for 4.0 version web browsers, we can even specify where the window should appear on the screen.

Why would we want to do this? If we are doing something like loading an image in a new web browser window, there is no real need for all of the browser buttons and fields, which take up valuable screen real estate. Furthermore, it allows us to create small web browser windows that can work like "tool palettes" in other computer programs.

How about trying it out?

Look at this photo of a crater in Mexico that opens in long skinny window or this one of a volcanic mountain in Arizona that opens in a window that fits the image. Finally, try looking at our pink JavaScript bottle in a small window that displays

browser buttons.

The general JavaScript approach to writing this code looks like:

```
<a href="URL"
onClick="window.open('URL', 'window_name', 'window_options'); return false">
linking text</a>
```

where **`URL`** is the URL for the image that we wish to display in this new window. As we saw in lesson 27a we use the same URL in the **`href`** part and include a **`return=false`** in the JavaScript **`onClick`** code just in case the person looking at our page has a browser that does not understand JavaScript. In this case, the link tag will jump to a page that displays the picture by itself in a full web page.

The **`onClick`** JavaScript event initiates the action; a command called **`window.open()`** that talks to the web browser and tells it to open a new window. The window is filled with the content specified by the URL in the first parameter, which in the above examples was an image file, but could also be a local HTML file or a remote URL. The second parameter provides a "name" for this window, in case we use it again to target a link (see lesson 25).

The third parameter is where we list all the options that indicate how the window will appear. These items are all in one string of text enclosed in quotes:

1. `toolbar`
   displays the browser buttons (forward, back, home, print, etc)
2. `location`
   displays the field that shows the URL for the window
3. `directories`
   displays other web browser directory buttons
4. `status`
   displays the browser status bar at the bottom
5. `menubar`
   displays the web browser menu bar
6. `resizable`
   allows user to change the size of the window
7. `scrollbars`
   provides scroll bars if the content is larger than the window size
8. `width=XX`
   specifies the width of the window when opened, in pixels
9. `height=YY`
   specifies the height of the window when opened, in pixels

You can list the first seven as `OPTION=yes` or `OPTION=no` to indicate that we want them to be used or not used in our newly opened window, separating each by a comma. For example:

```
I have provided a sneak peek of
<a href="myimage.gif"
onClick="window.open('myimage.gif', 'myWin',
'toolbar=no, directories=no, location=no,
status=yes, menubar=no, resizable=no, scrollbars=no,
width=300, height=200');
return false"
```

```
>my image</a> for you to see.
```

will create a link to open an image file **myimage.gif** in window named "myWin", that shows no browser buttons, and is pretty much a fixed window size; you cannot resize it or scroll it.

On the other hand, we could link to a URL in a new window:

```
How about trying the best
<a href="http://www.mcli.dist.maricopa.edu/tut/"
onClick="window.open('http://www.mcli.dist.maricopa.edu/tut/',
'myWin', 'toolbar=yes, directories=no, location=no, status=yes, menubar=no,
resizable=yes, scrollbars=yes, width=500, height=400'); return false"
>web page tutorial in the universe</a> for free!
```

This code provides the browser navigation buttons (but not the extra directory ones) and the URL display field. The window is opened at a window that is 500 pixels wide and 400 pixels high, and we allow the viewer to resize it and the window will have scrollbars activated.

You can write these options more compactly by just listing the ones that you wish to activate; the browser assumes the ones not mentioned are turned "off". For examples, here is the alternative way to write the two examples above:

```
I have provided a sneak peek of
<a href="myimage.gif"
  onClick="window.open('myimage.gif', 'myWin',
  'status, width=300, height=200'); return false">my image</a>
  for you to see.

 How about trying the best
 <a href="http://www.mcli.dist.maricopa.edu/tut/"
   onClick="window.open('http://www.mcli.dist.maricopa.edu/tut/',
   'myWin',
   'toolbar,status,resizable,scrollbars,width=500,height=400');
   return false">
   web page tutorial in the universe</a> for free!
```

With the 4.0 versions (and later) of NetScape and Internet Explorer browsers, we can also, through JavaScript, specify the location on the screen that the window opens. These dimensions are given in pixels, measured horizontally and vertically down from the top left corner of the screen. Older web browsers will just ignore these options.

Unfortunately, NetScape and Internet Explorer use different names for these options! So to make it work on both browsers, you must specify the settings twice:

1. `screenX=hh,screenY=yy`
   specifies the location of the upper left corner of the window, measured from the top left corner of the monitor (for NetScape 4.0 and later)
2. `left=hh,top=yy`
   specifies the location of the upper left corner of the window, measured from the top left corner of the monitor (for Internet Explorer 4.0 and later)

Adding on to our earlier example, This code will tell 4.0 version web browsers to place the window 100 pixels over and 75

pixels down from the top left corner of the viewer's screen:

```
I have provided a sneak peek of
<a href="myimage.gif"
onClick="window.open('myimage.gif', 'myWin',
'status, width=300, height=200, screenX=100,screenY=75,left=100,top=75');
return false">my image</a> for you to see.
```

## Putting it in Your <HEAD>

We will soon move on to our *Volcano Web* example for creating custom window openers. To date, we have written small JavaScript segments that are added to hyperlink tags or embedded into the body of a page to write dynamic content. For a task like opening windows to display a series of images, it makes sense to write a generic chunk of JavaScript code that can do this task, and all we have to tell it is which picture it should load. This is the purpose of a JavaScript **function**, a block of code we can re-use as much as we need to perform a similar task.

The typical place to write JavaScript functions is to put them into the non-display part of our document inside the **`<HEAD>...</HEAD>`** tags. This way, we can load the "brains" of our JavaScript into the web browsers, and then call it as we need from our document.

For our *Volcano Web* site, we are going to modify the page that discusses Volcanoes from the USA. We have a link from a small picture of a seismometer that loads a larger verson of that image into a separate web browser window (using the TARGET attribute of the hyperlink tag). We will add some links to this document that display pictures of other Volcano landforms, and we will use JavaScript to load these images into a new window.

But rather than just loading a picture in a window, we want to tell JavaScript, "Here is the image file, a title, and a caption; please load a new web browser window with the image and write some caption text on the bottom of the page". So we will use JavaScript to create the new window and then write dynamic content to that window from a template. We want this ability to be written as a function, so we can use it for 4 different images.

1. Open the **`usa.html`** file in your text editor.
2. Between the tags:

   ```
   <head>
   <title>Volcanic Places in the USA</title>
   ```

   and

   ```
   </head>
   ```

   add this JavaScript function:

   ```
   <SCRIPT LANGUAGE="JavaScript">
   <!-- hide this script from old browsers

   // This script opens a new browser window and writes
   // HTML to display an image with a title and caption

   function show_photo( pFileName, pTitle, pCaption) {
   ```

```
// specify window parameters
  photoWin = window.open( "", "photo",
     "width=600,height=450,status,scrollbars,resizable,
     screenX=20,screenY=40,left=20,top=40");

// wrote content to window
  photoWin.document.write('<html><head><title>' +
    pTitle + '</title></head>');
  photoWin.document.write('<BODY BGCOLOR=#000000 TEXT=#FFFFCC
    LINK=#33CCFF VLINK=#FF6666>');
  photoWin.document.write('<center>');
  photoWin.document.write('<font size=+3
    face="arial,helvetica"><b>' +
    pCaption + '</b></font><br>');
  photoWin.document.write('<img src="' +
    pFileName + '"><p>');
  photoWin.document.write('<font face=
    "arial,helvetica">');
  photoWin.document.write( '"' + pTitle +
    '" photo &copy; Lorrie Lava<br>');
  photoWin.document.write('<a href="mailto:
    lava@pele.bigu.edu">
    lava@pele.bigu.edu</a><br>');
  photoWin.document.write('Volcanic Studies,
    <a href="http://www.bigu.edu/">
    Big University</a>');
  photoWin.document.write('<p></font>
    </body></html>');
  photoWin.document.close();

// If we are on NetScape, we can bring the window to the front
        if (navigator.appName.substring(0,8) ==
           "Netscape") photoWin.focus();
}
// done hiding from old browsers -->
</SCRIPT>
```

If you would like to copy and paste this code, use this sample of HTML that will load in a new browser window.

Our function is named **`show_photo`**; when we use it in our pages, we will send it three pieces of information listed as the **variables** **`pFileName`** (the URL for the image file), **`pTitle`** (a string of text for the title of the web page we will create via JavaScript), and **`pCaption`** (a short caption to display below the image). These three items can be different every time we use it but are represented in our function by their variable names.

We then use the **`window.open`** function to create a window that is 600 wide by 400 pixels high; note that unlike our earlier examples, the first parameter for this function, the window URL, is empty. That's because the page that will fill the window does not exist. In fact, we will use the **`document.write`** function in the subsequent lines to write the entire contents for that page. By using the notation **`photoWin.document.write`**, we are telling the web browser to write the content to the new window using the template provided. Note that near the end, we let the browser know we are done by the code **`photoWin.document.close();`**

The last little bit is some special code that allows our function to tell NetScape browsers to bring this window in front of any other web browser windows (This function is not yet available to Internet Explorer browsers).

As it stands now, we only have built the functionality for our window opener. We will now create the code that "calls" the function:

1. In your **usa.html** document, replace the two ocurrences of:

   ```
   <a href="../pictures/seismo.jpg" target="photo">
   ```

   with this new code:

   ```
   <a href="'../pictures/seismo.jpg"
     onClick="show_photo('../pictures/seismo.jpg', 'Field Seismometer',
     'Volcanic Tableland, Long Valley, California'); return false"
     onMouseOver="window.status='view a large image of a field seismometer';
     return true">
   ```

   **NOTE: We use the onClick event to trigger a call to our JavaScript function. The picture to load in the new window is seismo.jpg, the title for the new page is Field Seismometer, and the caption to be placed at the bottom is Volcanic Tableland, Long Valley, California. If our function works, it will generate a web page in a new window that contains this information in the format specified by our function.**

   **We also use the onMouseOver event to control the display in the status bar when the user moves the mouse over this link (see lesson 27a)**

2. **Save** and **Reload** in your web browser.

When you click on either the postage stamp image of the seismometer or the hypertext link below it, a new web browser window should appear, and the larger picture is displayed in a black background page with yellow text (you may have to scroll down to see the caption). If it did not work, compare it to the sample web page for this part of this lesson.

We will now demonstrate the usefulness of writing JavaScript functions-- using the same code for different content. In our previous versions of the document we have been working on, **usa.html** the link listed under Mount St Helens went to another web page, **msh.html**, which had only one sentence and a link to an image. We will no longer use this second web page, and instead, we will modify our first page to just display this image in a new browser window.

1. Open your **usa.html** file in your text editor.
2. Change the HTML under the Mount St. Helens section that reads:

   ```
   <td valign=top>On May 18, 1980, after a long period of rest,
   this quiet mountain in Washington provided
   <a href="msh.html">detailed observations</a> on
   the mechanics of highly explosive eruptions.
   </td>
   ```

   to read:

   ```
   <td valign=top>On May 18, 1980, after a long period of rest,
   ```

```
this quiet mountain in Washington provided detailed observations on
the mechanics of highly explosive eruptions. The towering pine trees
of this once-quiet mountain were <a href="../pictures/msh.gif"
onClick="show_photo('../pictures/msh.gif', 'Trees Toppled',
'Mount St. Helens Blast Area'); return false"
onMouseOver="window.status='view a photo showing the strength of the eruption';
return true"> toppled over like toothpicks</a>.
</td>
```

**NOTE: We are using the same JavaScript function that we wrote for the link to the seismometer image to display a different image title, and caption for the Mount St. Helens picture.**

3. **Save** and **Reload** in your web browser.

But let's not stop here! We will add two more description sections to this document, and use our JavaScript function to display two new volcano images.

1. Go to the Lesson 27c Image Studio to get copies of the pictures of MacDougal Crater and Mt Agassiz.
2. Open your **`usa.html`** file in your text editor.
3. We will add a new row for our table in this page, so immediately before the `</table>` tag, add this code:

```
<tr>
<td>
  <font size=+1><b>
  San Francisco Peaks
  </b></font>
</td>

<td colspan=2>
  <font size=+1><b>
  MacDougal Crater
  </b></font>
</td>
</tr>

<tr>
<td valign=top>Scientists believe that the volcanic eruptions
several million years ago that shaped
<a href="../pictures/agassiz.jpg"
onClick="show_photo('../pictures/agassiz.jpg', 'Mount Agassiz',
'San Francisco Peaks, Arizona'); return false"
onMouseOver="window.status='view an image of a volcano in Arizona';
return true">this mountain in northern Arizona</a>
were very similar to the ones observed at Mount St. Helens.
</td>

<td valign=top colspan=2>
When hot volcanic magma encounters ground water,
these explosive eruptions can
form <a href="../pictures/macdougal.jpg"
onClick="show_photo('../pictures/macdougal.jpg', 'MacDougal Crater',
```

```
'Pincate Volcanic region, Mexico'); return false"
onMouseOver="window.status='view an image of a volcanic crater in Mexico';
return true">deep craters</a> seen just south of Arizona in
the Pincate Volcanic region of Mexico.
</td>
</tr>
```

4. Since we now have more than two volcanic places listed on this page, change the first sentence to read:

```
Listed below are places in the United States that are considered
"active" volcanic areas.
```

   (remove the word "two")
5. **Save** and **Reload** in your web browser.

If all works well, you will have four different links that reference the same JavaScript function to generate different content!

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Note that JavaScript is **very** sensitive to typographical mistakes -- one missing quote or semi-colon can ruin the page! Make sure to check that you do not have extra RETURN characters inside your JavaScript statements.

# Review

Review topics for this lesson:

1. What are the differences between opening a new browser window with JavaScript as compare to using the TARGET attribute in a hyperlink?
2. How can you open a new browser window to this web lesson page that has no browser buttons but shows the URLs in the display field?
3. How can you make a new web browser window open 300 pixels down from the top of the screen? Will it work on all web browsers?
4. Why do we place JavaScript functions in the **<HEAD>..</HEAD>** of our HTML documents?
5. What are advantages of using JavaScript functions?

# Independent Practice

Use our custom window opener in your own web pages. See if you can update the format for the HTML that it writes dynamically.

What are some purposes you might want to use a small window that opens from your site? Can it be used like a Navigation control?

---

# Coming Next....

It is our last JavaScript Dose... Swapping images in response to user interactions...

**GO TO.... | Lesson Index | previous: "JavaScript: Dyanmic Content" | next: "JavaScript: Swapping Images" |**

---

**Writing HTML:** Lesson 27c: A Wee Dose of JavaScript : Window Openers
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut27c.html

# 27d. JavaScript : Swapping Images

## Dynamic Animation!

## Move Your Mouse over the Bottle!

## Move it Away!

## Are You Ready to Create the Same Effect?

# Objectives

After this lesson you will be able to:

- Identify the areas of the HTML document that need to be edited to create JavaScript image swapping.
- Describe the different between mouse enter events and mouse exit events.
- Write JavaScript code that will hide the scripts if the viewer's browser does not support this feature.

Write the JavaScript code to implement image swapping.

---

# Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In this lesson we show you how to create an effect that you likely have seen at other web sites-- causing an inline image to change when you move the mouse over it or click on it. For the example below, the arrow will "bulge" when the mouse moves over it, and it will change to a "pushed-down" appearance, once it is clicked:

up state

Once the button is "locked" down, you must **Reload** this page in your browser to reset it.

The most common use of this technique is to animate hyperlinked images that act as buttons, so the viewer gets an extra reinforcement that the object under the mouse is something worth clicking.

While you are free to use what you learn in this lesson to make buttons pop and jump, our stodgy opinion is that this adds nothing to the information on the page, so our example will show you a use that we think is a bit more worth the effort.

You should consider another factor; for every image that you will cause to change when the mouse moves over it, you actually must download 2 different images, and as you will see shortly, this must be done before the HTML for the page loads. Therefore, this can increase the precious wait time that the viewer will be watching a blank screen.

The way image mouseOver swapping works goes something like:

1. Web browser starts reading the HTML for the page
2. The **<HEAD>...</HEAD>** area includes statements that first download two image files and place them into temporary hidden storage.
3. The normal **<img...>** tag loads the image that should first appear.
4. A JavaScript "event handler" in a hyperlink tag around the image tests for whether the mouse is leaving or entering the area of the page covered by the image.
5. If the mouse enters the image, we call a JavaScript function to swap the file source for the image for the file that represents the appearance for when the mouse is over the image.
6. When the mouse leaves the area of the image, we call a second JavaScript function to swap back the original image.

   **NOTE: To work correctly, the images that are swapped for each other must be the same width and height.**

Below is a generic method for button swapping. The one used in this page has a few more features that you are encouraged to explore by looking at the HTML source of this lesson page.

# JavaScript Image Swapping

| HTML code | explanation |
|---|---|
| `<html>`<br>`<head>`<br>`<title>My Page</title>`<br>`<head>`<br>`<SCRIPT LANGUAGE="JavaScript">`<br>`<!--` | Typical top of standard HTML file that has JavaScript functions in its **<HEAD>...</HEAD>**. |
| `if (document.images) {` | This is a test that determines if the web browser understands the image storage functions needed for swapping images. If this test is false, we would skip the rest of the code and just display a static image. |
| `var button1_up = new Image();`<br>`button1_up.src = 'button_up.gif";` | We create a **variable** that is the type that represents Images. The second line assigns the **src** property with the path to the first image file to the "up" version of the image. |

```
  var button1_over = new Image();
  button1_over.src = 'button_over.gif";
```

We create a second Image **variable** that assigns the **`src`** property to the path to the second image file for the "up" version of the image.

```
function over_button() {
  if (document.images) {
    document["buttonOne"].src = button1_over.src
  }
}
```

This function, when called will again make sure the browser can do the image swapping. If so, it looks inside its internal list of inline images stored in the document object, and assigns it the variable that represents the image for the mouse over or highlighted image.

```
function up_button() {
  if (document.images) {
    document["buttonOne"].src = button1_up.src
  }
}
```

This function works almost the same, except that it will make the image swap back to the "up" version.

```
//-->
</SCRIPT>
<body>
   :
   :
   :
```

End of the JavaScript and continuation of the HTML to display the page's content.

```
<a href="file.html"
  onMouseOver="over_button()"
  onMouseOut="up_button()">
  <img src="button_up.gif" alt="click me"
    width="XX" height="YY" name="buttonOne"
    border=0>
</a>
```

The tag used to load the first view of the image (the "up" version). Notice that we have assigned the image inside the **`<img...>`** tag a unique name, "buttonOne", that we can refer to it using code like **`document["button name"]`** The hyperlink tag contains a JavaScript event handler, **`onMouseOver`** that will trigger a call to our function to swap in the "highlight" image when the mouse moves over the image. Likewise, the **`onMouseOut`** event will call our other function to change the image back when the mouse leaves, or "moves out" of the image area.

Now we will see how this works in our volcano web example. Rather than just animating buttons, we are going to simulate a microscope!

As part of our Introduction, we will add a new section that describes volcanic rocks, and include an image of one kind, called "pumice". We will use JavaScript to change the image to a second one that shows a cut away view of how pumice looks under a microscope. By moving the mouse on and off the image, we can compare the two different views (Okay, we admit it

is not much more creative than animating buttons and one could achieve the same result by placing the images side by side...)

1. Go to the Lesson 27d Image Studio to get the two image files needed for this lesson. The should be placed in the **`pictures`** directory with all of your other image files.
2. Open the **`intro.html`** file in your text editor.
3. Inside the **`<HEAD>...</HEAD>`** tags, add the following JavaScript code:

```
<SCRIPT LANGUAGE="JavaScript">
<!--
if (document.images) {
   var pum1 = new Image();
   pum1.src = "../pictures/pumice.gif";
   var pum2 = new Image();
   pum2.src = "../pictures/pumice2.gif";
}

function show_rock() {
        if (document.images) {
                document["pum"].src = pum2.src;
        }
}

function hide_rock() {
        if (document.images) {
                document["pum"].src = pum1.src;
        }
}
//-->
</SCRIPT>
```

> **NOTE: We have created two JavaScript "holder" variables that represent the two images. We will call the `show_rock()` function to swap in the microscope view image `pumice2.gif` and the `hide_rock()` function to swap back the original image. These events occur for an image named "pum" we will next identify in the HTML code.**

4. After the end of our table on well known volcano eruptions, following the sentence that ends:

```
than ones observed by humans.
```

We will add a new section about Volcanic Rocks. It will contain the image for the pumice rock and the code for the JavaScript image swapping:

```
<h3 align=center>Volcanic Rocks</h2>
Scientists study rocks at many different scales.<p>
<b>Pumice</b> is one kind of rock formed by volcanic eruptions
<a href="intro.html"
  onClick="alert('Move the mouse over the rock to see a magnified view.');
  return false"
  onMouseOver="show_rock(); window.status='description of explosiveness scale';
  return true"
```

```
  onMouseOut="hide_rock()">
<IMG SRC="../pictures/pumice.gif" align=right ALT="picture of pumice"
  WIDTH="220" HEIGHT="170" hspace=12 vspace=12 name="pum" border=0></a>
that are very explosive. Hot, frothy volcanic <b>magma</b> quickly cools,
leaving a structure of many twisted air holes inside. Pumice is thus very
light weight.
<p>
If you move your mouse over the image, you can see how pumice looks
under the microscope.
<p>
A <b>thin section</b> is a layer of the rock cut so thin that the
light from a microscope shines through, allowing us to see the
structure of the rock.
<p>
The twisted chambers here represent the air pockets preserved inside
the rock when this rock blew out of a volcano.
<br clear=right>
```

> **NOTE: Take extra care on the code for the `<img...>` tag; it is pretty complex now! It now handles the mouseOver actions, displays a status bar message (like we did in lesson 27a), and will display an `alert` message in case the viewer clicks the image (keeping them on the same page). The image is also right aligned and has extra padding around it using (`hspace` and `vspace`)**

5. **Save** and **Reload**

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Note that JavaScript is **very** sensitive to typographical mistakes -- one missing quote or semi-colon can ruin the page! Make sure to check that you do not have extra RETURN characters inside your JavaScript statements.

# Review

Review topics for this lesson:

1. What areas of your HTML document must be edited to create JavaScript image swapping?
2. What is the difference between `onMouseOver` and `onMouseOut`?
3. What does the viewer see if their browser returns a value of `FALSE` for the expression `document.images`?
4. What are the essential ingredients to write JavaScript code for image swapping (hint: consider the image files, code to load images, code to swap images, and code to initiate events)?

# Independent Practice

Find some images that you can use for swapping; remember they must be the same exact size! Use our code to create the same effect in your own pages. What would it take to have two different images be able to change like this? Why would this approach be less desirable if our page had 10 active images that could be swapped for alternative ones?

Note that you can also write JavaScript functions to perform a different task generated by a mouse click:

```
<a href="file.html" onMouseOver="over_button()" onMouseOut="up_button()"
  onClick="doButtonClick()">
  <img src="button_up.gif" alt="click me" width="XX" height="YY"
  name="buttonOne" border=0>
</a>
```

provided that you had written a JavaScript function in your **<HEAD>...</HEAD>** to react to this event

---

# Coming Next....

It's time we livened things up with some interactive FORMs...

**GO TO.... | Lesson Index | previous: "JavaScript: Window Openers" | next: "FORMs"**

---

**Writing HTML:** Lesson 27d: A Wee Dose of JavaScript : Swapping Images
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut27d.html

.html

# 28. Adding some FORM to your webs

Transform your web pages from just `text` and pretty pictures to things you can

click

and

select

## Objectives

After this lesson you will be able to:

- Describe potential uses for web page forms.

---

## Lesson

In the next three lessons we will learn how to create web page forms. What are forms? You have almost certainly seen and used them on other web sites. We provide one on our alumni sign up page, as a place for you to send us information about what you have learned in Writing HTML.

Forms are used to add another level of interactivity to your web pages, to allow communication between your viewers and your web site, to gather information, and to offer different means of navigation.

The role of forms is to gather different kinds of user input, i.e. fields to type in text, menus to select items from, radio buttons to choose items. The web browser takes this information, and wraps it up into a packaged format that can be sent directly to a web server, where there is a customized program sitting and waiting for the form information. These programs can unpackage the information, manipulate it, store data, and send a feedback page back to the viewer.

In the next lesson, you will learn how to write the HTML to display different form elements, like the

ones displayed at the top of this lesson page. After that, we will show you first how you can make the forms work by emailing the content directly to you as well as how it can be processed with one of those web server ("CGI") programs. Finally, we will show you how you can use JavaScript to provide some of the same functionality as CGI programs, without having to deal with messy programming or running a web server.

# Check Your Work

This is an introduction, so we have not made any changes to our project.

# Review

Review topics for this lesson:

1. How are forms used?
2. Where have you seen and used a web form?
3. What would prevent you from being able to use forms that required CGI programming?

# Coming Next....

Let's ???? some web page forms...

**GO TO.... | Lesson Index | previous: "JavaScript: Swapping Images" | next: "Forming Forms" |**

---

**Writing HTML:** Lesson 28: Adding some FORM to your webs
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut28.html

# 28a. Forming Forms

This web page would be so much ◉ better ○ worse ○ no different if only we were able to use forms ☑ to add interactivity ☐ to make me look cool ☐ to enliven every page, plus it would really knock the _____ off of everybody at the ?? when they

pressed my button! and see the wonderful things that can happen by clicking, selecting, and typing in fields, and more clicking!

## Objectives

After this lesson you will be able to:

- Write the HTML for the basic web form elements
  - text fields
  - password fields
  - text area fields
  - radio buttons
  - checkboxes
  - menu selections
  - buttons

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In this lesson we will introduce the basic web form elements that you might use in your pages. They will not actually **do** anything until the next lesson.

A web page form is defined by a set of **`<form>...</form>`** tags where everything in between includes HTML to code the different text fields, buttons, and drop down menus that are used to store selections from the page viewer. You can also have other HTML code inside the form that serve as labels for your form elements.

As an example, look at the source HTML used for the Writing HTML Alumni page. This page includes several different form elements for inputting text. The page layout is defined by using tables to arrange the form elements. The HTML on that page form includes different input elements that as well as the table tags that define their layout.

The bare bones format for writing a form is:

```
<form action="[url for server program]" method="GET/POST">
  :
  :
  (form elements)
   :
```

```
</form>
```

We will talk more in the next lesson about the meaning of the options in the **<FORM>** tag. For now think of the value for **ACTION=...** as something that tells the browser the location of a script or program that will process the data sent by the browser via the selections in the form. The two values for **METHOD=...**, **POST** or **GET** define one of two ways the data from the form is sent to the program that will process the data.

Now we will review the different form elements you can use. All of the elements inside the **<FORM>** tags can send some information about their contents and whether or not they have been activated by the web page viewer. Each element includes a defined "type" plus a unique indentifying name for that element. When the form data is sent to the web server, each element sends its name and its current state or value.

## Text Input Elements (`type="text"`)

... are used to create one line fields that viewers can type text. The default width is 20 characters, and you can create fields of other sizes by the value in the **size** option. You can limit the number of characters by the value of the **MAXLENGTH** option. Text input fields will be empty when the page loads, unless you provide an initial text string for its VALUE option.

**sample web page**

A text field named "text1" that is 30 characters wide.

```
<input type="text" name="text1" size="30">
```

A text field named "text2" that is 30 characters wide but will only accept 20 characters.

```
<input type="text" name="text2" size="30" maxlength="20">
```

A text field named "text3" that is 40 characters wide with default value.

```
<input type="text" name="text3" size="40" value="We are not
alone">
```

We are not alone

## Password Input Elements (`type="password"`)

... are exactly the same as text input elements, except that when the viewer types in, they see "bullet" characters rather then the letter they are typing. Password text is scrambled during transmission and then unscramble when the form data is received at the server end. See the difference between typing in the fields below and the ones in the previous section.

**sample web page**

A password field named "pass1" that is 30 characters wide.

```
<input type="password" name="pass1" size="30">
```

A password field named "pass2" that is 30 characters wide but will only accept 20 characters.

```
<input type="password" name="pass2" size="30"
maxlength="20">
```

A password field named "pass3" that is 40 characters wide with default value.

```
<input type="password" name="pass3" size="40" value="We are
not alone">
```

## Text Area Input Elements (`type="textarea"`)

... are text fields that have more than one line and can scroll as the viewer enters more text. The tag options define the size of the field by the number of rows and character columns. By adding the option **`WRAP=VIRTUAL`**, the text entered will automatically wrap at the right hand side of the field. You can also include default text to appear in the field.

**sample web page**

A textarea field named "comments" that is 45 characters wide and 6 lines high.

```
<textarea name="comments" rows="6" cols="45" wrap="virtual">
The first time I ever saw a web page, I thought....
(continue)
</textarea>
```

The first time I ever saw a web page, I thought....
(continue)

## Radio buttons (`type="radio"`)

... are sets of buttons that are linked so that only one among button in each sets is selected at a time; if you click one button, the others in the set are automatically de-selected. A set of radio buttons is defined by providing them the same name. The value sent in the web form is the value of the radio button that was last selected. Adding the option **`CHECKED`** to one of the buttons in a set will make that button highlighted when the page loads.

4 radio buttons with default selection

```
<input type="radio" name="food" value="hotdogs" checked>
hotdogs are my favorite food<br>
<input type="radio" name="food" value="hamburgers"> i like
hamburgers<br>
<input type="radio" name="food" value="steak"> steak is
tasty<br>
<input type="radio" name="food" value="beans"> beans are for
veggie-lovers<br>
```

- (•) hotdogs are my favorite food
- ( ) i like hamburgers
- ( ) steak is tasty
- ( ) beans are for veggie-lovers

3 radio buttons with no default selection

```
<input type="radio" name="spread" value="ketsup"> ketsup<br>
<input type="radio" name="spread" value="mustard">
mustard<br>
<input type="radio" name="spread" value="mayo">
mayonnaise<br>
```

- ( ) ketsup
- ( ) mustard
- ( ) mayonnaise

**NOTE: See how the two sets of radio buttons, one named "food" and the other named "spread" operate independently from each other.**

## Check Boxes (`type="checkbox"`)

...are similar to radio buttons, but are not affected by other buttons, so you can have more than one in a group checked at a time. Note that every checkbox has a unique name. If there is no check in the box, clicking it will place an X or a check mark there. If the box is checked, clicking it again will remove the mark. The value sent in the web form is the value of the checkbox if it was selected; otherwise the value will be empty. Adding the option **`CHECKED`** to one of the buttons in a set will make that button highlighted when the page loads.

**sample web page**

4 check boxes with default selections

```
<input type="checkbox" name="food1" value="hotdogs" checked>
hotdogs are my favorite food<br>
<input type="checkbox" name="food2" value="hamburgers"> i
like hamburgers<br>
<input type="checkbox" name="food3" value="steak" checked>
steak is tasty<br>
<input type="checkbox" name="food4" value="beans"> beans are
for veggie-lovers<br>
```

- [x] hotdogs are my favorite food
- [ ] i like hamburgers

☑ steak is tasty
☐ beans are for veggie-lovers

3 check boxes with no default selection

```
<input type="checkbox" name="spread1" value="ketsup">
ketsup<br>
<input type="checkbox" name="spread2" value="mustard">
mustard<br>
<input type="checkbox" name="spread3" value="mayo">
mayonnaise<br>
```

☐ ketsup
☐ mustard
☐ mayonnaise

## Menu Select (`type="select"`)

... provides drop-down menus that allow the viewer to choose one from a list of choices. The **`<OPTION>...</OPTION>`** tag defines the text that is displayed in the menu. The value of the option last selected is returned when the form data is transmitted. Adding the **`SELECTED`** option indicates which element is displayed initially when the page loads; if this is not provided, the first item is selected by default.

**sample web page**

A four item select menu with the third item selected initially

```
<select name="cheeses">
<option value="colby"> Colby from Ohio</option>
<option value="sharp"> Sharp Cheddar from Oregon</option>
<option value="swiss" selected> Holy Cheese from
Switzerland</option>
<option value="longhorn" > English Longhorn</option>
</select>
```

Holy Cheese from Switzerland

## Submit and Reset (`type="submit"` and `type="reset"`)

... creates buttons on the form. The Submit button tells the web browser to gather up all the selections, values, and entered text in the form elements and send it off to the place defined in the **`ACTION`** part of the form tag. The Reset button restores the form to its default state, how it looked when the viewer first entered the page. The **`VALUE`** option defines the text string that is placed on these buttons.

**sample web page**

Submit button

```
<input type="submit" value="Send this form data now!">
```

Send this form data now!

Reset button

```
<input type="reset" value="Clear the web form">
```

Clear the web form

**NOTE: For this lesson we have created a simple JavaScript alert message to appear when you try the Submit button. If you had entered any text, or changed any button.menu selection, the Reset button above will revert them all to their initial state.**

We will now create a web page form that uses all of these elements. This is going to be an additional feature of our *Volcano Web* project portion. The purpose of the form is to provide a place for people who are doing their projects to submit their reports. We could use this in several ways; it could send the reports as email to the instructor, it could write the report data to a file on the web server, or it could generate a formatted web page report for the student which they could then print. There are other things we could do with data sent by a web form and this is but one example.

Hopefully now you will see an advantage of using the frames display we created for this area in lesson 26. We will create one new web page that will display the form in the main display area and we will modify the left side frame that contains the menu of choices to add a link for our new page.

1. Open the **`proj_menu.html`** file in your text editor.
2. After the HTML code that contains the link information for the "Reference" frames web page, add:

```
<a href="proj_report.html">
<font size=+2 face="arial,helvetica">R</font>EPORT...</a><br>
a form to submit your report
```

3. **Save** this HTML file.
4. Now we will begin building the new web page that will contain the form. Create a new file in your text editor and save it as a file named **`proj_report.html`** stored in the same directory/folder as your other HTML documents.
5. To better learn the parts of this document, we will present it one section at a time. The first part of the file contains the "normal" part of our HTML file. Add this to your new file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head></head>
<BODY BGCOLOR=#FFFFCC TEXT=#333333 LINK="#0000CC" VLINK=#FF6666>
<h2 align=center>Report Form</h2>
```

6. Now enter the code to mark the beginning of the form. Until the next lesson when we learn how to make it actually do something, we will not be writing the **`ACTION=.`**.. part.

```
<form method="post">
```

7. The report form will be displayed using HTML tables, which are useful for web forms because they allow us to align the text labels and the form elements. We will create a table that has 4 major parts of our form:
    a. Student Information: information about the person filling in the form
    b. Volcano report: A summary of the information they are presenting
    c. Sources: A place for students to provide the web site resources they used
    d. Submit and Clear: The buttons to click to either send or reset the form

Each of these sections will be marked by a large table cell that defines that section of our form. Add this content to start the table and to write the first section for "Student Information":

```
<table border=0 cellpadding=4 cellspacing=1>
<tr>
<td colspan=2 bgcolor="#003333"><font size=+3 color=#CCCCCC>
<tt><b>student information</b></font></td>
</tr>
<tr>
<td valign=top align=right><tt><b>name</b></td>
<td valign=top><input type="text" name="name" size="40"><br>
<font size=2 color=#000099>enter your full name</font>
</td>
</tr>

<tr>
<td valign=top align=right><tt><b>email</b></td>
<td valign=top><input type="text" name="email" size="40"><br>
<font size=2 color=#000099>enter an internet contact address</font>
</td>
</tr>

<tr>
<td valign=top align=right><tt><b>password</b></td>
<td valign=top><input type="password" name="pass" size="20"><br>
<font size=2 color=#000099>enter a code to identify you to your instructor</font>
</td>
</tr>
```

**NOTE: We have created a 2 column cell for the text "student information" followed by three table rows that contain text input form fields. We have put the labels for the field in the first column of each row, right aligned. We also provide brief instructions in small, blue text below each form element.**

**Note how each form element has a unique name. The third form field is actually a password type to shield the code name the user will type in.**

8. Now we will write the second section where the person using this web form will provide their volcano report-- this one uses text input, menu selections, radio buttons, and checkboxes:

```
<tr>
<td valign=top align=right><tt><b>volcano name</b></td>
<td valign=top><input type="text" name="vname" size="40"><br>
<font size=2 color=#000099>enter the name of the volcano you researched</font>
</td>
</tr>
<tr>
<td valign=top align=right><tt><b>location</b></td>
<td valign=top><input type="text" name="vlat" size="10"> latitude<br>
<input type="text" name="vlong" size="10"> longitude<br>
<font size=2 color=#000099>enter the map coordinates that locate this volcano</font>
</td>
</tr>

<tr>
```

```
<td valign=top align=right><tt><b>type</b></td>
<td valign=top>
<select name="vtype">
<option value="select" selected>Select the type...</option>
<option value="Hawaiian">Hawaiian</option>
<option value="Surtseyan">Surtseyan</option>
<option value="Strombolian">Strombolian</option>
<option value="Phreato-Plinian">Phreato-Plinian</option>
<option value="Plinian">Plinian</option>
</select><br>
<font size=2 color=#000099>select the volcano type
(see <a href="term.html" target="_top">volcano terminology</a>)</font>
</td>
</tr>

<tr>
<td valign=top align=right><tt><b>activity</b></td>
<td valign=top>
<input type="radio" name="active" value="active" checked> active
<input type="radio" name="active" value="inactive"> inactive<br>
<input type="text" name="vdate" size="40"><br>
<font size=2 color=#000099>enter the date of last known eruption, if known</font>
</td>
</tr>

<tr>
<td valign=top align=right><tt><b>features</b></td>
<td valign=top>
<input type="checkbox" name="note1" value="danger risk">danger risk to nearby
population<br>
<input type="checkbox" name="note2" value="historic eruptions">has erupted in
historic time<br>
<input type="checkbox" name="note3" value="observed">has been observed in detail<br>
<input type="checkbox" name="note4" value="explosive eruptions">explosive
eruptions<br>
<input type="checkbox" name="note5" value="mild eruptions">mild eruptions<br>
<font size=2 color=#000099>check all that apply</font>
</td>
</tr>

<tr>
<td valign=top align=right><tt><b>more info</b></td>
<td valign=top><textarea name="info" rows="12" cols="40" wrap=virtual>
</textarea><br>
<font size=2 color=#000099>write any other information
that you have learned about this volcano</font>
</td>
</tr>
```

**NOTE: Be sure to compare the format of radio buttons versus check boxes; each radio button in a set has the same `name` while all of the check boxes have different names.**

9. The third section of the web page form is used for providing resources used in the report through provides three field input fields. Since the data entered will be web site addresses, we can provide an initial **`VALUE`** of "http://" for the text input tags:

```
<tr>
<td colspan=2 bgcolor="#003333"><font size=+3 color=#CCCCCC>
<tt><b>sources</b></font></td>
</tr>

<tr>
<td valign=top align=right><tt><b>references</b></td>
<td valign=top><input type="text" name="ref1" size="40" value="http://"><br>
<input type="text" name="ref2" size="40" value="http://"><br>
<input type="text" name="ref3" size="40" value="http://"><br>
<font size=2 color=#000099>provide three web site URLs that you used in your
research</font>
</td>
</tr>
```

10. The final section of our table/form contains the buttons that will submit the report content in the form and another button that can be used to reset the form to is initial, empty state. It also ends the table, the form, and the rest of the HTML document.

```
<tr>
<td colspan=2 bgcolor="#003333"><font size=+3 color=#CCCCCC>
<tt><b>send report</b></font></td>
</tr>

<tr>
<td valign=top align=right> </td>
<td valign=top>
<input type="submit" value="Send in My Report">
<input type="reset" value="Erase Report Form">
</td>
</table>
</form>

</body>
</html>
```

11. **Save** and **Load** the **`proj.html`** document in your web browser. The left side of this framed web page should now have a link that you added in step 2 above, that loads the web page form you created in the following steps. The form will not do anything yet, but the buttons, menus, and fields should all be editable.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. What is the structure of a web page form?
2. What are the differences in function and HTML coding for radio buttons and check boxes?
3. How do you create a drop down menu in a web page?
4. What are the differences between **Submit** and **Reset** form buttons?

# Independent Practice

Experiment with writing web page form elements to your own web pages. How would you create a web page that has more than one form on it?

---

# Coming Next....

Let's make that web page form do something!

**GO TO.... | Lesson Index | previous: "Forms: Intro" | next: "Form Action with email and CGI" |**

---

**Writing HTML:** Lesson 28a: Forming Forms
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut28a.html

# 28b. Form Action by email and CGI

*Enough layout! Let's see some form* action!

## Objectives

After this lesson you will be able to:

- Make a web page form work by sending content via e-mail
- Describe the content of a web form data submission
- Make a web page form that sends data via email in a readable format
- Make a web page form work by sending it to a CGI script
- Write the HTML for a hidden form element

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

In our previous lesson we created the HTML code to put form elements into our web page. Now we will make it do something.

Web forms were designed for viewers to send content from a web page to a special program on a web server, that would do something with it and return a message back to the person who sent it. This whole transaction takes place in a matter of seconds or even quicker! if you want to provide this type of interaction in a web page, it must communicate with these special programs, typically called **CGI** for **Common Gateway Interface**. For more information, see NSCA's overview of CGI.

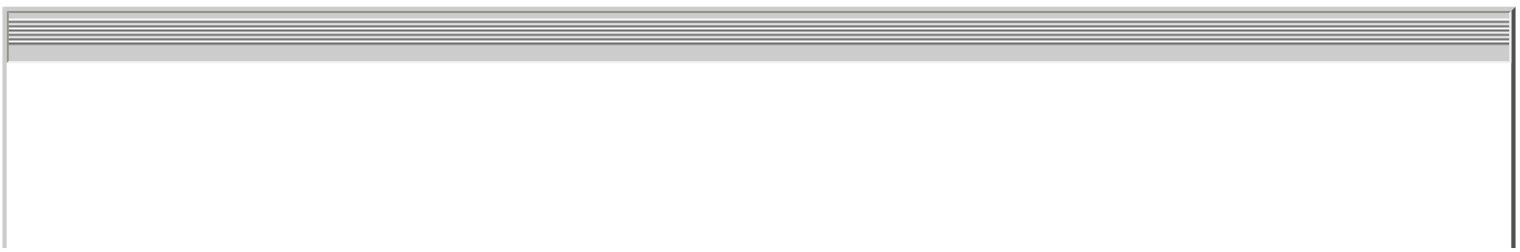
To have this functionality, you must write them in a complex programming language. Or, you may have access to a web server that provides these programs. Later in this lesson we will show you how to make forms work with such programs, but first we will show you an alternative way to get information from web page forms... by old fashioned e-mail.

You can add code to a web page form so that it emails the data that the form would normally send across the internet to a CGI program. To do this. modify the **FORM** tag you created in the last lesson to read:

```
<form method="post" action="mailto:me@myemail.bigu.edu">
```

We have here assigned the "action" of a form to take all of its data and send to whatever email address is provided.

But what does it look like? It's not very pretty! If we added this to the Report form we created in the last lesson, we would get an email message that contains something that looks like:



```
Date: Mon, 21 Dec 1998 15:44:18 -0700
From: alan levine <alan.levine@domail.maricopa.edu>
Subject: Form posted from Mozilla
To: me@myemail.bigu.edu
MIME-version: 1.0
X-Accept-Language: English, en

name=Alan+Levine&email=levine%40maricopa.edu&pass=ilovehtml&vname=Big+Volcano
&vlat=142+N&vlong=28+S&vtype=Phreato-Plinian&active=active&vdate=April+1%2C+1999
¬e1=danger+risk¬e3=observed&info=Big+Volcano+is+located+on+the+edge+of+a+
huge+mountain+range.+It+is+the+part+of+the+local+legends+of+the+original+inhabitants
+of+this+region.&ref1=http%3A%2F%2Fwww.abc.com&ref2=http%3A%2F%2F&ref3=http%3A%2F%2F
```

This example may give you an idea what a web form does with all of the text you write in and buttons you click on it-- it attaches it all in a long, single string of text. If you look closely, you can see that the format is:

```
element1_name=element1_value&element2_name=element2_value...
&elementN_name=elementN_value
```

so that each form element (fields, radio buttons, text area) sends its name connected by equal signs to its value, and they are strung together connected by "&" symbols. Furthermore, all of the blanks in the input are translated to "+" signs ("Alan Levine" entered in a text field becomes "Alan+Levine"), and other symbols such as ":", "/", are converted to things like "%3A" and "%2F".

This is done because it is a useful format for a CGI computer program to extract the content, evaluate it, and then do something in response.

While you could use this technique on your own web forms, the results are not very useful to work with. There is another option you can use, however, to improve the format in which form data is sent via email, by again modifying the **FORM** tag to read:

```
<form method="post" action="mailto:me@myemail.bigu.edu"  enctype="text/plain" >
```

The tag **enctype=** for **encoding type** instructs the web browser to send the form data not as form data like the example above, but to send it as a simple text listing. For example, adding this to our report form for the Volcano Web site, the email message we receive now looks like:

```
Date: Mon, 21 Dec 1998 15:44:18 -0700
From: alan levine <alan.levine@domail.maricopa.edu>
Subject: Form posted from Mozilla
To: me@myemail.bigu.edu
MIME-version: 1.0
X-Accept-Language: English, en

name=Alan Levine
email=alan.levine@domail.maricopa.edu
pass=ilovehtml
vname=Big Volcano
vlat=142 N
vlong=28 S
vtype=Phreato-Plinian
active=active
vdate=April 1, 1999
```

```
note1=danger risk
note2=historic eruptions
note3=observed
info=Big Volcano is dangerous! It has killed many people.
ref1=http://www.bigu.edu/volcanoes
ref2=http://www.usgs.gov/
ref3=http://www.volcano.nodak.edu/
```

which is now in a much more readable form.

> **NOTE: Although you can develop web forms that work by routing the information via e-mail, this approach is not very reliable for many people. It requires that the web browser is configured to send email through someone's account, so it may not work on say a web browser configured to be accessed in a public place. For more details, see the April 2000 newsletter from NetMechanic.**
>
> **There are a number of free sites that will host more reliable CGI mailing scripts for you, such as FormMail.To, FormMailer, Response-O-Matic, and others listed at The Free Site.**

Sending form data by email can be useful if you do not have access to CGI scripts or a web server, but it provides only limited interaction; the form data can be emailed to you, but the person that sends it gets no feedback from sending the form.

We will now modify the report form so that it is processed by a CGI script. Because not everyone has the capability to run these scripts, we have written it for you and you can run it from our web server.

This script will perform two functions for our Volcano Web Report Form. It gives the person using the form an option to have their report sent to their instructor by e-mail as well as an option to display the report as a web page (which could then be printed).

The form will have a new feature that allows us to embed in the HTML code another web form element (that is not displayed on the page) where we could also send the email address for the instructor, allowing different email addresses to be used on different web pages.

## Hidden Input Elements (`type="hidden"`)

... are used to send form data from the HTML code without it appearing in the layout of the web page.

**sample web page**

A hidden input element named "snack" with a value of "cheese puffs"

```
<input type="hidden" name="snack" value="cheese puffs">
```

As you can[not] see, the form element written here (trust us it really is here) is not displayed but contains data we can send with the form. In fact, you can use a small script to test the value of this hidden form element:

What is in the hidden element?

1. Open the **`proj_report.html`** file in your text editor.
2. Modify the **`<form>`** tag near the top of the document to read:

```
<form method="post" action="http://www.mcli.dist.maricopa.edu/cgi-bin/tut/report.pl">
```

**The `action` tag contains the URL for a CGI script on the MCLI web server that will do the tasks we have programmed into it.**

**This script is programmed in a langauge called Perl (Practial Extraction and Report Language), described by its creator Larry Wall as "an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It's also a good language for many system management tasks. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal)." Perl is one of the leading tools, but by no means the only way to program interaction from web forms.**

**For the purpose of this tutorial, and often for many web development tools, you do not even have to understand how it works to use it! If you are interested we do provide the perl source code, but you do not need it to do this lesson.**

3. Next we will modify the bottom part of the web form to provide the options for emailing or printing the report. Below the table row that contains the label for **send report** and above the table row that contains the form Submit and Reset buttons, add this new table row:

```
<tr>
<td valign=top align=right><tt><b>format</b></td>
<td valign=top>
<input type="checkbox" name="rep_email" value="y">
send to my instructor via email<br>
<input type="checkbox" name="rep_web" value="y" checked>
generate a web page for preview/printing<br>
<font size=2 color=#000099>
select options for processing your report</font>

<!-- change the value to have the report sent to a real address -->
<input type="hidden" name="instructor" value="lava@pele.bigu.edu">
</td>
</tr>
```

**We have added two new checkboxes; the first one tells the script to send the report by email and the second one (checked by default) will display the report as a web page. The last form tag is a hidden form element named "instructor". The value of this tag is the email address that the report will be sent to. If you want to see how this works, you should insert your own e-mail address into this tag.**

4. **Save** and **Load** the `proj.html` page in your web browser. Try the form now and see what happens when you check the different options. (What happens if none are checked?)

## More with Forms and CGI

While useful, the report form we created is a very basic example of what we can do with CGI scripts. We could have added features so that the submitted reports were written as files to the web server so that other people could see them, it could have checked the different input against a database of known information for volcanos, or many more tasks.

Writing CGI scripts is not overly complex but complex enough to be beyond the scope of this tutorial. Generally these scripts can be customized to do almost anything you can think of! But you need to have some knowledge of a scripting/programming language. You can find more resources from our reference page.

But for now, we will create one more page that uses a CGI script to perform a calculation. Again adding to our Volcano Web Project page, we will create a web page that has a tool for estimating the velocity of different volcanic flows.

1. First we have to modify the left frame of our Projects page to provide a new link. Open the **proj_menu.html** file in your text editor.
2. Above the link information created for the Report Form, add this HTML:

```
<a href="proj_calc.html">
<font size=+2 face="arial,helvetica">C</font>ALCULATION...</a><br>
tool for estimating volcanic flow velocities
<p>
```

3. **Save** this document
4. For our new page, we will need a graphic image that shows a diagram to present the concept for this calculation page. You can get the image from the Lesson 28b Image Studio and it should be saved inside the **pictures** folder/directory with your other image files.

5. Create a new HTML file in your text editor and save it as **proj_calc.html**
6. Write this HTML in this new document:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head></head>
<BODY BGCOLOR=#FFFFCC TEXT=#333333 LINK="#0000CC" VLINK=#FF6666>
<h2 align=center>Simple Calculations</h2>
<h3>Equation Kinetic and Potential Energies</h3>
To better understand the many kinds of volcanoes, we can use some
math and the laws of physics. In an eruption, you can track a
"block" of volcanic material from some point where all of its
energy is potential energy and equate it at some other point
where it is at a maximum of kinetic energy:
<p>
<center>
<img src="../pictures/hb_graph.gif" alt="energy diagram" width="342" height="199">
</center>
<p>
where <font color="red"><b>m</b></font> is the mass of the
"block", <font color="red"><b>g</b></font>  is the gravity
acceleration constant, <font color="red"><b>h</b></font> is the
height where all energy is potential energy, and <font
color="red"><b>v</b></font> is the velocity when the kinetic
energy is at a maximum.
<p>
Assuming conservation of energy, with some algebra we can write
this relationship as:
<P>
<center>
<font size=+3><tt> 2 g h = v<sup>2</sup></tt></font>
</center>
<p>
This means that if we know a height at which a volcanic flow
surmounted an obstacle, we can estimate its maximum velocity at
some point before or after the obstacle. This technique was used
to estimate the maximum flow velocity of a landslide in Iran that
climbed a 600 meter hill as well as a volcanic eruption in Japan
that climbed 500 meters over a mountain pass. The estimates are
supported by observations of these events.
```

```
<h3>Equation Kinetic and Potential Energies</h3>
Use the form below to calculate estimated maximum velocities for
volcanic eruptions where you can document how far they have
climbed (for large <a href="term.html" target="_top">Plinian</a>
eruptions, some researchers use the maximum height of the
eruption cloud).
<p>
<center>
<form method=post action="http://www.mcli.dist.maricopa.edu/cgi-bin/tut/energy.pl">
<table border=0 cellpadding=6 cellspacing=2>
<tr>
<td valign=top align=right><tt><b>maximum height</b></td>
<td valign=top><input type="text" name="height" size="10">
</td>
</tr>

<tr>
<td valign=top align=right><tt><b>units</b></td>
<td valign=top>
<input type="radio" name="units" value="meters" checked> meters
<input type="radio" name="units" value="feet"> feet
</td>
</tr>

<tr>
<td colspan=2 align=center><input type="submit" value="calculate velocity">
</td>
</table>

</form>
</center>
Note that this is a very generalized way to look at volcanic
eruptions; i.e. it does not account for losses of energy due to
friction nor the different mechanics for fluid flow. However, it
has proven to be useful to compare different volcanoes.
</body>
</html>
```

> **NOTE: Most of this document is HTML you should be familiar with from previous lessons, used to describe the calculation form at the bottom of the page. The `<form>` tag references another perl script we have written for you. The form's input is a text field for the number representing the height (H) and radio button options for its units of length.**
>
> **The CGI script checks the input (making sure it is a positive number) and returns an answer in the appropriate unit. (view CGI source code)**

7. **Save** and **Load** the **`proj.html`** page in your web browser. Try the new calculation form.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor.

# Review

Review topics for this lesson:

1. How can you make a form send its data to you by e-mail?
2. What does form data look like?
3. How can you make the form data mailed to you in a format that is easy to read?
4. How do you write a form tag to send the data to a CGI script?
5. What is the HTML form code for a hidden form element? How might you use this?

# Independent Practice

See if you can write a web form that has the same elements as our form but is designed in a different page layout-- can you get the form to work?

---

# Coming Next....

Web page / form interaction fueled by JavaScript.

**GO TO.... | Lesson Index | previous: "Forms: Intro" | next: "Form Action with JavaScript" |**

---

**Writing HTML:** Lesson 28b: Forming Forms
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut28b.html

# 28c. Form Action by JavaScript

## Can't do the CGI stuff?.. Do it with JavaScript

## Objectives

After this lesson you will be able to:

- Describe advantages and disadvantages of using JavaScript to process web forms
- Write JavaScript code to perform a mathematical calculation
- Describe how JavaScript and CGI scripts can work together in a form
- Write JavaScript code to create drop down navigation menus for our own documents
- Write JavaScript code to create drop down navigation tools that link to external sites

---

## Lesson

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

If you do not have access to a web server or the arcane knowledge for programming CGI, you can still create functional web forms with JavaScript. One advantage is that all of the "work" is done on the viewer's desktop ("client-based") rather than running from a web server-- so there is less back and forth communication across the Internet. Another is that you can create functional web pages that can still work even if you are not connected to the net. And also, JavaScript provides some extra features for checking the validity of entered data before it is processed.

However, there are things JavaScript cannot do-- it cannot write or store any data sent from a web form. Also, JavaScript code must be embedded into a web page, so complex scripts add to the file size (and download time) for your web pages. Also, because the scripts are part of a page, anyone can view the code by looking at the HTML source-- if you create a JavaScript quiz, somewhere in the code you must include the answers!

Finally, you may run into situations where visitors to your web pages have an older browser that does not support JavaScript.

So now we will modify our Volcano Web documents so that they do some form work with JavaScript. The first page is the form we worked on in the previous lesson that via a CGI script calculated an estimated velocity based upon an input height entered into a form field. The math is pretty simple, so this is something that can be easily done with JavaScript.

But rather than having our calculation display its results in a new web page like the CGI script does, we can create a form input text field and have JavaScript insert the calculated value into this field. By doing it this way, the person using the form stays on the same page and can try a series of different numbers.

1. Open the **`proj_calc.html`** file in your text editor.
2. After the table row that contains the radio buttons for units and above the table row that contains the submit button, add this new table row:

```
<tr>
<td valign=top align=right><tt><b>result</b></td>
<td valign=top><input type="text" name="result" size="20">
</td>
</tr>
```

3. Now we will write a new JavaScript function that will be placed inside the **<HEAD>...</HEAD>** like we did in lesson 27c. So inside the **<HEAD>...</HEAD>** tags add this JavaScript code:

```
<script language="JavaScript">
<!--
/* Velocity Calculation
   Calculates a maximum theoretical velocity based upon an
   input height by equating potential and kinetic energies  */

function calc_vel (form) {
// Check the input for good values
        if ( isNaN(form.height.value) || (form.height.value <= 0) ) {
                alert('The entered height \''+ form.height.value +
                        '\' is not valid. To complete the calculation, ' +
                        'the height must be a number greater than 0.');
                form.result.value = '';
        } else {

// Assume metric first
                var gravity = 9.8;
                var myUnits = 'meters';

// If feet were checked, update the units and gravity constant
                if (form.units[1].checked) {
                        gravity = 32.0;
                        myUnits = 'feet';
                }

// Calculate velocity and put results in display field
                var velocity = parseInt(Math.sqrt( 2 * gravity * form.height.value));
                form.result.value = velocity + ' ' + myUnits + '/second';
        }

// return FALSE value so form does not call CGI
        return false;
}
//-->
</script>
```

We have introduced some things you have not seen before! You do not have to understand them to cut and paste this code, but we will give an overview of what it actually does.

The first part of the code uses a multi-line comment marker, **/* ... */** for the description of the script. Our function called "calc_vel" will be sent some information via a parameter (the thing inside the parentheses) called "form"-- this is going to be a JavaScript data structure for all of the things selected or entered in our web page form.

The first thing our script does is to make sure the numbered entered is a good value. So we do a test to see if it is a non-numerical using the **NaN** JavaScript built-in function for testing if something is "Not a Number" and we test to make sure the number is greater than 0. If the input represented by the value of the height input data sent to us fails either test, we generate an alert message.

Otherwise (after the **`else`**) we proceed with the calculation, first assuming the input is in metric units. We then check the status of the radio buttons, and adjust these values if the non-metric units were selected. Then, the script uses more JavaScript built-in function to calculate the answer, returning it in an integer format. We then can put this result into the value of the form field we create in step 2.

The function returns a value of "false" to whatever it was that called this function. In the next step we will see what this means.

The last thing we need to do is to edit our **`<form>`** tag so that it sends a request to this new JavaScript function.

1. Edit the **`<form>`** tag to read:

```
<form method=post
  action="http://www.mcli.dist.maricopa.edu/cgi-bin/tut/energy.pl"
  name="energy" onSubmit="return calc_vel(this)">
```

The **`onSubmit`** is a new option for this tag that performs what ever JavaScript is in its quotes when the Submit button for the form is clicked. It does this *before* making a call to the CGI script in the **`ACTION=`** attribute. If the result of **`calc_vel(this)`** is true, then the CGI is run; if it is false, then the CGI script is not called. This is the feature that allows you to perform JavaScript before a form is sent off to a CGI script, often so that you can use JavaScript to verify the input data.

Another advantage of combining your code like this is that if for some reason a viewer is using a web browser where JavaScript is not active, it ignores all of the **`onSubmit`** code and sends the data to the web server CGI to process.

Sending the **`calc_vel`** the parameter **`this`**, means "send this function all of the data in my form, the names and values of all of my form elements".

2. **Save** and **Load** the **`proj.html`** page in your web browser. Try the new calculation form. If all goes well, it should display the calculation results right into the empty field of the web form. Test what happens if you provide a non-numerical or negative height.

## JavaScript Navigation Menus

The next thing we will do with JavaScript is to make it easier to navigate among the pages of our site.

Until now, we have created hypertext links at the top of every page that allow the viewer to go forward, backward one web page in our series, or to return to the index page. We can use web page forms to replace this with drop down menus that permit a viewer to move to any page in our site. This is a very valuable feature in a complex web site, and it reduces the number of pages viewers must navigate to see your content. Drop-down menus also collapse the navigation information into a compact display (compare the space taken up by 25 hypertext links to one drop down menu).

Our approach is to write a general JavaScript function that can go into every document that uses the navigation menus, and then make some minor adjustments to each one.

1. So inside the **`<HEAD>...</HEAD>`** tags add this JavaScript code:

```
function goPage (newURL) {
// This function is called from the pop-up menus to transfer to
// a different page. Ignore the value returned is a null string

        if (newURL != "") {

// skip the menu dividers and reset the menu selection to default
                if (newURL == "-" ) {
```

```
                        resetMenu();
                } else {
// send page to designated URL
                        document.location.href = newURL;
                }
        }
}

function resetMenu() {
// resets the menu selection upon entry to this page
   document.gomenu.selector.selectedIndex = 2;
}
```

These functions perform three different tasks. If the value sent to **`function goPage`** is blank **`(newURL="")`**, we do nothing. This is the case if the person selected the page currently in view. The second possibility is that the value is "-", which we will use to indicate a "divider" line in our menus, in which case we will then call a second function, **`resetMenu()`** that resets the menu to its default state (in this case, selecting the third item in the menu-- Javascript starts counting things from 0.). And the third case is the one where some action really takes place, transferring the document to the value of the URL.

Let's start with our **`intro.html`** file, which already has JavaScript code, so we can just copy the two functions and paste it anywhere before the end of the ending JavaScript tag.

2. Now we will create the menu. Replace the part of the document that looks like:

```
<h5>Volcano Web /
<a href="index1.html">Index</a> /
<a href="term.html">next</a></h5>
```

to read:

```
<form name="gomenu">
<h5>Volcano Web /
<select onChange="goPage(this.options[this.selectedIndex].value)" name="selector">
<option value = "index1.html">Volcano Web</option>
<option value = "-"> --------------</option>
<option value = "" selected>Introduction</option>
<option value = "term.html">Volcano Terminology</option>
<option value = "usa.html">Volcanic Places in the USA</option>
<option value = "mars.html">Volcanic Places on Mars</option>
<option value = "proj.html">Research Project</option>
</select>
<noscript>
<a href="index.html">Index</a> /
<a href="term.html">next</a>
</noscript>
</h5>
</form>
```

We have inserted a form named "gomenu" that contains a drop down menu named "selector". The **`onChange`** Javascript event is called whenever the menu selection is changed, and if so, it calls the function **`goPage`** and sends it the value of whatever is in the **`value`** portion of the menu item that corresponds to the selection.

The menu item that corresponds to this page ("introduction") will be selected when the page loads by the **`selected`** keyword in the **`option`** tag. Also note that the value for this tag is empty, or "", meaning that if this menu item were to be chosen, our JavaScript

function will know not to change anything. Finally, we have used a line of dashes below the first item as a menu divider; if this item is selected our JavaScript function calls a second function called **`resetMenu`** that simply restores the menu to its initial selection (because we do not want to take any action if the viewer selects the dividing line.

The HTML we put in the **`<noscript>...</noscript>`** tags displays our original HTML links in case the viewer is running a web browser that does not support JavaScript.

Now we will add one more small feature to make our menu fully operational. This piece of code will make sure the menu item corresponding to this page will be reset if the viewer should use the menu to navigate to another page and then use the browser **back** button to return to this page. Without this feature, the menu would load with the last chosen menu item selected.

3. Change the **`<body>`** tag to read:

```
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666 onLoad="resetMenu()">
```

   **The `onLoad` Javascript event is called every time the web page is read into the browser, so that it calls our menu resetting function every time the Introduction page loads.**

4. **Save** and **Reload** in your web browser. Test to see that the JavaScript navigation menu works to send you to any of the other web pages it lists.

Now to make our menu navigation complete, you will modify the links in a similar fashion in the other main documents of our Volcano web site, copying the code from steps 1,2, and 3 above. There are a few subtle differences that you will have to make for each one, as summarized in this chart:

# JavaScript Menu Edits

## Volcano Terminology
**`term.html`**

a. This page has no JavaScript code in it, so you will have to insert the script tags

```
<SCRIPT LANGUAGE="JavaScript">
<!--
```

and

```
//-->
</SCRIPT>
```

around the code in step 1

b. function **`resetMenu()`** reads:

```
document.gomenu.selector.selectedIndex = 3;
```

c. These lines in the form menu (step 3 above) should read:

```
<option value = "intro.html">Introduction</option>
<option value = "" selected>Volcano Terminology</option>
```

## Volcanic Places in the USA
**usa.html**

a. The code in step 1 can be inserted with the other JavaScript code in the **HEAD** of this document.

b. function **resetMenu()** reads:

```
document.gomenu.selector.selectedIndex = 4;
```

c. These lines in the form menu (step 3 above) should read:

```
<option value = "intro.html">Introduction</option>
<option value = "" selected>Volcanic Places in the USA</option>
```

## Volcanic Places on Mars
**mars.html**

a. This page has no JavaScript code in it, so you will have to insert the script tags

```
<SCRIPT LANGUAGE="JavaScript">
<!--
```

and

```
//-->
</SCRIPT>
```

around the code in step 1

b. function **resetMenu()** reads:

```
document.gomenu.selector.selectedIndex = 5;
```

c. These lines in the form menu (step 3 above) should read:

```
<option value = "intro.html">Introduction</option>
<option value = "" selected>Volcanic Places on Mars</option>
```

## Research Project
(navigation document of this framed page)
**proj_nav.html**

a. This page has no JavaScript code in it, so you will have to insert the script tags

```
<SCRIPT LANGUAGE="JavaScript">
<!--
```

and

```
//-->
</SCRIPT>
```

around the code in step 1

b. Change the line in the **`function goPage (newURL)`** function that reads:

```
document.location.href = newURL;
```

to read:

```
parent.document.location.href = newURL;
```

which is what needs to be done to make the script work in a framed web page.

c. function **`resetMenu()`** reads:

```
document.gomenu.selector.selectedIndex = 6;
```

d. These lines in the form menu (step 3 above) should read:

```
<option value = "intro.html">Introduction</option>
<option value = "" selected>Research Project</option>
```

That was a lot of work!

If all went well, your main web pages should now all be connected by a menu navigation tool that now allows a visitor to your site to jump immediately from one page to the other without having to click through a series of pages in between.

# One More JavaScript-Powered Form

Now we will combine a bit of what we have done to create a JavaScript tool to navigate to pages at another web site. We are taking advantage of the well-designed web site structure of the Views of the Solar System site that offers content information on all of the planets that is written in three different languages. By examining the URLs for this site, we can see that they look like:

`http://solarviews.com/`*`language`*`/`*`planet`*`.htm`

where *language* is:

1. **`eng`** for English
2. **`span`** for Spanish
3. **`portug`** for Portuguese

and *planet* is simply the name of the planet (e.g. "mars", "jupiter")

Knowing this, we can create a web form where the viewer can select a planet from a drop down menu, and a language from a set of radio buttons, to view content from this site. This creates a simpler and more compact navigation scheme than a list of hypertext links.

1. Open the **mars.html** file in your text editor.
2. Inside the JavaScript code you created for the navigation menu from the last section, add this new JavaScript function:

```
function goPlanet () {
// Function for navigation to different parts of the
// Views of the Solar System site

// get the planet selected from the menu
        var planet =
document.solar.planets[document.solar.planets.selectedIndex].value;

// make sure valid entry is selected
        if (planet == "") {
                alert ('Please select a planet!');
        } else {

// determine which language button is selected
                for (i=0; i<3; i++) {
                        if (document.solar.lang[i].checked) {
                                lang = document.solar.lang[i].value;
                                break;
                        }
                }

// construct the URL for the off-site link
                var url = 'http://solarviews.com/' + lang + '/' + planet + '.htm';

// open the URL in a new window
                var planet_window = window.open( url , "planets",
"toolbar,status,location,menubar,scrollbars,resizable,width=550,height=450");

// If we are on NetScape, we can bring the window to the front
                if (navigator.appName.substring(0,8) == "Netscape")
planet_window.focus();
        }
}
```

3. In the BODY of this HTML document, after the one sentence about Olympus Mons, add this HTML and web form:

```
<p>Compare the volcanic landforms on Mars with the other planets<br>
<form name="solar">
<center>
<table border=0 cellpadding=10 cellspacing=2>
<tr>
<td valign=top><select name="planets">
<option value = "" selected>Select a Planet...</option>
<option value = "mercury">Mercury</option>
<option value = "venus">Venus</option>
<option value = "earth">Earth</option>
<option value = "mars">Mars</option>
```

```
<option value = "jupiter">Jupiter</option>
<option value = "saturn">Saturn</option>
<option value = "uranus">Uranus</option>
<option value = "neptune">Neptune</option>
<option value = "pluto">Pluto</option>
</select>
</td>

<td valign=top>Show the information in:<br>
<input type="radio" name="lang" value="eng" checked>English<br>
<input type="radio" name="lang" value="span">Spanish<br>
<input type="radio" name="lang" value="portug">Portuguese
</td>
<td valign=bottom><input type="button" value="Show Info"
    onClick="goPlanet()"></td>
</tr>
</table>
</center>
</form>
```

**We have created a new web form that contains a drop down menu with the names of the planets and radio buttons to choose the language to display the content. Our JavaScript function simply takes the form elements as selected, and constructs a proper URL for the external web site that contains this information. As a bonus, it opens this in a new JavaScript window, as we learned in lesson 27c.**

4. **Save** and **Reload** in your web browser.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. This lesson has presented a large amount of JavaScript to cut and paste, and it is very easy to make a typographical mistake in this process.

# Review

Review topics for this lesson:

1. What are some advantages about using JavaScript to make your web page forms work? What are some problems in using JavaScript for forms?
2. Describe an approach for having a JavaScript enabled form that performs a calculation of the average of a series fo numbers that would be entered in a web form.
3. How can JavaScript and CGI scripts work together in a web form?
4. What are the essential parts to create your own JavaScript navigation menus?
5. What is needed to create a JavaScript navigation tool that links to external web sites?

# Independent Practice

Try changing the navigation links of your own web pages so that they use JavaScript navigation menus.

This is but a small sampling of what you can do with JavaScript. There are numerous web sites that offer JavaScript code that you can freely copy and use, and you do not even have to understand how it all works (though it helps to know!). Visit some of these sites, and try to find a code sample you can include in your own web pages:

- JavaScript Cut n' Paste
  http://www.infohiway.com/javascript/
- JavaScript Source
  http://javascript.internet.com/
- JavaScript City
  http://www.javascriptcity.com/
- JavaScript World
  http://www.jsworld.com/
- Builder.com Spotlight on JavaScript
  http://builder.cnet.com/Programming/JsSpotlight/

Also, we have created another tutorial/resource called the jClicker, a template that shows you how to easily create a JavaScript slideshow.

---

# Coming Next....

Adding sound, video, animation to your web pages.

**GO TO.... | Lesson Index | previous: "Forms: Form Action with email and CGI" | next: "Multimedia in a Page " |**

---

**Writing HTML:** Lesson 28c: Form Action with JavaScript
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut28c.html

# 29. Multimedia In Your Page

## "Lights... Camera... Action!"

plus animation, plus sound, plus interaction, plus java, plus....

## Objectives

After this lesson you will be able to:

- Describe the factors to consider creating/acquiring multimedia
- Describe the considerations for visitors to your web site if it includes multimedia

---

## Lesson

In the beginning (1993!)... there were only a few web pages... mostly text with a few pictures. Now there is quite a bit more you can load up onto a web page to appeal to the senses. You can now have media types such as motion, animated cartoons, video clips, sounds, interactive applications (no smell plug-ins yet). And some web pages will use all of these, several times over, in a single page, so that it may be unbearable to look at (or wait to download). If you really want to taste a few, see the WebHound's Good, Bad, and Bandwidth-Hogging Ugly.

In the next several lessons, we will review the basics of adding a few different types of media to your Volcano web site. This will not cover every possible media type available, just the ones that may be most accessible or useful to you. Also, we can not cover the basics of how you create the media (that would be several years of tutorial writing!) but where possible, we will try to identify sources where you can find existing media elements that are free to use.

Before using any type of media, you should carefully consider what value it adds to your content. Sure, the first few dancing animated gif icons you saw may be cute or eye catching, but they can also be distracting, and after a while, cliché. So your first question is, **"Does this [sound, video, applet, animation] add important information to my site?"** The converse to this question is, **"Is this [sound, video, applet, animation] just [window dressing, eye candy, etc...]?"**

When you decide to add a media element to your web site, a first cost to consider for you is **"What will**

**it take for me to create the [video clip, sound file, Java applet]?** You may need special equipment, software, expertise, and above all time. Fortunately, you have some alternatives- you can identify and use free libraries of media elements, or "clip media".

For every media element you use, there are also "costs" to consider for a visitor to your site. You should ask yourself, **"Does the media element require some special functionality (a plug-in, an Active-X control, a different/newer web browser)"** By the time a visitor to your site download, installs and sometimes restarts their computer, they may have forgotten all about your site!

The second cost in using multimedia in your web page is time. Ask yourself, **"How long it may take the content to download, or at least start revealing itself?"** The longer the delay for visitors to your site, the least likely they will stick around or come back. Yes, the music may play, the icons animate, and the video will roll when you test it from your desktop computer, but that will not be the experience someone else has when they are connected to the Internet via an older modem with last year's web browser.

This boils down to making some guesswork assumptions about who the people are that will come to your web site. If you know that it will only be viewed internally on your company's fast network, you can more comfortably use bandwidth intensive media. If you know that your content will be viewed in training center with **Version 4.9xgi of the SpiffGo Web browser**, you can use content that relies on special plug-in supported by that browser.

But more likely, you want the **world** to visit your site (after all it is the first "w" in the "www"), and you will have little idea how they are connected to the net, what browsers they are using, etc. Our advice is to be more judicious (and creative) in your use of multimedia and provide alternatives where appropriate.

So in the next lessons we will provide examples of adding GIF animations, video clips, sound files, shockwave interaction, Flash, and Java to your web sites. For each media type, we provide some media selection guidance in the form of a chart as shown below.

| media type: | *(name of media type)* | | |
|---|---|---|---|
| what it does well: | *(types of information it works best with)* | | |
| **issue to consider** | **rating** | | **comments** |
| "hurdle" or barrier for creating media | low | high | what it takes to create or acquire the media type |
| "breadth" of audience that can view media | narrow | wide | relative range of audiences that can view the media type |

| | | |
|---|---|---|
| bandwidth consumption | low high | time and file size factors needed to view the media type |

These ratings are merely a guide to help you consider different media types; in fact they are fairly subjective based upon our experience in using them all. Again the most important considerations are on the value of the media type to the content/message you hope to deliver.

One of the ways you can design your pages that contain multimedia is to make them as optional links from your main pages. This means that rather than inserting a 5 Mb video clip into your page, you create a link from that page to the video clip. This way, your site visitor decides if they want to view the clip. We will do this for most of our examples, which we will create as links that display the media in a separate browser window, using the JavaScript techniques we learned in lesson 28c.

# Check Your Work

This is an introduction, so we have not made any changes to our project.

# Review

Review topics for this lesson:

1. What are the "costs" to you for adding multimedia to your web pages?
2. What are the "costs" to a visitor to your web site if it requires a special plug-in to view a media type?

# More Information

Here are some more resources that you may find useful:

- **Web Developers Virtual Library** Multimedia for the Web
  http://wdvl.com/Multimedia/
- **Multimedia Authoring Web**
  http://www.mcli.dist.maricopa.edu/authoring/

---

# Coming Next....

Multimedia Element number 1... Add a GIF that does more than just sit there!

**GO TO.... | Lesson Index | previous: "JavaScript: Window Openers" | next: "Multimedia: Animated GIFs" |**

---

**Writing HTML:** Lesson 29: Multimedia in Your Page
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut29.html

# 29a. Animate my GIF!

It's a GIF!
It Moves!
It's an Animated GIF!

do we have your attention now?

can you get it back?

## Objectives

After this lesson you will be able to:

- Identify types of information that are appropriate for GIF animations
- Describe the difference between a GIF image file and an animated GIF image file
- Write the HTML to insert an animated GIF into your web page
- Write JavaScript code to close a window

---

## Lesson

We doubt we have to describe what animated GIF image looks like, as you can find them all over the web. These range from animated icons, to commercial advertisements ("ad banners"), to web site "credits", first pages of web sites that look like the opening screens to a movie. GIF animations are useful because you can count on them being able to be viewed on almost every web browser and they do not require any special add-ons to work.

| media type: | animated GIF | |
|---|---|---|
| what it does well: | time-based information, sequences, processes, cartoon style/flip card style animation | |
| **issue to consider** | **rating** | **comments** |
| "hurdle" or barrier for creating media | low high | Most newer graphics programs as well as freeware/shareware can create GIF images. Many sites offer free collections of animations you can use |
| "breadth" of audience that can view media | narrow wide | GIF animations can be viewed in nearly every graphical web browser in use today; no plug-ins needed |
| bandwidth consumption | low high | can be quite small file sizes and the data can "pseudo stream" as they play. |

An animated GIF image is just an extension of the same file format used for inline GIF images we saw in lesson 7. But rather than just showing one image, an animated GIF has layers of different images, and information in the file can control how long it waits before displaying the next image, and how many times to loop through the entire sequence.

Several current graphics programs should have options to create animated GIF files. Plus, you can find free or low cost shareware programs (e.g. http://shareware.com/) that can turn a series of images into a GIF animation. And since the web is fully of dancing, blinking, and moving GIFs, there are plenty of free clip-art sites where you can download animations to use for your own pages. We provide a list of links to these sites at the end of this lesson.

Another advantage of this media type is that the entire graphics file does not have to be downloaded before it starts to play; an animated GIF will commence as soon as enough information is downloaded to display the first "frame" of the animation. Therefore, it appears to "stream" the animation and you can create animation files that are several hundred k in size that may start playing when perhaps 20% of the file has been received be the viewer.

What are some down sides of animated GIFs? The primary one is that because the movement attracts your eye, they can be distracting to someone trying to comprehend the other information on the same page. The other disadvantage so that they generally tend to be used as a novelty, and after the first few times you see the same animated teddy bear. it can start to look cliché or juvenile.

Animations can be very useful to show time-ordered information, i.e. to show changes in a feature over time (e.g. erosion of a beach, growth of a plant, changes in stock market prices) or to demonstrate a process (e.g. how to change a tire, how blood moves through the heart). Like GIF images, the files are much smaller for graphics that have large continuous regions of solid color and sharp edges, as compared to images with more spatial variation such as photographs.

## HTML for Inserting an Animated GIF

This should be an easy lesson since the HTML code for inserting an animated GIF is exactly the same as we learned for inserting a "regular" ("un-animated"?) GIF that we saw in lesson 7a:

```
<img src="filename.gif" alt="alternative text" width=X height=Y>
```

where **X** is the width of the image and **Y** is the height of the image in pixels. You can also include the left and right alignment options we saw in lesson 20.

## Adding a GIF-animated Slide Show

**Note: If you do not have the working documents from the previous lessons, download a copy now.**

For our *Volcano Web* site we are going to add an animated GIF slide show that reviews the events that led to the May 18, 1980 eruption of Mount St Helens. This consists of photographs, and diagrams, used courtesy of the USGS/Cascades Volcano Observatory.

To view and download the GIF animation, visit the Lesson 29a Image Studio. Save this image inside of your **`pictures`** folder/directory with your other image files.

We are going to first create a basic HTML page that displays the GIF animation. Then we will build a link from our Volcanic Places in the USA page that will load this animation in a separate browser window.

1. Use your text editor to create a new HTML file called **msh_may18.html** in your workspace directory/folder.
2. Enter the following HTML in this file:

```
<html>
<head>
<title>Mount St Helens: Events of May 18, 1980</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
<center>
<img src="../pictures/msh_events.gif" alt="Mount St Helens Animation"
vspace=10 width="400" height="135"><br>
<font face="verdana,helvetica,arial" size=2 color=#999999>
slide show will load and play automatically</font><p>
<i>Courtesy of the
<a href="http://vulcan.wr.usgs.gov/">USGS/Cascades Volcano Observatory</a>.</i>
</center>
</body>
</html>
```

> **NOTE: Since the GIF animation may stream in slower "chunks" via a slow internet connection, we provide a message in the small text below the image so a visitor to this page knows to wait while the animation loads and runs.**

3. **Save** and **Reload** in your web browser. This should be a fairly simple page with the animated GIF slide show in the center.

Now we will modify the HTML in the Volcanic Places in the USA file **usa.html** so that it uses JavaScript to display our new document in a smaller web browser window.

1. Use your text editor to open the file **usa.html**
2. After the last sentence in the Mount St. Helens section ("...toppled over like toothpicks.") add this HTML:

```
<p>
<a href="msh_may18.html" onClick="window.open('msh_may18.html',
'msh', 'width=440,height=280,status,menubar'); return false"
onMouseOver="window.status='view animation of eruption events';
return true"><img src="../pictures/projector.gif" alt="showtime!"
width="46" height="32" align="left" border=0></a>
See the <a href="msh_may18.html" onClick="window.open('msh_may18.html',
'msh', 'width=440,height=280,status,menubar'); return false"
onMouseOver="window.status='view animation of eruption events';
return true">animation of events</a> for this volcanic eruption
[194k GIF Animation].
```

> **NOTE: We are using JavaScript code from lesson 27d to open a specific HTML file msh_may18.html in a new web browser window that is 440 pixels wide and 280 pixels high. We have also included the JavaScript mouseOver code to provide a descriptive text a viewer moves the mouse over the link (see lesson 27a). The slideshow page is set to open by clicking either the icon of the projector or the hypertext link that is adjacent to the icon.**

**Note also that our link information gives the site visitor a sense of what kind of media (and file size) will be included on the page if they follow the link.**

3. **Save** and **Reload** in your web browser. If all went well, when you click the text **`animation of events`** from the Volcanic Places in the USA page, the slide show should play in a new browser window.

Now you may realize that as we continue adding features that open new windows via JavaScript, some visitors to your web site may not be sure how they can easily return to the main page, as they tend to leave an array of windows that clutter up or completely cover their desktop. We can help out a little bit by adding a JavaScript button to close a window.

1. Open the file **`msh_may18.html`** in your text editor.
2. After the line with the link to the Cascades Volcanic Observatory (and above the **`</center>`** tag) insert this HTML:

```
<form>
<input type=button value="Return to Volcanic Places in the USA"
onClick="self.close()">
</form>
```

**NOTE: We've added a new `FORM` element, `type=button` which inserts a normal interface style button, with the label of the text in the `value=...` option. We add a JavaScript event to happen when the button is clicked that tells the window this document reside in (`self`) to close itself. In effect, we have created a custom button that will close this window just as if the viewer had clicked the standard window close button from the titlebar.**

3. **Save** and **Reload** in your web browser.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Compare your work to the HTML of the samples (look for something like **Source** from your browser's **View** menu).

# Review

Review topics for this lesson:

1. What kind of information is good to use to present as a GIF animation?
2. What is the difference between a GIF animation file and a GIF image file?
3. What is the HTML code to place a GIF animation in a web page?
4. How do you write JavaScript code to close a browser window?

# Independent Practice

Add an animated GIF image to your own web pages. Try some of these resources to find images you can use:

- **Animation City**

http://www.animationcity.net/

- **Cool Archive**
  http://www.coolarchive.com/gifanimations.cfm
- **The Free Site**
  http://www.thefreesite.com/freegraphics.htm
- **XOOM Free Clip Art- Animated GIFs**
  http://www.xoom.com/clips/website/animated-gifs
- **Absolute Animation**
  http://www.altwebmasters.com/aag/
- **A+ Art Free ClipArt**
  http://www.aplusart.com/
- **Gifart.com**
  http://www.gifart.com

# More Information

If you tire of viewing animations on a web page, you can freeze them by clicking the **Stop** button in your web browser. When you create your own GIF animation files, you have some options to control how many times it will loop through an animation sequence, so can prevent it from endlessly spinning, dancing, etc.

And like static GIFs, you can make them act as hyperlinks like we learned in lesson 8e. If we found on the Cheeselover's Free Clip art site an animated gif of a dancing block of cheese, we could create a hypertext link to another site or HTML file by writing:

```
<a href="http://www.cheeseanonymous.com/"><img src="dancing_cheese.gif"
width=200 height=100 alt="animation of dancing cheese" border=0></a>
```

For more information about GIF Animation, see Royal E. Frazier's GIF Animation on the WWW as well as the Web Developer's Virtual Library

---

# Coming Next....

Viva Video! Movies in your web page!

**GO TO.... | Lesson Index | previous: "Multimedia in a Page" | next: "Multimedia: Movie Time" |**

---

**Writing HTML:** Lesson 29a: Animate my GIF!
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut29a.html

# 29b. Movie Time

*Adding Movies to Your Web Page*

please pass the popcorn!

## Objectives

After this lesson you will be able to:

- Identify the types of content appropriate for using video
- Describe the issues related to using video over the web
- Write the HTML to insert a digital video file into your web page

---

## Lesson

Blame it on television, Hollywood, MTV, etc, but we are a society tuned to video. And with the web, we can also include information in a video format.

But just because you *can* does not mean that you *should*. Why not? Very simply, video is a lot, a lot, of data to send across the internet, and likely your experiences to date are those "tiny" squares of herky jerky video segments where the voices seem horribly out of synch with the action. If you think about it, to make a "moving picture" means sending a lot of still images (like the frames of old movie film) to create the sense of movement, not to mention adding a layer of audio information.

So until high speed internet access is the norm (and it might be some day), we use technology to compensate via some clever techniques that "compress" video to make the file sizes smaller, and other techniques that allow the video information to "stream" to us (meaning we see content as it is sent, not having to wait for a large file to download.

| media type: | digital video | |
|---|---|---|
| what it does well: | time-based information, sequences, historic events, places impossible or unsafe to experience | |
| **issue to consider** | **rating** | **comments** |

| | | |
|---|---|---|
| "hurdle" or barrier for creating media | low high | Special equipment needed to take video source material and convert it to digital computer files. There are low cost solutions, but they still call for special hardware, a lot of disk space, and video editing software. |
| "breadth" of audience that can view media | narrow wide | To view video, you typically need a web browser "plug-in", but most newer browsers come with one already installed. |
| bandwidth consumption | low high | Can be quite large. Video represents a tremendous amount of information and is reduced quality over modem speeds. Options exist to "stream" content but it may require special servers. |

We suggest you try to think about the use of video without comparing it to television or the movies. The worst uses are what we call "Talking heads", i.e. a person sitting still talking to you. The video adds very little, and as we see later, you can send the equivalent information using less bandwidth intensive media by combing still images and audio.

When might it be crucial to use video in your web page? Most likely will be to show something that is changes over time, like we saw in animation, but perhaps needs to have a more "real-world" look over as compared to graphics. Often it might be a historic event (the assassination of John F Kennedy). Video may be important to show us something that we cannot see easily because of location (landing on the Moon, the Mars rovers) or physical limitations/ safety considerations (deep ocean exploration, inside a nuclear reactor). Or video may be critical to demonstrating a procedure (surgery, welding a steel beam).

You can use a "digitized" movie simply to play back like video, non-stop, from start to finish. But you can also use a sliding "control" to quickly jump to any point in the movie sequence. This can allow you to explore "snapshots" of a time-based process.

Let's shift and talk about video technology. Traditionally, movies/films were recorded via a photographic process (think about movie reels) and played back through projector. To use video on a computer, all of the information must be **digitized** or turned into a data file, essentially a series of still images that when played back at an appropriate speed, looks to the human eye like actual motion. Generally, this is achieved when we see about 30 images every second (or "frames per second").

To create a video clip you must have some special computer hardware to take a video input (from a source such as a VCR, a television output, a video camera) to do the magic of making a video file. While several years ago this was considered high end equipment, there are many low end (i.e. cheap) tools to do this. You would also likely need some special software to edit video clips once they are on your computer, which again range from very cheap (simple) to expensive (complex, professional).

When we start talking about digital video, we enter a region of acronym soup for the various kinds of video files. Probably the most important and commonly used formats are Apple's QuickTime and Moving Picture Experts Group (MPEG). There are of course, many others, and we cannot really review all of them here. For more information visit the About.com guide to Desktop Video. For this tutorial, we will present the code to embed a QuickTime movie into your web page. While other video formats may be used (e.g. MPEG, AVI) QuickTime is currently the most (and architecturally robust) format that can be used by the widest range of web browsers.

For more information on video, see the Web Developer's Virtual Library.

When the web first emerged, video clips were rarely used because, to watch them, you had to download the entire video file (which can be several megabytes for even short segments) before you could see anything. Later as web technology has evolved, solutions have emerged that allow video to be "streamed" to you, meaning when ask for video (clicking on a web page link), you will start to see the video information as soon as enough has been sent to show you a portion, and the rest continues to "stream" in as you watch the beginning parts. The leading technology here is RealNetworks; the limitation for you, the humble web page developer, is that RealVideo requires specialized servers to transmit streaming data. You can find great examples of RealVideo from the CNN news site.

It also takes some special web browser "add-ons" from the viewers end to see digital video. RealVideo and QuickTime require browser "plug-ins" to work. Fortunately, most web browsers now come with these technologies already included, but sometimes they may be updated. You should not, however count on everyone being ready to view video content!

Because video is a high bandwidth media format, as a design principle you should place video at a "click away", meaning that you present video as an option for your web site visitor to click to view, with some advance notice. We will demonstrate this in our lesson.

To include a digital video clip in a web page, we use the **<EMBED>** tag, which in some ways is similar to the **<IMG)** tag in that we specify a file source and the dimensions of the media we want inserted into the page:

```
<EMBED src="video.mov" width="320" height="240"></embed>
```

There are other options to add to this tag to control the playback:

| embed option | values | notes |
|---|---|---|
| **src** | path to movie file name | can be relative or full URL |
| **width height** | dimensions of movie file | |
| **autoplay** | true/false | whether the movie should automatically start playing or wait for user to click the play button |
| **controller** | true/false | whether the movie displays a control bar the bottom |
| **href** | URL | location to send visitor if they click on the movie |
| **target** | window name | allows **href** to be displayed in another browser window |
| **loop** | true/false | whether the repeat the movie in a loop |
| **pluginspage** | URL | where to send a visitor that does not have the QuickTime plug-in |

For more details on the EMBED parameters, see Apple's QuickTime Authoring site.

## Adding Volcano Videos

**Note: If you do not have the working documents from the previous lessons, download them now.**

In this lesson, we will add a QuickTime movie file to our *Volcano Web* site. The QuickTime movie we use here has been

saved in the format that allows it to start playing before the file is fully downloaded.

This movie was generated by a computer model of the AD79 eruption of the volcano Vesuvius, which destroyed the city of Pompeii. We obtained permission for use of this video from its creator-- see our request message.

1. Since we will be adding a new media type, first create a new folder/directory in your main workspace and name it **movies**. This should be at the same level of your computer files as the **pictures** folder/directory where we have been storing our image files.
2. Go to Lesson 29b Movie Theater to get a copy of the QuickTime movie needed for this lesson (If you cannot see the movie on this page, then you may have to download the software necessary to view it). This movie should be saved as a file called **vesuvius.mov** inside your "movies" folder.
3. Create a new HTML file in your text editor and save it as a file called **vesuvius.html** in the same folder/directory as your main volcano web site HTML files.
4. Enter the following HTML code into the new **vesuvius.html** file:

```
<html>
<head>
<title>Vesuvius Simulation</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
<center>
<embed src="../movies/vesuvius.mov" width="301" height="191"
       autoplay="false" controller="true"
       href="http://tribeca.ios.com/~dobran/" target="_blank"
       loop="false"
       pluginspage="http://www.apple.com/quicktime/download/">
</embed>
<p>
<h2>Computer Simulation of the A.D. 79 Pompeii Eruption</h2>
This animation sequence simulates a <b>Plinian</b> type
eruption where the collapse of a tall eruption column
generates a high speed pyroclastic flow which travels
laterally along the ground. View the entire animation
sequence several times, and then use the slider to
identify the point where the pyroclastic flow is initiated.
<p>
<i>used with permission from Flavio Dobran</i><br>
<a href="http://tribeca.ios.com/~dobran/">http://tribeca.ios.com/~dobran/</a>
<p>
<font face="verdana,arial,helvetica" size=2>
Note: If the movie does not display in this page, try linking
directly to the <a href="../movies/vesuvius.mov">movie file</a></font>

<form>
<input type=button value="Return to Volcano Terminology" onClick="self.close()">
</form>
</center>
</body>
</html>
```

**NOTE: We have assigned a URL (href="http://tribeca.ios.com/~dobran/",**

**`target="_blank"`) so that any click inside the movie area will launch a connection to the Vesuvius web site that is the original source in a new browser window. In this movie we will not have the movie play automatically when the page loads (`autoplay=false`) because we want the person watching the page to drag the movie slider to any point in the movie file. This allows them to interactively explore the movie as a sequence of events in time. If they desire, they can click the play button and watch it play as a movie.**

**We have also added a hyperlink near the bottom of the page for cases where the viewer may not have an appropriate plug-in to play the video embedded in the web page. A direct link may work for them by opening an external software program to play the video.**

5. **Save** and **Load** this html file in your web browser. If the movie does not appear, first check that the file **`vesuvius.mov`** is stored inside a folder called **`movies`**.

By this point you should have a stand alone web page that includes a QuickTime movie file. We will now modify one of our existing web pages, "Volcano Terminology", so that it links to this new page, using some JavaScript to open it in a new window. We will link it using the same method we used in the previous lesson to launch a window with an animated GIF.

1. Open the file **`term.html`** in your text editor.
2. After the end of the **`<map>..</map>`** code, and after the line that reads **`<base target="_self">`** add the following HTML:

```
<h3>Volcanoes in Action</h3>
<a href="vesuvius.html" onClick="window.open('vesuvius.html', 'plinian',
'width=400,height=400,status,menubar'); return false"
onMouseOver="window.status='view computer animation of Plinian eruption';
return true"><img src="../pictures/projector.gif" alt="showtime!"
width="46" height="32" align="left" border=0></a>
Watch a <a href="vesuvius.html" onClick="window.open('vesuvius.html',
'plinian', 'width=400,height=400,status,menubar'); return false"
onMouseOver="window.status='view computer animation of Plinian eruption';
return true">computer animation of a Plinian style eruption</a>,
modeled after the AD 79 eruption of Vesuvius which destroyed
the city of Pompeii. Use the animation to understand the
formation of Plinian eruption columns and how their collapse
generates pyroclastic flows [295k QuickTime movie].
```

**NOTE: In this code we have linked both an icon of a movie projector and the adjacent text, some JavaScript code that will open our new `vesuvius.html` page in a new browser window. If you have trouble with this method, be sure to review lesson 27c where we introduced how to use JavaScript to open new windows.**

**Also note how we provide information to our visitor that the link they may click in will load a file that is 295k in file size, so if they are using a slow Internet connection, they may choose whether they want to receive the content.**

3. **Save** and **Reload** the **`term.html`** file in your web browser. Test the link to see if it opens the new page that includes the lava flow movie.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Compare your work to the HTML of the samples (look for something like **Source** from your browser's **View** menu).

# Review

Review topics for this lesson:

1. What are examples of types of information that might be worth presenting as video over the Internet?
2. What are some of the limitations of placing video in your web page?
3. What is the HTML code for including a digital video clip in your web page?
4. What would you write in an **`<embed>`** tag to have a video clip automatically start playing when the page loads?

# More Information

Unfortunately the variety in types of digital movie formats makes it difficult to choose the right one. If the QuickTime file format is not a viable solution for you, you can try the same approach using two other video formats, AVI and MPEG. For these format, certain web browsers will not be able to display them embedded in the web page but can still be viewed via the direct link.

We have not included these other formats in the downloaded version of the tutorial, but you can find them from our main web site:

- **`vesuvius.mpg`** (MPEG format)
- **`vesuvius.avi`** (AVI format)

Once you download these video clips from the Lesson 28b Movie Theater, you can substitute all HTML references to **`vesuvius.mov`** with the new file names.

# Independent Practice

Try adding some video clips to your own web pages. **Before you use any video clips in a published web page, be sure to obtain permission from the person that created it.**

Visit Volcano World to view other examples of movie clips of volcanoes. Or if you are tired of lava, try:

- **Lycos Multimedia Search**
  `http://multimedia.lycos.com/`
- **Excite Video Search**
  `http://www.excite.com/search/video/`
- **WebSeek Video Search**
  `http://disney.ctr.columbia.edu/webseek/`
- **Yahoo Multimedia:Video**
  `http://dir.yahoo.com/Computers_and_Internet/Multimedia/Video/`
- **NASA's Observatorium**
  `http://observe.ivv.nasa.gov/`

# Coming Next....

"Watson, Come Quick! I need sound in my web page"

**GO TO.... | Lesson Index | previous: "Animate My GIF!" | next: "Multimedia: Sounds Like Audio" |**

**Writing HTML:** Lesson 29b: Movie Time
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut29b.html

# 29c. Sound of [web] Music

"Reading just text on a web page is so.... so... 1994!
I want to hear it
**s i n g !!"**

## Objectives

After this lesson you will be able to:

- Identify the types of content appropriate for using audio
- Describe the issues related to using audio over the web
- Write the HTML to insert a sound file into your web page

---

## Lesson

| media type: | digital audio | |
|---|---|---|
| what it does well: | historic events, narration, providing environmental cues, background music, grabbing attention | |
| **issue to consider** | **rating** | **comments** |
| "hurdle" or barrier for creating media | low high | Hardware needed to digitize sounds is commonly built into most newer computers and may be as simple as a microphone. Free or low coast shareware can be used to edit and modify sound, and many sites exist with downloadable audio files. |
| "breadth" of audience that can view media | narrow wide | Most recent computers have built in ability to play back sound and it is typically built into most web browsers. Many people will have their volume turned down. |

| | | |
|---|---|---|
| bandwidth consumption | low high | Can be quite large but is much less than video. Small sound effect files can be small and some sound file formats (MIDI) are designed to be very small in size. Options exist to "stream" content but it may require special servers. |

Sound is one of our most crucial human senses for receiving information and it can be used very effectively for web sites.

However, as you visit different web sites, the main time you come across sound is hearing some corny rendition of marching music, some vague familiar television show theme song, or tinker-toned Beethoven as soon as a web page loads. When a designer adds irrelevant graphics to a work, it is called "Eye Candy"; using sound for many web sites amounts to "Ear Candy".

"So, M. Opinionated, when is sound good?" Good question. Audio can be very useful for re-casting historic moments (speeches, special events, news, etc). It often is a less bandwidth alternative to video when the content is mostly spoken (e.g. classroom lecture). Obviously it might be used to learn about music and acoustics, to compare renditions of classic works, and simply to entertain-- the web can act as a radio. Sound **may** be used to set an environment (soft music for reading a poem).

Like video, sound must be converted analog to digital, from audio waves that travel through air, to numerical representations of wave frequencies. It can be done as easily as talking into a microphone plugged into the back of your computer, to high end digital recording in a sound studio. In the end, you end up with a computer file that can be played back through a speaker and, again travels through the air to your ears.

Sound files come in many different formats, which creates another layer of alphabet soup confusion. Some of the common file formats are WAVE (WAV), AIFF, AU, MIDI, MP3 etc. A comprehensive overview of the differences between them is beyond our tutorial, and we will just have to accept that we can have a sound file on a computer that can be sent over the Internet and played at the other end.

One format is of special interest, MIDI, because it generally makes for the smallest sound files. Rather than digitizing the sound waves itself, in MIDI format sounds are represented as more or less text characters representing the notes on different types of instruments, and when it is played back, the computer uses a built in sound library to play back the designated note. Therefore, the data that sent is very small. MIDI sounds tend to have a "toy-like" playback and are no where as rich as a CD audio.

Many sites offer sound in RealAudio format in which the sound files streams from a special web server to your web browser. This is most efficient for so called "web radio" stations, sites that offer real time audio news feeds, sites that promote musical events, etc (e.g. World Radio Network, RadioTower, Broadtcast.com, NPR, CNN). Because it requires special equipment, RealAudio is not part of this tutorial.

We also should mentioned the format known as MP3, or MPEG Audio Layer 3, a special form of high quality audio that can compress a very rich sound file into a much smaller file. It may be a music format that changes the music industry. You can find many sites such as MP3.com that offer MP3 music, but again, encoding content into this special format calls for special hardware and software (and touches on tricky legal issues of encoding other artist's content).

For more information on audio over the web, see the Web Developer's Virtual Library for Audio for the WorldWide Web

To include a sound file in a web page, we typically use the same **<embed>** tag that we saw in the previous lesson on web video:

```
<embed src="muzak.wav" autoplay=true controller=false></embed>
```

which is the typical way to write a sound file that will be played automatically as soon as a web page is opened, and by having the controller invisible, to the person seeing your page, the music just starts playing magically. If you want to provide them the choice of playing the sound or not, you would have the controller visible and the autoplay set to **`false`**:

```
<embed src="muzak.wav" autoplay=false controller=true></embed>
```

Again, if you are going to use music or sound in your web page, think carefully about the purpose of the content and if it will be reasonable to hear it every time a person goes to that page. You should never assume that the audio will work for every visitor (and some visitors are deaf), so consider providing text alternatives or suggestions for people that may not be able to hear the content.

## Adding Audio

**Note: If you do not have the working documents from the previous lessons, download them now.**

In this lesson, we are going to add some audio content to our main volcano web page. To add some relevancy to the importance of Volcanoes, we will use content from a letter written by an observer of the volcano Vesuvius, which in the year 79 AD destroyed the city of Pompeii. Along with the text of the letter, we have an audio recording read by a starving actor pretending to be the writer, plus a photo of one of the remains of an actual victims of this event. In this treatment, we provide information in written, visual and audio formats!

The sound file we use is actually saved in the format of a QuickTime digital video, with no video, only the sound track, to take advantage of the wide support for QuickTime in web pages and the advanced file compression available for this format. Plus we can save it in the format that streams it from a web server. The file could also have been a WAV or AIF file.

1. Go to the Lesson 29c Sound Studio to download a copy of the sound file. Save it as a file named **`pliny.mov`** inside your **`movies`** folder.
2. Go to the Lesson 29 Image Studio to download a copy of the image file we will use in this lesson. SAve it as a file called **`bodies.jpg`** inside your **`pictures`** folder.
3. Create a new file in your text editor and name it **`pliny.html`**
4. Enter the following HTML in this new file

```
<html>
<head>
<title>Pliny's Words</title>
</head>
<BODY BGCOLOR=#000000 TEXT=#FFFFCC LINK=#33CCFF VLINK=#FF6666>
<h1 align=center>Pliny the Younger's Observations</h1>
In 79 AD, Pliny the Elder died during the eruption of
the volcano Vesuvius. His nephew, Pliny the Younger
escaped the destruction of Pompeii and left a written
account of the eruption. As you read the letter below,
listen to this audio version read by a well-known actor:
<p>
<center>
<embed src="../movies/pliny.mov" width="180" height="16"
       autoplay="false"  controller="true"
       pluginspage="http://www.apple.com/quicktime/download/">
</embed>

<!-- start of Volcano Web standard footers -->
<SCRIPT LANGUAGE="JavaScript">
```

```
<!-- hide  scripts from old browsers

document.write('<p><hr><FONT FACE="helvetica,arial" size=-1><i>Volcano Web:</i>
<b>');
document.write(document.title + '</b><BR>');
document.write('created by Lorrie Lava, ');
document.write('<a href="mailto:lava@pele.bigu.edu?subject=' + document.title +
'">');
document.write('lava@pele.bigu.edu</a><br>');
document.write('Volcanic Studies, <a href="http://www.bigu.edu/">');
document.write('Big University</a><p>');

// append a modification date only if server provides a valid date
if (Date.parse(document.lastModified) > 0) {
        document.write('<b>last modified: </b>' + document.lastModified + '<BR>');
}
document.write('<b>URL: </b>' + document.location.href + '</FONT><P>');

// done hiding from old browsers -->
</SCRIPT>
<!-- end of Volcano Web standard footers -->
</body>
</html>
```

**NOTE: We have written a new web page with a title, some introductory text, a footer, and the `<embed>` tag to include a control that will allow the site visitor to play a QuickTime sound file.**

5. **Save** and **Load** this html file in your web browser. Verify that the sound file loads and can be played back with the QuickTime control bar.

Now we will add some more content, the text of the letter, inside a HTML table, with a right aligned image.

1. Add this HTML after the end of the **`<embed>..</embed>`** tag:

```
<p>
<table border=0 cellpadding=6 cellspacing=1 width=70%>
<tr>
<td><font size=+1 color=#FF9999>
The carts that we had ordered brought were moving in
opposite directions, though the ground was perfectly flat,
and they wouldn't stay in place even with their wheels
blocked by stones. In addition, it seemed as though the sea
was being sucked backwards, as if it were being pushed back
by the shaking of the land. Certainly the shoreline moved
outwards, and many sea creatures were left on dry sand.
Behind us were frightening dark clouds, rent by lightning
twisted and hurled, opening to reveal huge figures of flame.
These were like lightning, but bigger. At that point the
Spanish friend urged us strongly: "If your brother and uncle
is alive, he wants you to be safe. If he has perished, he
wanted you to survive him. So why are you reluctant to
escape?" We responded that we would not look to our own
safety as long as we were uncertain about his. Waiting no
```

longer, he took himself off from the danger at a mad pace. It wasn't long thereafter that the cloud stretched down to the ground and covered the sea. It girdled Capri and made it vanish, it hid Misenum's promontory. Then my mother began to beg and urge and order me to flee however I might, saying that a youngs man could make it, that she, weighed down in years and body, would die happy if she escaped being the cause of my death. I replied that I wouldn't save myself without her, and then I took her hand and made her walk a little faster. She obeyed with difficulty, and blamed herself for delaying me. <p> Now came the dust, though still thinly. I look back: a dense cloud looms behind us, following us like a flood poured across the land. "Let us turn aside while we can still see, lest we be knocked over in the street and crushed by the crowd of our companions." We had scarcely sat down when a darkness came that was not like a moonless or cloudy night, but more like the black of closed and unlighted rooms. You could hear women lamenting, children crying, men shouting. <img src="../pictures/bodies.jpg" alt="bodies buried in ash" width="358" height="135" align="right">Some were calling for parents, others for children or spouses; they could only recognize them by their voices. Some bemoaned their own lot, other that of their near and dear. There were some so afraid of death that they prayed for death. Many raised their hands to the gods, and even more believed that there were no gods any longer and that this was one last unending night for the world. Nor were we without people who magnified real dangers with fictitious horrors. Some announced that one or another part of Misenum had collapsed or burned; lies, but they found believers. It grew lighter, though that seemed not a return of day, but a sign that the fire was approaching. The fire itself actually stopped some distance away, but darkness and ashes came again, a great weight of them. We stood up and shook the ash off again and again, otherwise we would have been covered with it and crushed by the weight. I might boast that no groan escaped me in such perils, no cowardly word, but that I believed that I was perishing with the world, and the world with me, which was a great consolation for death. <p> At last the cloud thinned out and dwindled to no more than smoke or fog. Soon there was real daylight. The sun was even shining, though with the lurid glow it has after an eclipse. The sight that met our still terrified eyes was a changed world, buried in ash like snow. We returned to Misenum and took care of our bodily needs, but spent the night dangling between hope and fear. Fear was the stronger, for the earth was still quaking and a number of people who had gone mad were mocking the evils that had happened to them and others with terrifying prognostications. We still refused to go until we heard news of my uncle, although we had felt danger and expected more. <p> You will read what I have written, but will not take up your pen, as the material is not the stuff of history. You

```
have only yourself to blame if it seems not even proper
stuff for a letter. Farewell. </td>
</tr>
</table>
<p>
<font size=2>Pliny the Younger's Letter, text from the
<a href="http://www.iath.virginia.edu/pompeii/pliny.html">Pompeii Project</a><br>
Pictures of Pompeii victims, from
<a href="http://www.geo.mtu.edu/~boris/VESUVIO_79.html">photos
taken in 1992 by Werner Keller</a>.</font>
<p>
<font size=+2>If this historic account shows you<br>
the impact of a volcano on human life,
<a href="index1.html">continue on ...</a></font>
```

> **NOTE: There is nothing new in this section, so you can easily copy and paste it to your HTML. Note that we have included an image file inside the table and allowed the text to trap around it. Below the table are some links to related source materials as well as a link to our main Volcano web site.**

2. **Save** and **Reload** in your web browser
3. The last thing we will do is build a link from our main *Volcano Web* page to draw new visitors to this new page. Open the HTML file **`index1.html`** in your text editor.
4. After the line that reads:

   ```
   who died of asphyxiation after
   observing the destruction of Pompeii by the
   79 A.D. eruption of Mount Vesuvius.
   ```

   add this line:

   ```
   <a href="pliny.html">"Hear" Pliny's account!</a>
   ```

5. **Save** and **Reload** in your web browser

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Compare your work to the HTML of the samples (look for something like **Source** from your browser's **View** menu).

# Review

Review topics for this lesson:

1. What are examples of types of information that might be worth presenting as sound over the Internet?
2. What are some of the limitations of placing sound in your web page?
3. What is the HTML code for including an audio clip in your web page?
4. How would you write the HTML that would embed a sound that would continuously loop?

# More Information

Like you saw in our previous lesson on video, the QuickTime movie format we used here for the sound file may not work for everybody. You can use the same HTML code to play back a sound in WAVE format. However, keep in mind the QuickTime employs a great deal of effective compression to reduce the file size of a sound file; the 360k sound file in QuickTime format is 10 times as big (3.2 MB) as a .wav file!

We have not included the WAVE file in the downloaded version of the tutorial, but you can find it from our main web site, as files:

Once you download the sound file from the Lesson 28c Sound Studio, you can simply substitute all HTML references to **`pliny.mov`** with the new file name, **`pliny.wav`**.

# Independent Practice

Try adding some sound to your own web pages. **Before you use any sound files in a published web page, be sure to obtain permission from the person that created it.**

- **SoundAmerica**
  http://soundamerica.com/
- **The Free Site**
  http://www.thefreesite.com/
- **Yahoo : Multimedia**
  http://dir.yahoo.com/Computers_and_Internet/Multimedia/audio/

# More Information

You can always build a link to any media file by using our familiar hypertext link:

```
Try out my latest <a href="new_riff.wav">guitar riff</a>
or <a href="trumpet.mid">trumpet call</a>
```

assuming we had a WAV file and a MIDI file with these names. The web browser will leave the page it was linked from and then try to load the sounds the best it can, in blank web page. After hearing the sound, you would have to press the browser **back** button to return to the page you came from. The implementation is not as seamless as embedding it in the page but it works.

---

# Coming Next....

Stand back and brace yourself... a S H O C K W A V E! is coming

**GO TO.... | Lesson Index | previous: "MovieTime!" | next: "Shockwave" |**

---


©1994-1999 Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut29c.html

# 29d. Hit Me With a Shockwave

## "Interactive multimedia *inside* a web page?" I am shocked!"

Yes you are, you can have **shockwave** with your HTML and "knock 'em out" with impact...

## Objectives

After this lesson you will be able to:

- Identify the benefits of interactive multimedia
- Describe the issues related to using shockwave over the web
- Write the HTML to insert a shockwave file into your web page

---

## Lesson

Shockwave is a technology introduced in early 1996 by Macromedia that leverages the strength of its powerful and widely used multimedia development program, Director, so that interactive multimedia content can be included within a web page. To get a small taste of what web based multimedia can do, visit Macromedia's Gallery.

| media type: | shockwave | |
|---|---|---|
| what it does well: | synchronized media, animations, simulations, manipulatives, communicate with databases, realistic interfaces, non-linear content | |
| **issue to consider** | **rating** | **comments** |
| "hurdle" or barrier for creating media | low high | To create Shockwave content you need Macromedia Director software and the ability/skills to design and program in it. |
| "breadth" of audience that can view media | narrow wide | Playing shockwave within a web page requires extra software (ActiveX controls for Internet Explorer/NetScape Plug-ins), but estimates show more than 50% of web users have this already. Technology has improved so it can automatically update its components when needed. |
| bandwidth consumption | low high | Quite variable as it depends on the type of interaction, media used, and the programming skills of the content creator. Shockwave content can now be "streamed" so it can be delivered in small chunks as the viewer starts interacting with it. It is very capable of internally compressing the size of its media to make it move more "quickly" across the internet. Shockwave audio is high quality and fully streams without special hardware. |

What Director does well, and thus Shockwave allows you to do in a web page, is to deliver "rich" content experiences that may include text, graphics, animation, sound, video, in almost any kind of visual design you can imagine. It is powerful because it can manipulate and synchronize these different media types, as well as communicate with other systems, such as databases or other software.

The challenge to the average web page designer is that to do Shockwave well, they must learn Director, well, no small task, and not cheap software to buy. Because most shockwave designs are unique, often they cannot be easily re-purposed like "clip art" but there are some exceptions we will point out later.

One of the limitations is that Shockwave calls for extra software functionality to work in your web browser. Many web browsers and new computers are shipped with this already installed. The technology has improved enough so that if a person accesses a site that requires Shockwave, it can either start downloading the needed "parts" or send you to a site so you can easily install Shockwave on your computer. Once you have it installed for one web site, any other web site can now use this functionality.

This lesson will not cover the creation aspects of Shockwave, but will show you how to add an existing shockwave application to your web site. Shockwave files always end in ".dcr"

Because of the way Shockwave is implemented in web browsers, we actually have to write almost duplicate HTML that works for both; Internet Explorer uses a **<object>** tag while NetScape uses the **<embed>** tag (similar to what we saw for video, lesson 29b and audio, lesson 29c, but with a few different parameters).

The general HTML code looks like:

```
<object classid="clsid:166B1BCA-3F9C-11CF-8075-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab#version=7,0,2,0"
WIDTH="480" HEIGHT="300" NAME="sw" ID="swmovie">
        <PARAM NAME="SRC" VALUE="sw_file.dcr">
        <PARAM NAME="BGCOLOR" VALUE="#FFFFFF">

        <embed src="sw_file.dcr"
        width=480 height=300 bgcolor=#FFFFFF
        type="application/x-director"
        pluginspage="http://www.macromedia.com/shockwave/download/">
        </embed>
</OBJECT>
```

As you may see, the **<embed>** tag for NetScape (shown in **purple**) is contained within the Internet Explorer **<object>** tag (shown in **red**). What it means is that the NetScape browser will ignore the <object> tags and the Internet Explorer browser will ignore the **<embed>** tags. Here is a bit more explanation of the other options in the tags:

| purpose of option | OBJECT tag | EMBED tag |
|---|---|---|
| web location of shockwave installer code, redirects browser there to download needed software if not present | object classid="....." codebase="http://...." | pluginspage="http://..." |
| dimensions in pixels | width=..., height=... | width=..., height=... |
| server MIME type, identifes the file as shockwave format | (not used) | type="application/x-director" |
| identification code, can be used with browser scripting (JavaScript, VBScript) | NAME="..", ID="..." | NAME=".." (not required) |
| relative file location of shockwave file | <PARAM NAME="SRC" VALUE="sw_file.dcr"> | src="sw_file.dcr" |
| background color while loading, helps blend with color of page | <PARAM NAME="BGCOLOR" VALUE="#FFFFFF"> | bgcolor=#FFFFFF |

## Adding Shockwave

**Note: If you do not have the working documents from the previous lessons, download them now.**

In this lesson, we are going to add a shockwave application that can read in any number of photographic images of volcanoes, and has tools to allow you mark the height and width of the volcano and calculate this as a ratio. Because of the way it has been designed, you can add any number of additional volcanoes by collecting more images and editing two lines of the HTML code.

We could have provided quite a bit more function in this if we chose to. We could have programmed it so that after you used the tool, it could save the data to our web server by having Shockwave send data to a CGI form. We could have added a button that would send an email message to an instructor. We could have created another button that would allow you to print or save the results from using the tool. We could have...

But we decided to keep it simple, as this is not a **real** site!

The shockwave file itself is only 22k in size, and the images it uses are dynamically linked only when needed. This makes shockwave load quicker in the web page plus allows you to add the additional images without having to re-program the shockwave file.

Before you start, you can visit a copy of the page you will build. This will allow you to try the shockwave application, as well as test if it is already installed on your computer.

If you are ready, we can now show you how to set up this web page that includes shockwave:

1. Create a new directory/folder in your workspace named **`dswmedia`**. This is the place we will store our shockwave file and the images it uses. We must place shockwave files within a folder with this name for all of its features to work when run from your desktop computer.
2. Go to the Lesson 29d Image Studio to download four images of volcanoes that we will use within Shockwave. Save these inside your **`dswmedia`** folder.
3. Now you will have to get a copy of the shockwave file.

    a. For users of Windows-based computers , download the shockwave file in a 22k zip file. You may need a program such as WinZip or StuffIt Expander to decompress the archived file.
    b. For users of Apple Macintosh-based computers , download the shockwave file in a 30k BinHex file. You may need a program such as StuffIt Expander to decompress the archived file.
    c. You should end up with a file named **`volc_hw.dcr`** in the place wher your Internet downloads are stored. Move this file into your **`dswmedia`** folder. This folder should now contain 4 image files and one shockwave file.

4. Create a new file in your text editor, name it **`measure.html`**, and save it in the same workspace folder/directory as your other HTML files.

5. Enter the following HTML in this new file:

```
<html>
<head>
<title>Measuring Volcano Shapes</title>
<body bgcolor=#000000>
<center>

<!-- begin ---------= s h o c k w a v e =-------------------
The OBJECT tag is used by Internet Explorer and the EMBED tag
is used by NetScape Navigator. You can add/edit the volcano
functionality by editing the values for:

  sw1 : names of all volcanoes
  sw2 : the file names of the volcano images ------------------>

<!-- OBJECT tag used by Microsoft Internet Explorer ----------->
<object classid="clsid:166B1BCA-3F9C-11CF-8075-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab#version=7,0,2,0"
WIDTH="480" HEIGHT="400" NAME="volcs" ID="hw">
```

```
        <PARAM NAME="SRC" VALUE="../dswmedia/volc_hw.dcr">
        <PARAM NAME="BGCOLOR" VALUE="#000000">

        <PARAM NAME="sw1"
          VALUE="San Francisco Peaks AZ,Black Butte CA,Mauna Kea HI,Popacatepetl
Mexico">
        <PARAM NAME="sw2"
           VALUE="sfpeaks.jpg,black_butte.jpg,mauna_kea.jpg,popo.jpg">

<!-- EMBED tag used by NetScape Navigator ---------------------->
<embed src="../dswmedia/volc_hw.dcr"
        width=480 height=400 bgcolor=#000000
        type="application/x-director"
        pluginspage="http://www.macromedia.com/shockwave/download/"
        sw1="San Francisco Peaks AZ,Black Butte CA,Mauna Kea HI,Popacatepetl Mexico"
        sw2="sfpeaks.jpg,black_butte.jpg,mauna_kea.jpg,popo.jpg">
</embed>
</OBJECT>
<!-- end -----------= s h o c k w a v e =--------------------->

<form>
<input type="button" value="close tool" onClick="self.close()">
</form>
</center>
</body>
</html>
</font></pre>
```

**NOTE: Most of this HTML code is for including the Shockwave application; the only other feature is a JavaScript Form at the bottom that creates a button for closing the window.**

**The `height` and `width` of the shockwave file is 480 pixels wide and 400 pixels high (you cannot scale shockwave files like images; if the dimensions in the HTML code are smaller, it will crop out part of the shockwave content). The shockwave file is located in the `dswmedia` folder, which is one level above the file location of this HTML file.**

**The values of the parameter sw1 lists in order, the titles of each of the images that will be read into the Shockwave file. Likewise, the values of the parameter sw2 lists in order, the corresponding image file names. Note how this information is included twice, once for the Internet Explorer OBJECT tag and once for the NetScape EMBED tag.**

6. **Save** and **Load** this html file in your web browser. Verify that the Shockwave application loads and plays.

**NOTE: If there is a problem loading the images, make sure that you have entered the sw1 and sw2 strings exactly as shown above (there should be no spaces after the commas).**

Once our shockwave page is working, we need to add a link that will open it in its own browser window, like we have done in the previous multimedia lessons.

1. Open the file **`term.html`** in your text editor.
2. After the section we created in the last lesson (the header "Volcanoes in Action" with the link to the QuickTime movie), add this HTML:

```
<h3>Shapes of Volcanoes</h3>
Their general "shape" can help you classify volcanoes by a
measure known as the <b>aspect ratio</b>, or the ratio of
height to width. Since this value is a ratio, it does not
matter if we measure it in actual dimensions
(feet, meters, etc) or if we use a ruler on a photograph.
Use the <a href="measure.html" onClick="window.open('measure.html',
'plinian', 'width=500,height=480,status,menubar'); return false"
onMouseOver="window.status='aspect ratio measuring tool';
```

```
return true">Volcano Web tool</a> to measure the aspect
ratio of several volcanoes. (Requires
<a href="http://www.macromedia.com/shockwave/">Shockwave</a>)
```

3. **Save** and **Reload** in your web browser. Test this and make sure it opens a browser window that then loads the Shockwave equipped page.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Compare your work to the HTML of the samples (look for something like **Source** from your browser's **View** menu).

# Review

Review topics for this lesson:

1. What are some of the things you can do in Shockwave that make it useful?
2. What are some of the limitations of using Shockwave?
3. What is the HTML for including a Shockwave file? Why does it have some duplicate information?

# Independent Practice

Experiment with the Shockwave page you created. See if you can figure out how to change the order in which the volcanoes are displayed. Try to find some other images of Volcanoes and see if you can add them to the set of four that we used here.

We also have created two Shockwave templates that you can use without having to know anything about how Shockwave works. These are part of an online workshop that shows you how to add Shockwave and JavaScript functionality to your own web pages.

1. The Quizzer allows you to create sets of multiple choice quizzes, where each time the quiz is taken, the order in which the questions appear, and the order in which the answers for each are displayed, are randomized.
2. The Clicker allows you to create web-base slide shows that may include multiple sets of captions (two languages) and can have audio associated with each of the slides.

# More Information

This is but one example of how shockwave turn a web site into a place that is much more than a static collection of text and pictures. You can find numerous other examples of Shockwave from:

- **Macromedia's Shockwave.com**, especially their Shocked Site of the Day
  http://www.shockwave.com/
- **DirectorWeb's Shockwave List o' Sites**
  http://www.mcli.dist.maricopa.edu/director/shockwave.html
- **The Shocked WebRing**
  http://venus.aros.net/~jjreese/shocked/
- **ZDNet's Shockwave Showcase**
  http://www.zdnet.com/devhead/resources/showcase/search/shockwave.html

---

# Coming Next....

Sit back as we try to pour a hot cup of J A V A !

**GO TO.... | Lesson Index | previous: "MovieTime!" | next: "Java" |**

---

**Writing HTML:** Lesson 29d: Hit Me with a Shockwave!
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut29d.html

# 29e. Small Cup of Java (to go)

# J A V A

Just a small cup, I need to watch my caffeine intake. Oh yes, it does energize me, but I also get the shakes... in my web page? Of course, why not? Everyone uses it..."

## Objectives

After this lesson you will be able to:

- Identify the benefits of using Java applets
- Describe the issues related to using Java in your web page
- Write the HTML to insert a Java applet into your web page
- Write the HTML to display a message if a visitor to a site cannot view a Java applet
- Create a framed page that uses JavaScript to dynamically create the HTML for the Java applet

---

## Lesson

The promise of Java, when introduced by Sun Microsystems in the early 1990s, was huge and some may say, "hyped". In about as brief as we can try to state it, Java is a new programming langauge that allows someone to create applications, or "applets" that can run on any system, any place, that supports the "Java Platform":

> *With Java™ technology, you can use the same application from any kind of machine -- a PC, a Macintosh computer, a network computer, or even new technologies like Internet screen phones.*

So a Java applet does not need any extra software to run, when it is requested, all of its functional pieces are sent over a network to whomever/whatever requested it, and then the Java runtime technology on the receiving end then runs it.

For more about Java, see http://java.sun.com/

| media type: | java | |
|---|---|---|
| what it does well: | tools, data manipulation, charts and graphs, interactive navigation controls, dynamic text, image alteration effects. | |
| **issue to consider** | **rating** | **comments** |

| | | |
|---|---|---|
| "hurdle" or barrier for creating media | low high | To build your own Java applets you must have an understanding of the programming language or have access to some of the visual type application builder tools. There are, however, many online repositories of Java applets that you can download and use. |
| "breadth" of audience that can view media | narrow wide | Java is well-supported on most web browsers beyond version 3.x, although it tends to run more slowly and less reliably on Macintosh computers. There can be a long delay and an empty white space while the browser reads "Java loading" |
| bandwidth consumption | low high | Quite variable as it depends on the type of interaction and the complexity of the task performed. When a page includes a Java applet, it must download anywhere from a few to 30 or more small files all of which must be received before the page is functional. |

As we mentioned in lesson 27, despite the name similarity, Javascript-- a browser scripting language, is quite different from Java, a computer programming language. JavaScript is literally interpreted line by line as the browser accesses it. Java applets are written as line computer code which is then "compiled" or converted to a run-time application. A compiled application can be more powerful, secure, and faster than interpreted script, but is also more complex to create.

Unfortunately, beyond a number of well-integrated uses of Java, the most common use you will find on the web is a Java applet to display a scrolling banner of text, which may roll horizontally or vertically (see lesson 17 for our thoughts on this!), or the common effect of adding ripples to an image.

Like we saw in our previous lesson on Shockwave, the Java files tend to be closed black boxes that we do not even worry about how they work. With some applets we can alter or control their functions through information we send the applet from our HTML code.

This lesson will not cover the creation aspects of Java programing, but will show you how to add an existing applet to your web site.

The simple HTML code for including a Java applet is

```
<APPLET CODE=MyApplet.class WIDTH=XX HEIGHT=YY>
</aPPLET>
```

where **`MyApplet.class`** is the file name of a compiled Java applet file, and **`WIDTH`** and **`HEIGHT`** are the pixel dimensions that the applet occupies on screen. Quite often, you may include other information that are sent to the applet via parameters:

```
<APPLET CODE=MyApplet.class WIDTH=XX HEIGHT=YY>
<param name="param1" value=my1Value>
<param name="param2" value=my2Value>
```

```
</aPPLET>
```

where each parameter has a name the applet is looking for and some value, which may be a string of text or a number:

```
<APPLET CODE=MyApplet.class WIDTH=XX HEIGHT=YY>
<param name="param1" value="The Meaning of Life is Cheese;
<param name="param2" value=129;
</aPPLET>
```

Finally, you can include text that will be displayed only if the web browser does not support Java (remember, the browser will ignore anything inside tags it does not understand), similar to using the **ALT** option for **<img ...>** tags (lesson 7a) or the **<NOFRAMES>** tag for frames (lesson 26).

```
<APPLET CODE=MyApplet.class WIDTH=XX HEIGHT=YY>
<param name="param1" value=my1Value>
<param name="param2" value=my2Value>
Sorry, but it looks as thought your web browser cannot
display this cool Java applet.
</aPPLET>
```

So any string of text *inside* the **<APPLET>...</aPPLET>** tags is ignored by a Java-enabled browser since it is not written as am applet tag or a parameter tag) and is the only portion displayed for a Java-disabled web browser.

## Adding Java

**Note: If you do not have the working documents from the previous lessons, download them now.**

In this lesson we are going to use a Java applet that allows us to send it an image file, and in our web page, allow the viewer to zoom in and out. The pictures we are going to use are of volcanic rocks taken with a special microscope that allows us to see the minerals and structures in the rock. In our web page, we can use the Java applet to act like a virtual microscope.

The Java applet used here is called "ImageZoom" and more information is available from http://www.vivaorange.com/ImageZoom/. This applet is free for non-commercial use (we found it from using one of the Java resource sites listed below).

The general HTML for using this applet is:

```
<applet code="ImageZoom.class" width="[width]" height="[height]">
  <param name="IMAGE" value="[image file]">
  <param name="ZoomLevel" value="[zoom level]">
  <param name="PanSpeed" value="[speed]">
  <param name="Cursor" value="[cursor]">
  <param name="Preload" value="[preload]">
</applet>
```

where:

- [width] is the width of the image in pixels
- [height] is the height of the image in pixels
- [image file] is the file location of the image file
- [zoom level] is the number of zoom levels allowed, from 1 to 10

- [speed] is the speed of moving when the image is panned, from 1 to 10.
- [cursor] is the type of cursor; 1=a hand cursor and 2=a cross-hair cursor
- [preload] is the preload mode for faster image zooming. Value is either ³ON" or OFF"

The applet file, **`ImageZoom.class`** itself is only 5k in size, and the image it uses is read in as a parameter, as well as the other options listed above. Check the source web site for more information about this applet.

1. First create a new folder/directory in your workspace named **`scope`**
2. Go to the Lesson 29e Image Studio to download four images taken from a petrographic microscope. Save these inside your **`scope`** folder.
3. Now you will have to get a copy of the applet file.

    a. To do this you will need to access the "secret" menu by either right mouse clicking (Windows, Unix) or clicking and holding the mouse down (Macintosh) on **this link to the Java Applet file (ImageZoom.class)** until you see a pop-up menu.
    b. From the menu that appears, select **Save this Link As...** or **Save Target As...**
    c. When a dialog box appears, be sure to select **Source** if there is menu labeled **Format**
    d. Save it as a file named **`ImageZoom.class`** inside your **`scope`** folder

4. Create a new file in your text editor, name it **`javascope.html`**, and save it in your **`scope`** folder

5. Enter the following HTML in this new file:

```
<html>
<head>
  <title>Java Microscope</title>
</head>
<body bgcolor="#000000" text="#EEEEEE">
<center>
<applet code="ImageZoom.class" width=400 height=265 vspace=14>
    <param name="IMAGE" value="pw_vis.jpg">
    <param name="ZoomLevel" value="6">
    <param name="PanSpeed" value="4">
    <param name="cursor" value="1">
    <param name="Preload" value="on">
Sorry, but your web browser cannot load this Java Applet :-(
<p>Here at least is a picture of the sample:<br>
<img src="pw_vis.jpg" width=400 height=265><p>
</applet>
<font face="verdana,helvetica" size="2">
<br>BISHOP TUFF: Partly Welded (Visible Light)
click to zoom, move mouse to edge to pan
</center>
</body>
</html>
```

**NOTE: We are only using one of our image files here; later we will show you a more dynamic way to use this applet. We have also added a parameter `vspace` in the `applet` tag to allow for 14 pixels of vertical "padding" above and below the applet on the page. (This is similar to what we did with images in lesson 20)**

**The `width` and `height` of the images file are 400 pixels wide and 265 pixels high. We set the other**

**parameters to have 6 levels of zooming, a panning speed of 4, and the cursor option to use a hand cursor. Also note the text we inserted inside the `applet` tag to provide feedback if the person vising this site does not have a Java enabled browser. This way, they at least can see the content, even if they cannot zoom in on the image. You can test this if you look in your web browser preferences and find the option to turn Java off (be sure to turn it back on!).**

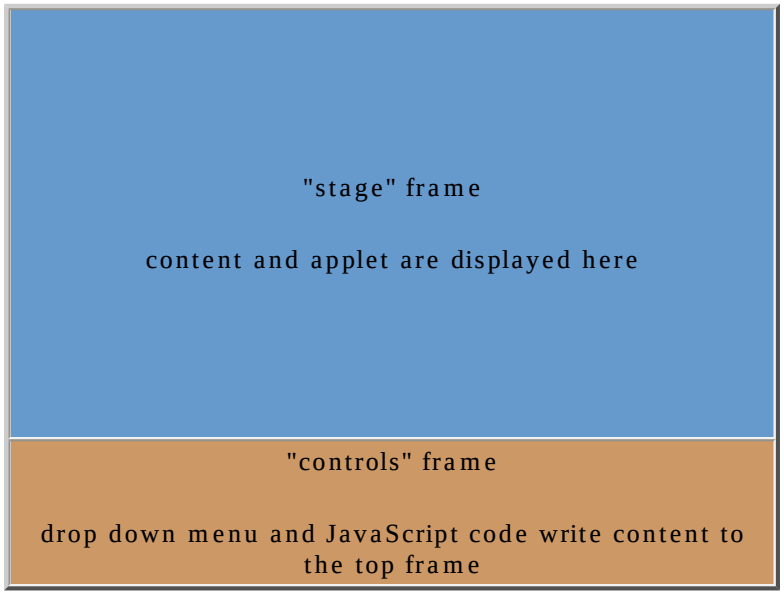
**Also, on some computers (especially Macintosh computers), Java applets quite often will convert the cursor used in NetScape even when you are finished with the applet.**

6. **Save** and **Load** this html file in your web browser. Verify that the Java applet loads and plays. Click on the image several times to see that it enlarges the image and then move the cursor to an edge (you may have to leave it there a few seconds) to verify that the image "pans" or slides across the viewing region.

   **NOTE: If your web page does not work as expected, compare your HTML to the source of the example page.**

We now have seen how to load the ImageZoom applet with a defined image so you can zoom and pan as if you were using a microscope (actually the image would be much more clear at high magnifications on a real microscope; on the computer we are simply enlarging it and we start to see the artifacts of square image pixels).

But we can do something more flexible by using some JavaScript to dynamically create the HTML for the applet, allowing us to choose from a series of images to load in the applet. For more on this aspect of JavaScript, review lesson 27b. To set this up, we are going to create a framed web page (see lesson 26) that will look like:




1. Create a new file in your text editor, name it **`index.html`**, and save it in your **`scope`** folder
2. Write this HTML in this new file to create the frameset page (refer to lesson 26 if you need a refresher on frames)

```
<html>
<head>
        <title>Volcanic Rock Microscope</title>
</head>
<frameset rows="*,70" border=0>
        <frame src="stage.html" name="stage"
               marginheight="12" marginwidth="12"
               scrolling="auto" noresize>

        <frame src="controls.html"
```

```
                marginheight="8" marginwidth="8"
                scrolling="no" noresize>
</frameset>
<noframes>
<h2 align=center>NOTE: This site uses frames, but apparently
your browser does not support this feature.</h2>
</noframes>
</html>
```

**NOTE: This sets up a framed web page that has a bottom frame 70 pixels high and the top frame uses the rest of the window space. Also note the parameters `marginheight` and `marginwidth` that allows you to define the amount of "padding" or extra space between the content of a frame and its edge.**

3. **Save** this html file.

4. Create a new file in your text editor, name it **`stage.html`**, and save it in your **`scope`** folder. This is the page loaded in the top frame.
5. Write this HTML in this new file:

```
<html>
<head>
        <title>Microscope Stage</title>
</head>
<body bgcolor="#000000" text="#FFFF00">
<center>
<table width=80% height=80% border=0 align=center>
<tr>
<td align=center>
<h1>Volcanic Rock Microscope</h1>
<font face="verdana,helvetica" size=3>Select a rock sample from the
menu below to view it as it would appear in a petrographic microscope.
<p>
Each time you click the mouse it will zoom in. Move the mouse to
the edge of the image to pan across  the view (Panning will work
only when the image has been magnified at least once).
<p>
The microscope requires a Java-enabled web browser.
</td>
</tr>
</table>
</center>
</body>
</html>
```

**NOTE: This is just static content for the first view of the whole page. Note how we used the table sizing technique introduced at the end of lesson 21 to center the content.**

6. **Save** this html file.

7. Create a new file in your text editor, name it **`controls.html`**, and save it in your **`scope`** folder. This is the page loaded in the bottom frame and contains JavaScript code to activate a drop down menu and to generate content into the top frame

```
<html>
<head>
<script language="JavaScript">
<!--
function scope( rockmenu ) {
// Called from menu to either load static content into the top frame
// or to dynamically write code for embedding a Java applet

// rock identifies the file name, blurb is  the caption
  rock = rockmenu[rockmenu.selectedIndex].value;
  blurb = rockmenu[rockmenu.selectedIndex].text;

  if (rock != "") {
    // ignore blank menu values and reselect the first menu item
      rockmenu.selectedIndex = 0;

    if (rock== "help") {
      // selected help, load the opening page
      parent.frames[0].location.href="stage.html";
      rockmenu.selectedIndex = 0;

    } else if (rock== "close")  {
        // call function to close the microscope
      close_scope();
      rockmenu.selectedIndex = 0;

    } else {
      // load applet with selected image
      with (parent.frames[0]) {
        document.write('<html><head><title>' + blurb + '</title></head>');
        document.write('<body bgcolor="#000000" text="#EEEEEE">');
        document.write('<center><applet code="ImageZoom.class"');
        document.write(' width=400 height=265 vspace=14>');
        document.write('<param name="IMAGE" value="'+ rock +'.jpg">');
        document.write('<param name="ZoomLevel" value="6">');
        document.write('<param name="PanSpeed" value="4">');
        document.write('<param name="cursor" value="1">');
        document.write('<param name="Preload" value="on">' );
        document.write('Sorry, but your web browser cannot load this Java Applet :-
('');
        document.write('<p>Here at least is a picture of the sample:<br>');
        document.write('<img src="'+ rock +'.jpg" width=400 height=265><p>');
        document.write('</applet>');
        document.write('<font face="verdana,helvetica" size="2">');
        document.write('<br>BISHOP TUFF: ' + blurb);
        document.write('click to zoom, move mouse to edge to pan</center>');
        document.write('</body></html>');
        document.close();
      }
    }
  }
}

function close_scope() {
// provide a confirmation dialog box before closing the window
```

```
  if ( confirm( "Are you sure that you want to close the microscope?" ) )  {
    parent.close();
  }
}

//-->
</script>
</head>
<body bgcolor=#333333 text=#FFFFFF link="#CCFFFF" vlink="#FFCC99">
<center>
<form>
<font face="verdana,helvetica" size=1>java microscope viewer</font><br>
<select name="rock" onChange="scope(this)">
<option value="">Select a sample...
<option value="dw_vis">Densely Welded Bishop Tuff (visible light)
<option value="dw_pol">Densely Welded Bishop Tuff (polarized light)
<option value="pw_vis">Partly Welded Bishop Tuff (visible light)
<option value="pw_pol">Partly Welded Bishop Tuff (polarized light)
<option value="">------------------
<option value="close">Close Microscope
<option value="help">Help
</select>
</form>
</center>
</body>
</html>
```

**NOTE: We have quite a bit of code here! The menu created in the body of the document sends a message each time it is changed, sending a reference to the menu. The Javascript function looks at the `value` of the selected item. If it is empty, then we do nothing. If it is "help", we simply load the first page into the top frame. If it is "close" we call a JavaScript function that first displays a confirmation dialog box, and if the viewer click OK, it closes the entire window.**

**All of the major work is done when the menu sends the file name of an image. We then re-write all of the HTML in the top frame to load the applet with the specified file name, and write a caption using the text from the menu itself.**

8. **Save** this html file.
9. **Load** the `index.html` file into your web browser and check its functionality. If nothing happens when you change the menu in the bottom frame, triple-check your HTML with the sample code here.

Once our java page is working, we need to add a link that will open it in its own browser window, like we have done in the previous multimedia lessons.

1. Open the file `intro.html` in your text editor.
2. At the bottom of the section with the header "Volcanic Rocks" add this HTML after the sentence that reads:

```
A <b>thin section</b> is a layer of the rock cut so
thin that the light from a microscope shines through,
allowing us to see the structure of the rock.
<p>
<br clear=right>
```

so that it now reads:

```
A <b>thin section</b> is a layer of the rock cut so
thin that the light from a microscope shines through,
allowing us to see the structure of the rock.
<p>
To see a more detailed view of volcanic rocks,
try our <a href="../scope/index.html" onClick="window.open('../scope/index.html',
'scope', 'width=540,height=480,status,menubar'); return false"
onMouseOver="window.status='open the volcanic rock microscope';
return true">Volcanic Rock Microscope</a> (requires Java)
<br clear=right>
```

3. **Save** and **Reload** in your web browser. Test this and make sure it opens a browser window that then loads the Java equipped page.

# Check Your Work

Compare your web pages with this sample of how it should appear. If your pages are different from the sample or the hypertext links do not work correctly, review the text you entered in the text editor. Compare your work to the HTML of the samples (look for something like **Source** from your browser's **View** menu).

# Review

Review topics for this lesson:

1. What are some of the things you can do with a Java applet that make it useful?
2. What are some of the limitations of using Java?
3. What is the HTML for including a Java applet file? Why might one applet use different set of parameter tags?
4. How can you address situations where the person visiting your site cannot see the Java applet?
5. What is the benefit of using JavaScript to write the HTML for the Java applet?

# Independent Practice

Experiment with the ImageZoom applet using one of your own images. Try to see how it works if you adjust some of the parameters.

# More Information

Although Java may be a high-level computer programming language, you can find many web sites that provide free applets that you can use in your own pages.

- **Gamelan**
  http://www.gamelan.com/
- **FreeJava**
  http://www.freejava.com/
- **Java Boutique**
  http://www.builder.com/Programming/JavaCenter/
- **TheFreeSite**

http://www.thefreesite.com/freejava.htm

- **Freewarejava**
  http://www.freewarejava.com/
- **ZDNet Applet Library**
  http://www.zdnet.com/devhead/resources/scriptlibrary/applets/

For more Java resources, see our small, but high quality, reference list of Java tutorials. Also, see the CNET Java Center as well as Sun's main Java site.

---

# Coming Next....

Watch out as we completely revamp everything to take advantage of the next generation of web page design... let's rev up to HTML 4.0!

**GO TO.... | Lesson Index | previous: "Shockwave" | next: "HTML4.0" |**

---

**Writing HTML:** Lesson 29e: Small Cup of Java (to go)
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/tut29e.html

# WARNING!

You have just crossed the yellow tape into our construction zone. This lesson is currently in development.

Hard hats are NOT required, but we do ask that you use your browser back button and be patient while we work.

---

**Writing HTML**
©1994-1999 Maricopa Center for Learning and Instruction (MCLI)
Maricopa Community Colleges

The 'net connection at MCLI is Alan Levine
Questions? Comments? Visit our **feedback center**

URL: http://www.mcli.dist.maricopa.edu/tut/