



**UNESCO-NIGERIA TECHNICAL &
VOCATIONAL EDUCATION
REVITALISATION PROJECT-PHASE II**



NATIONAL DIPLOMA IN COMPUTER TECHNOLOGY



Web Technology **(COM225)**

**YEAR 2-SEMESTER 2
THEORY**

Version 1: December 2008

Table of Contents

INTERNET	12
BENEFITS OF INTERNET APPLICATION	13
<i>The World Wide Web (WWW):</i>	13
What is the WWW?	13
How Does the WWW Work?	13
How Does the Browser Fetch the Pages?	14
How Does the Browser Display the Pages?	14
Who is Making the Web Standards?	14
THE HISTORY OF WWW.	16
HOW A WEB PAGE WORKS.	16
What Is a Website?	17
THE HTML (How it works)	17
The Process of Web Design	20
Step 1: Planning	20
Step 2: Content Building	22
Step 3: Design	23
Step 4: Development	24
Failing Designs	24
Introduction to HTML	25
The HTML Skeleton	26
HTML Editors	27
Making Your First Web Page	27
HTML Elements	31
Basic HTML Tags	33
Headings	33
Paragraphs	33
Line Breaks	34
 or 	35
Comments in HTML	35
Recap on HTML Elements	35
HTML Attributes	36
HTML Tag Attributes	36

Use Lowercase Attributes.....	36
Always Quote Attribute Values	36
Text Alignments.....	37
HTML Text Formatting.....	38
FONT Tag: Good or Bad?	39
The HTML Tag.....	40
How to View HTML Source.....	41
HTML Character Entities	41
Character Entities.....	42
Non-breaking Space.....	42
<i>The Most Common Character Entities:</i>	43
HTML Links	44
The Anchor Tag and the Href Attribute	44
The Target Attribute.....	44
HTML Lists.....	46
Unordered Lists.....	46
Numbered Lists	47
Definition Lists.....	48
List Tags.....	50
HTML Images	51
Attributes.....	51
HTML Backgrounds	52
Bgcolor.....	52
Background.....	52
HTML Colors.....	54
Color Values	54
W3C Standard Color Names	54
Cross-browser Color Names.....	54
Cross-browser Color Values	54
HTML Color Names	56
HTML Forms and Input.....	60
Form Element	60
Input.....	61
Text Fields.....	62
Radio Buttons	62

Checkboxes.....	63
The Form's Action Attribute and the Submit Button	64
Form Tags	67
HTML Frames	68
The Frameset Tag	68
The Frame Tag	68
<i>Frame Tags</i>	69
HTML Tables	69
Tables and the Border Attribute	71
Headings in a Table	72
Empty Cells in a Table.....	73
Welcome, to my web page!	79
The Alt Attribute	81
Basic Notes - Useful Tips.....	82
Image Tags.....	82
Introduction to CSS.....	83
What is CSS?	83
CSS Demo.....	83
Styles Solve a Common Problem	83
Style Sheets Can Save a Lot of Work	84
Multiple Styles Will Cascade into One.....	84
Cascading Order.....	84
CSS Syntax	84
Grouping	85
The class Selector	85
Add Styles to Elements with Particular Attributes	87
The id Selector	87
CSS Comments	87
How to Insert a Style Sheet.....	88
External Style Sheet.....	88
Internal Style Sheet.....	88
Inline Styles	89
Multiple Style Sheets	89
CSS Background.....	91
CSS Text	92
CSS Text Properties	92

CSS Font	93
Examples	93
CSS Font Properties	94
CSS Border	96
CSS Border Properties	96
CSS Margin	98
CSS Margin Properties	98
CSS Padding	99
CSS Padding Properties	99
CSS List	100
CSS List Properties	100
CSS Table	101
CSS Table Properties	101
HTML Scripts	104
Adding a Script	105
Triggering a Script	105
Dynamic modification of documents	110
Calling an External Script	111
Hide Scripts from Older Browsers	111
Alternate Information for Older Browsers	111
Set a Default Scripting Language	112
Introduction to JavaScript	113
What You Should Already Know	113
What is JavaScript?	113
Are Java and JavaScript the Same?	113
What can a JavaScript Do?	114
The Real Name is ECMAScript	114
JavaScript Designs	115
How to Put a JavaScript Into an HTML Page.....	115
Example Explained.....	115
HTML Comments to Handle Simple Browsers	116
Where to Put the JavaScript.....	116
Using an External JavaScript.....	118
JavaScript Statements	119
JavaScript is Case Sensitive.....	119
JavaScript Statements	119

JavaScript Code	119
JavaScript Blocks.....	120
JavaScript Comments.....	120
JavaScript Multi-Line Comments	121
Using Comments to Prevent Execution	121
Using Comments at the End of a Line	121
JavaScript Variables.....	122
Example	122
Declaring (Creating) JavaScript Variables	122
Assigning Values to Undeclared JavaScript Variables.....	123
Redeclaring JavaScript Variables	123
JavaScript Arithmetic.....	123
JavaScript Operators	124
JavaScript Arithmetic Operators.....	124
JavaScript Assignment Operators	125
The + Operator Used on Strings	125
Adding Strings and Numbers.....	126
What is DHTML?	127
DHTML in IE 4	128
The style object of IE 4.....	128
Dynamic content	130
Dynamic content in IE 4.....	130
Moving elements around in the document.....	130
Moving elements in IE 4.....	130
Creating cross-browser DHTML	131
Creating a "cross-browser" layer.....	131
Browser sniffing- object detection.....	132
Introduction to Web Multimedia.....	135
What is Multimedia?	135
Browser Support	135
Multimedia Formats	136
Multimedia Sound Formats.....	136
The MIDI Format.....	136
The RealAudio Format	137
The AU Format	137
The AIFF Format	137

The SND Format.....	137
The WAVE Format.....	137
The MP3 Format (MPEG)	138
What Format To Use?	138
Multimedia Video Formats	138
The AVI Format.....	138
The Windows Media Format	139
The MPEG Format.....	139
The QuickTime Format.....	139
The RealVideo Format.....	139
The Shockwave (Flash) Format	139
Playing Sounds On The Web	140
Inline Sound	140
Using A Helper (Plug-In)	140
Using The <bgsound> Element	140
Using The Element	141
Using The <embed> Element	141
Using The <object> Element	142
Using A Hyperlink	142
Playing Videos On The Web	142
Inline Videos	143
Using A Helper (Plug-In).....	143
Using The Element.....	143
Using The <embed> Element	144
Using The <object> Element	144
Using A Hyperlink.....	144
Windows Multimedia Formats	145
The ASF Format	145
The ASX Format.....	145
The WMA Format.....	146
The WMV Format.....	146
Other Windows Media Formats.....	146
Introduction to Flash	146
What you should already know	146
What is Flash?	147
Flash vs. Animated Images and Java Applets	147

Who can View Flash?	147
Who can Create Flash Movies?	147
Where to Start?	148
Flash in HTML	148
Flash Embedded in HTML	148
Let the Flash Program do the Work.....	149
Introduction to FrontPage	150
What is FrontPage?	150
What's new for 2003?	150
Changes in Design	150
Changes in Coding	150
Extending FrontPage.....	151
The FrontPage Interface.....	151
HTML Basics.....	151
An HTML overview	151
Naming files.....	151
Creating New Pages.....	153
Opening a New Page.....	153
Adding new pages.....	153
Saving pages	153
Opening pages.....	154
Using Page templates.....	154
Page Properties.....	155
Using Page Properties dialog box	155
Using the General tab.....	156
Using the Formatting tab	156
Using the Advanced tab.....	157
Creating a New Web File.....	157
Using Web site templates.....	157
Opening a web site.....	158
Closing a web site	159
Deleting a web site.....	159
Building a Web Site	161
Using a Wizard or Template	161
Creating a New Web Site.....	161
Creating a Web Site with a Wizard.....	162
Creating a Web Site using a Template.....	170

Web Location Options	172
Using the Web Location Options area	172
Choosing Web Hosting Service Providers.....	173
Choosing an ISP.....	175
Using a Personal Web Server	175
Designing a Web Site.....	175
Creating a new Page using a Template	176
Checking Spelling.....	177
Checking Hyperlinks	178
Setting Tasks.....	179
Setting Permissions	180
Publishing	180
Publishing a Site	181
Previewing the web site	181
Trouble-Shooting	181
Using a server or disk-based web sites	182
Using web servers	182
HTTP publishing.....	183
FTP publishing.....	184
Introduction to XML	186
What is XML?	186
The Difference Between XML and HTML	186
XML is Just Plain Text.....	186
With XML You Invent Your Own Tags.....	187
XML is Not a Replacement for HTML.....	187
XML is a W3C Recommendation.....	187
XML is Everywhere	187
How Can XML be Used?.....	188
XML Separates Data from HTML.....	188
XML Simplifies Data Sharing	188
XML Simplifies Data Transport	188
XML Simplifies Platform Changes	189
XML Makes Your Data More Available	189
XML is Used to Create New Internet Languages	189
If Developers Have Sense.....	189
XML Tree	190

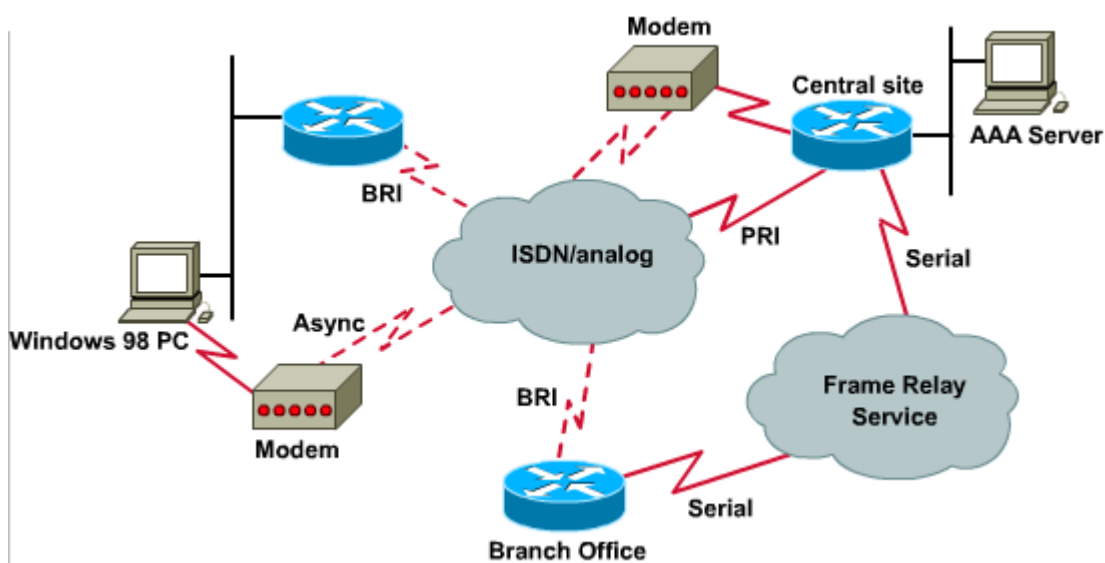
An Example XML Document.....	190
XML Documents Form a Tree Structure	191
Example:	191
XML Syntax Rules	192
All XML Elements Must Have a Closing Tag.....	192
XML Tags are Case Sensitive.....	193
XML Elements Must be Properly Nested	193
XML Documents Must Have a Root Element	193
XML Attribute Values Must be Quoted	194
Entity References	194
Comments in XML	195
With XML, White Space is Preserved	195
XML Stores New Line as LF.....	195
XML Elements.....	195
What is an XML Element?	195
XML Naming Rules.....	196
Best Naming Practices	196
XML Elements are Extensible	197
XML Attributes.....	198
XML Attributes.....	198
XML Attributes Must be Quoted	198
XML Elements vs. Attributes	199
My Favorite Way	199
Avoid XML Attributes?	200
XML Attributes for Metadata	200
XML Validation.....	201
Well Formed XML Documents	201
Valid XML Documents.....	202
XML DTD	202
XML Schema	202
A General XML Validator	203
XML Validator.....	203
XML Errors Will Stop You	203
Syntax-Check Your XML.....	203
Syntax-Check an XML File	204
Validate Your XML Against a DTD	204
Viewing XML Files	205

Viewing XML Files	205
Viewing an Invalid XML File.....	206
Other XML Examples.....	206
Why Does XML Display Like This?	206
Displaying XML with CSS	206
Displaying your XML Files with CSS?	206
Displaying XML with XSLT	207
Displaying XML with XSLT	208
Transforming XML with XSLT on the Server	208

WEEK One

INTERNET

This is a virtual world provided by networks of computers with multi-users. It is an international network of networks of computers linking different types of users: Academic, Industries, Government, Health Institutions, military, individuals, etc, for the purpose of sharing information. As a communication network among computers, the internet allows you to locate and retrieve information on other computers linked to the internet as well as send messages electronically to and from other people elsewhere on the internet. Whenever Internet software application is used, the client software will either be on your personal computer, the computer you log onto for access to the internet (your host), or yet another computer to which you connect in order to use client software you may not have on your computer. As you navigate through the Internet, you will find yourself logged onto different host computers, sometimes gaining access to different client programs and also accessing different servers; it can be complicated. Fortunately, the purpose of advanced Internet software is to hide these complexities from users so as to achieve success.



The internet has six application protocols: Electronic mail (E-mail), World Wide Web (Hypertext Transfer Protocol or HTTP), Gopher, Telnet, File Transfer Protocol (FTP), Each of these application protocol has special client software, and many web browsers, such as Netscape from communication and internet Explorer from Microsoft, which are capable of reading and displaying data from all the applications.

BENEFITS OF INTERNET APPLICATION

- (a) Downloading of information
- (b) Advertisement
- (c) Accessing newspapers, magazines and academic journals
- (d) On-line banking
- (e) Accessing international media (CNN, BBC, VOA)

The World Wide Web (WWW):



What is the WWW?

- WWW stands for the **World Wide Web**
 - The World Wide Web is most often called **the Web**
 - The Web is a network of computers **all over the world**
 - All the computers in the Web can **communicate with each other**
 - All the computers use a **communication standard called HTTP**
-

How Does the WWW Work?

- Web information is stored in documents called **Web pages**
- Web pages are files stored on computers called **Web servers**
- Computers reading the Web pages are called **Web clients**

- Web clients view the pages with a program called a **Web browser**
 - Popular browsers are **Internet Explorer and Mozilla Firefox**
-

How Does the Browser Fetch the Pages?

- A browser fetches a Web page from a server **by a request**
 - A request is a standard HTTP request containing **a page address**
 - A page address looks like:
http://www.someone.com/page.htm
-

How Does the Browser Display the Pages?

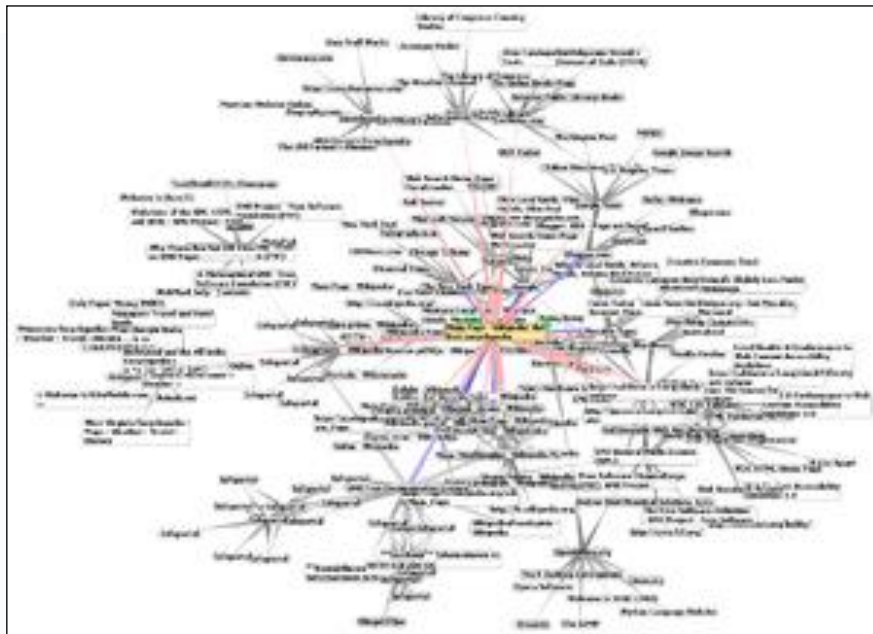
- All Web pages contain instructions on **how to be displayed**
 - The browser displays the page by **reading these instructions**
 - The most common display instructions are called **HTML tags**
 - The HTML tag for a paragraph looks like this: **</p>**
 - A paragraph in HTML is defined like this **<p>**This is a Paragraph**</p>**
-

Who is Making the Web Standards?

- The Web standards are **not made up** by Netscape or Microsoft
- The rule-making body of the Web is the **W3C**
- W3C stands for the **World Wide Web Consortium**
- W3C puts together specifications for **Web standards**
- The most essential Web standards are **HTML, CSS and XML**
- The latest HTML standard is **XHTML 1.0**

The World Wide Web (WWW): World Wide Web (often termed simply as "Web") is the most exciting new tool for the Internet. It is based on the technology called hypermedia. With hypermedia, information in one document can be linked to another; related information can consist of not only text but graphics, audio and video information as well. WWW is an ambitious, exciting and powerful attempt to link connected information wherever it may be located on the Internet, allowing the user to easily access and retrieve related files. The terms *Internet* and *World Wide Web* are often used in every-day speech without much distinction. However,

the Internet and the World Wide Web are not one and the same. The Internet is a global data communications system. It is a hardware and software infrastructure that provides connectivity between computers. In contrast, the Web is one of the services communicated via the Internet. It is a collection of interconnected documents and other resources, linked by hyperlinks and URLs. These hyperlinks and URLs allow the web servers and other machines that store originals, and cached copies, of these resources to deliver them as required using HTTP (Hypertext Transfer Protocol). HTTP is only one of the communication protocols used on the Internet.



Graphic representation of a minute fraction of the WWW, demonstrating hyperlinks

Web services also use HTTP to allow software systems to communicate in order to share and exchange business logic and data.

Software products that can access the resources of the Web are correctly termed *user agents*. In normal use, web browsers, such as Internet Explorer, Firefox and Apple Safari, access web pages and allow users to navigate from one to another via hyperlinks. Web documents may contain almost any combination of computer data including graphics, sounds, text, video, multimedia and interactive content including games, office applications and scientific demonstrations.

THE HISTORY OF WWW.

Using concepts from earlier hypertext systems, the World Wide Web was begun in 1989 by English scientist Tim Berners-Lee, working at the European Organization for Nuclear Research (CERN) in Geneva, Switzerland. In 1990, he proposed building a "web of nodes" storing "hypertext pages" viewed by "browsers" on a network, and released that web in 1992. Connected by the existing Internet, other websites were created, around the world, adding international standards for domain names & the HTML language. Since then, Berners-Lee has played an active role in guiding the development of Web standards (such as the markup languages in which Web pages are composed), and in recent years has advocated his vision of a Semantic Web.

HOW A WEB PAGE WORKS.

Viewing a Web page on the World Wide Web normally begins either by typing the URL of the page into a Web browser, or by following a hyperlink to that page or resource. The Web browser then initiates a series of communication messages, behind the scenes, in order to fetch and display it.

First, the server-name portion of the URL is resolved into an IP address using the global, distributed Internet database known as the domain name system, or DNS. This IP address is necessary to contact and send data packets to the Web server.

The browser then requests the resource by sending an HTTP request to the Web server at that particular address. In the case of a typical Web page, the HTML text of the page is requested first and parsed immediately by the Web browser, which will then make additional requests for images and any other files that form a part of the page. Statistics measuring a website's popularity are usually based on the number of 'page views' or associated server 'hits', or file requests, which take place.

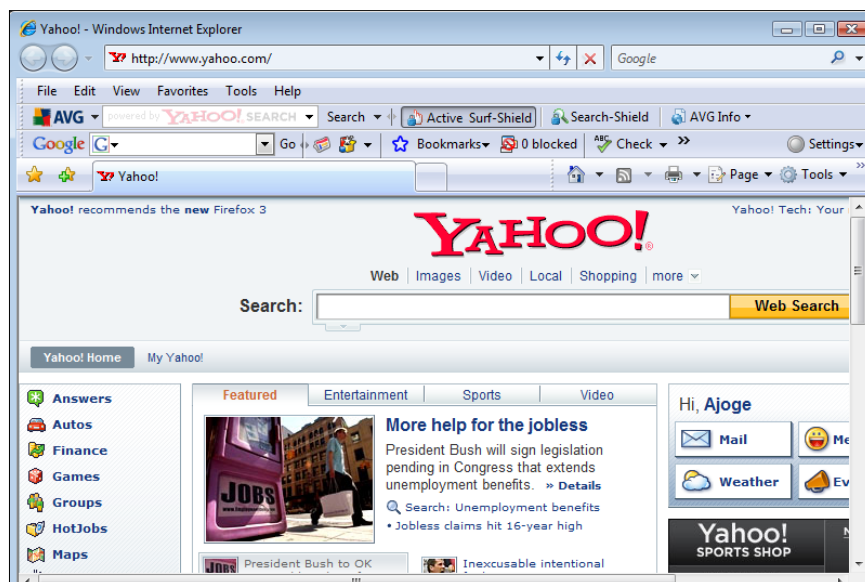
Having received the required files from the Web server, the browser then renders the page onto the screen as specified by its HTML, CSS, and other Web languages. Any images and other resources are incorporated to produce the on-screen Web page that the user sees.

Most Web pages will themselves contain hyperlinks to other related pages and perhaps to downloads, source documents, definitions and other Web resources. Such a collection of useful, related resources, interconnected via hypertext links, is what was dubbed a "web" of information. Making it available on the Internet created what Tim Berners-Lee first called the **WorldWideWeb**.

What Is a Website?

A website is a collection of web pages (documents that are accessed through the Internet), such as the yahoo website below. A web page is what you see on the screen when you type in a web address, click on a link, or put a query in a search engine. A web page can contain any type of information, and can include text, color, graphics, animation and sound.

When someone gives you their web address, it generally takes you to their website's home page, which should introduce you to what that site offers in terms of information or other services. From the home page, you can click on links to reach other sections of the site. A website can consist of one page, or of tens of thousands of pages, depending on what the site owner is trying to accomplish.



Yahoo website

THE HTML (How it works)

HTML (HyperText Markup Language) is a web programming language designed to create web documents or web pages. Based upon SGML (Standard Generalized Markup Language), HTML's basic concept involves the use of "tags". These "tags", or "mark up" alert the browser that the document contains hypertext so it can be interpreted and rendered as a web page document. All HTML documents consist of a mix and match of HTML "tags" and regular text. Tags only aid in describing the document content or text, and thus leave the actual appearance and layout decisions for a web browser to handle when the web page is opened.

HTML documents are plain-text files created using any basic or high-level text editor, such as Notepad, TextPad, Microsoft Word, or any other HTML authoring program. When you create an HTML document, you must save it with a .html or .htm extension. By default, most text editors save documents with a .txt extension. The .html or .htm extension allows the document to be rendered (using an HTML interpreter by browser) and displayed by a browser.

Unfortunately for web designers, web documents are browser dependent. Different browsers display content differently. A document may look crisp and clean in Internet Explorer, but it may have a slightly different look in Netscape, or vice versa. Web page designers should make every attempt to create portable HTML documents that can be opened by many different web browsers while showing little or no visual differences. Reliable HTML web pages are created by following all syntax rules and understanding which tags are supported by all web browsers. For more information on creating error-free HTML code and performing validation checks, visit The World Wide Consortium (W3C), which provides free HTML resources.

What does a person need to create a web page? A text editor, and a web browser. In most cases, a web page or web site, which is a collection of related web pages, should initially be designed locally on a computer, and then once completed, the web documents and files may be uploaded for publishing on the World Wide Web. This makes web site or web page creation extremely easier than trying to edit existing documents on the WWW. Before reading further and learning actual HTML code, you should become familiar with the following terms:

HTML	- short for HyperText Markup Language; basic programming language of the World Wide Web based upon SGML (Standard Generalized Markup Language)
web browser	- application capable of interpreting and rendering HTML code and other web programming languages.
URL	- (uniform resource locator) - the address to any web site or web page document that is apart of the World Wide Web (WWW).
hyperlink, link	- text, image, or object in a web page document that "links" or "points" to another document on the World Wide Web.
element	- a fundamental component of structure in a web document; web pages are ultimately divided into

	many individual elements.
tag	- used to denote various elements in a document; signals a command or instruction for a web browser describing the type of content to be rendered.
attribute	- additional information included inside the start tag of an element.
web document	- actual web page text file with an extension of .html or .htm that is capable of being displayed by a browser.

WEEK Two

The Process of Web Design

So many designers, especially newer, freelance workers, jumble up the process of design. This is not a very good idea as it leads to failures which the designers may not even be aware of. There are series of steps that a good designer follow to achieve a perfect design. We'll be focusing on each of the aspects of the process of design in just a bit, but first, let's look at an overview.

- **Planning**

Before you even get a single idea in your head about what your latest Web design is going to look like, you need to do all the appropriate planning. Mostly, this includes knowing the answers to important questions.

- **Content Building**

You need to build all of the content of your site after you have a clear plan. This doesn't necessarily mean getting every detail down, but you need to know what your content will be and where it will go.

- **Designing**

Now you'll design the Web site. This is a crucial step, of course, but if you've done the planning and content building first, this part will be the most fun, and most likely the most profitable! But keep in mind that there are several key considerations you'll be taking into account, so you won't be getting wild with your digital paintbrushes here.

- **Development**

Once you've finalized your design, you'll turn it into XHTML/CSS/PHP/MySQL and whatever else your site needs. Development typically refers to the technical side of Web design: the coding and backend stuff.

As you can see (pay attention to this part, it's important!), **designing the site comes late in the game.** Many designers make the mistake of wanting to jump right into the design step, because that's where a designer's talents shine. But if you avoid or simplify the planning and content building steps, you will end up with a design that probably fits the site pretty well, but will ultimately fail in helping the site achieve its goals.

Therefore, it's important that you know and follow this process in your professional work. Let's take a look at the process in greater detail.

Step 1: Planning

There's a lot of work ahead of you before you actually get to start drawing, coloring, laying out, and generally designing your Website.

First, you need the answers to these questions. You will know these answers if you are designing a site for yourself, or your client will know them.

You need to know the answers to these:

- **What is your goal with this site?**
Are you trying to sell something? Deliver a message? Share information? Keep in touch? Etc.
- **Who are your site's visitors?**
In other words, who is your target audience? This can be students for a website developed for online students registration.
- **What do you want your visitors to do?**
Usually, if you're getting visitors to your site, you want them to actually *do* something. Some answers to this question could be, "buy a product," "sign up for a newsletter," or "learn how to build better Websites." This is an important question; make sure you have a good, clear answer for it.
- **Why should they do it?**
Look at this site from the viewer's perspective. Why should they be interested in your site? What's in it for them?

If you or your client don't have clear answers for the above questions, it's time to do some research. Find some of your customers and get these answers or make some calls. Visit other sites that are similar to yours in idea and find out what kind of crowd they're drawing. If this is the type of crowd you'll be working with, then you'll have some answers.

Once you have good, clear answers to all of these questions, you can start building content for your site and start making some actual design decisions. With just these answers, you can determine the basics, like:

- Color
- Layout
- Font family
- Font size

Colors? Layouts? Fonts? Ah, here we go; this is starting to sound more like Web design! If you know, for example, that your goal is to provide wildlife safety tips to new campers, you will want a layout that allows you to provide clear, readable text. Because your tips will come mostly in the form of text and illustrations, you'll want plenty of body space and a font that looks good on screen and spaced well with CSS (Arial is a good default for something like this). Since you're dealing with wildlife-related material, your colors might mimic those of the woods, with deep greens, sandy and earthen tones, and dark wood colors.

But don't jump the gun and open Photoshop just yet! You're still not quite ready to start drawing and coloring. Next, you want to build the content of your site.

Step 2: Content Building

There's a phrase that's often thrown around in the Web development world: "content is king." This is true for most Websites out there. Most Websites want to be found and one key way sites are discovered is through search queries. Search engines frequently "spider" their database of Websites for new, clear, up-to-date, and original content, and Websites that have good content are rewarded with higher ranking and thus they are found more often.

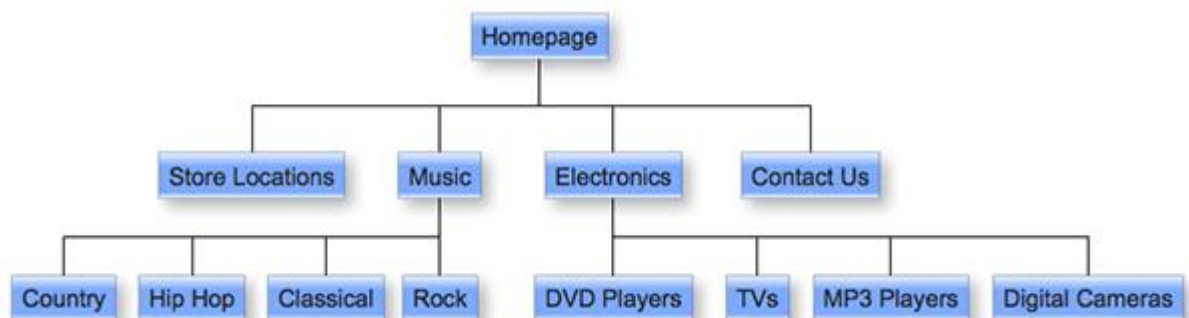
It's no wonder, then, that content building is such an important step in Web design. With the answers you got from Step 1: Planning, you should know what kind of content you will want to start building. If your goal is to sell music and electronics, and your target audience is college students looking to buy things like iPods and digital cameras, then you'll want to write some content that's light-hearted, down-to-earth, and to the point while still building on the popular "lifestyle" trend that sells gadgets like these so well.

Or, as another example, if your target audience is a group of professionals in the medical research field and your goal is to pose challenging new ideas for collaboration or discussion, you'll want to write content that comes off as something of a technical blog.

Ultimately, though, the most important thing you want to do in this step is get all your ideas for content down on paper or in a text file. You don't need to write the ads for your music and electronics right now, for example, and of course you wouldn't want to spend the time in this step to write a long medical journal entry. For now, it's enough just to get all of your content down in shorthand. Just be sure you get it all.

Once you have all your content in a place where you can look at it, you'll want to build your sitemap. This is the final step before you actually start designing the site. You can use a computer program to draw your sitemap like I did below, but most of the time a pencil and paper works best for this part.

Let's take a look at a small sample sitemap for the music and electronics store we talked about earlier.



Here we can see that our homepage links to four other pages: Store Locations, Music, Electronics, and Contact Us. The Music and Electronics pages each link to different subsections. The Music page lets you choose a genre of music to buy, and the Electronics page lets you browse through different gadgets.

When designing your sitemap, it's important to remember the 3-click rule. Simply, the 3-click rule states that a visitor should never have to click more than three times on your site to do anything. In our sample sitemap, we can see that if a visitor wanted to get our other store locations, they would just need 1 click on the homepage. Or, if they wanted to browse through rock music to buy, they would just need to click Music on the homepage, then Rock on the Music page. Just 2 clicks.

While building your sitemap, you should make sure that every box (or page) on the sitemap will hold some of the content you outlined earlier. All of your content should have a place on your site, and you need to be able to visually see on what pages you'll put each bit of your content.

And now that we have our planning and content building done, let's get designing!

Step 3: Design

Many designers make the mistake of skipping steps 1 and 2, jumping instead right here to Step 3: Design. But if you got here after having done the necessary work beforehand, then you're on the right track!

In Step 1: Planning, you probably already got a good idea about what types of layouts and design you want to use. Let your creativity flow here. Remember, there are multiple layouts you will probably need for your design. You'll want a layout for your homepage, which, according to your sitemap, will just contain some introductory information or highlights, some links, and, depending on your goals, maybe some exciting imagery.

You'll also want a layout for your transitional pages. Transitional pages are pages that just shuttle the user to more important things. In the sample sitemap above, the Music and Electronics pages are just transitional pages. The Music page would be small, and contain links to the different genres of music below (Country, Hip Hop, Classical, and Rock). The Electronics page would also be small, and just contain small bits of information while it links to the more important stuff, like the DVD Players, TVs, MP3 Players, and Digital Cameras pages. Transitional page layouts should match the whole site, but also be simple so users can spend as little time on them as possible.

You probably don't want to hear this, but even at this point, you might consider leaving Photoshop closed for a bit. Instead, try drawing several layouts for your site on paper first. It's a good idea

to build multiple layouts just to see how they "feel" with the site you're building, and using Photoshop to build lots of layouts that you might not even use is time consuming. It's much faster to draw some quick and dirty layouts on paper to get a rough feel for which way you want to go with the design.

Make sure you finalize your design. Once you have your design ready, there's no going back.

Step 4: Development

If you've finalized your design (either to yourself or with your client), then you can begin developing the site. Depending on how you want the site to work, this could involve such complexities as PHP and MySQL backend programming, or it could just be as simple as some basic HTML and CSS or FrontPage. Either way, this step involves the actual coding of the site.

Some designers are solely designers and deal with only the basics of HTML and CSS, opting to work with a partner or hire someone to do the coding of a site. Other designers know the development and coding side just as well as the design side, but these designers aren't altogether too common.

Either way, we won't go into the development of the site here. For the purposes of this article, it's enough to say that this is the final step, after the design is done, in the creation of a Website.

Failing Designs

As a designer, do you know where your work *really* fits in the process of design?

We all love Web design. Looking at a blank white box on a computer screen and using only your mind's eye, a mouse, and a keyboard to transform it into a living, breathing Website is no minor feat, and there is undoubtedly a creative rush when it comes to doing something like this. Web design can be a strong artistic outlet and it inevitably brings with it the joy that comes with looking at your finished work and presenting it to others.

Failing design is not necessarily poor design.

This means that a "failing design" is not necessarily an ugly Website. Rather, a failing design is one that fails to function properly for the site where it's used. A design can be very beautiful, but be terrible for its purpose.

If your (or your client's) goal with a Website is to sell something or generate excitement over a specific idea or campaign, you'll want to design accordingly. For a site like this, you don't want to use a calm ocean picture as your header with deep ocean blues and blacks in

your design. Sure, you may be able to make a very pretty design this way, but it's a failing design because it doesn't supplement the site's goal or message.

The above example is a gross exaggeration, but it serves to show why so many designs fail. The problem lies in the misunderstanding of the process of design.

Introduction to HTML

HTML is short for **H**yper**T**ext **M**arkup **L**anguage. Web pages are built using html tags.

So that you understand what a "tag" is, here is an example:

<html>

Each tag consists of the containers, which are the lesser than (<) and greater than (>) arrows, and the HTML element within them. The arrows and the HTML element together are commonly called an HTML *tag*, or an HTML *command* by some. The example tag above is the beginning of an HTML document. It tells the browser the document is a page written in the HTML language so it can be interpreted and displayed according to the specifics of that language.

There are dozens of tags, but only a few that you have to know to make a simple web page. If it looks confusing, just play along, it really isn't that difficult. These lessons in the basic section will walk you through it one step at a time. In about 20 minutes you'll have made your first web page!

To begin with, you'll need a browser, (d-oh!) and a text editor. There are HTML editors that make web page building easier for some people, but for this tutorial we'll assume you don't have one. If you do and want to use it, feel free, as long as it allows you to enter code directly it doesn't matter what you use. It is best to learn to code by hand even if you plan on using an HTML editor later on. That's the only way you'll know how to troubleshoot problems, and problems do arise from time to time even for the most skilled webmasters.

We will use Notepad, so will refer to that later in this tutorial. Notepad comes with every Windows PC, and if you learn the simple language, it's all you really need. You'll find it in your Accessories folder.

The HTML Skeleton

Below are the tags every HTML document must have. In the past you could type the HTML tags in either lower case or capital letters, but there are a few that cause problems in certain browsers when entered in upper case.

Nowadays, because other languages such as XHTML *require* that tags be in lower case, and because many people are calling for the same requirement for HTML, I strongly urge you to code in lower case at all times. It may seem fussy, but cross-platform display consistency is the goal, and that can only be accomplished by having standards everyone follows.

These tags are the bare necessities:

```
<html>
<head>
<title> Title Text Goes Here </title>
</head>
<body>
Content Area
</body>
</html>
```

That is the basic HTML skeleton, but it's not really as confusing as it looks. I've shown the HTML tags in red, but the tags themselves do not show up on a web page unless an error is made. They are in the "source code," which is simply the underlying text file that contains the HTML code and content you want displayed for people see.

The browser interprets the HTML tags in the source code and displays your content according to those tags. It may help you to understand the concept by thinking of HTML tags as little instruction sets that inform the browser how to display the actual content.

If the above code were a real web page, the words "Content Area" would be the only thing that would show up in a browser window.

Here's a brief explanation of each tag...

<html>

Identifies the language used (file type) for the browser, in this case, a web page written in HTML language.

<head>

Acts as a container for the page title and meta data. The title is necessary, but meta data is not.

<title>

The title is the name of the page. The title shows up in the title bar at the top of your browser, as the text for bookmarks unless it's changed, and in some search engines as the link text. If you look at the very top of your browser window you should see "The HTML Skeleton Explained" as the title of this web page.

</title>

Closes the title. The forward slash (/) in front of the HTML element means that command is now cancelled.

</head>

Closes the head section.

<body>

Between the opening and closing body tags is where you place the actual content you want displayed to the public.

</body>

Closes the body section

</html>

Closes the html element, end of page.

HTML Editors

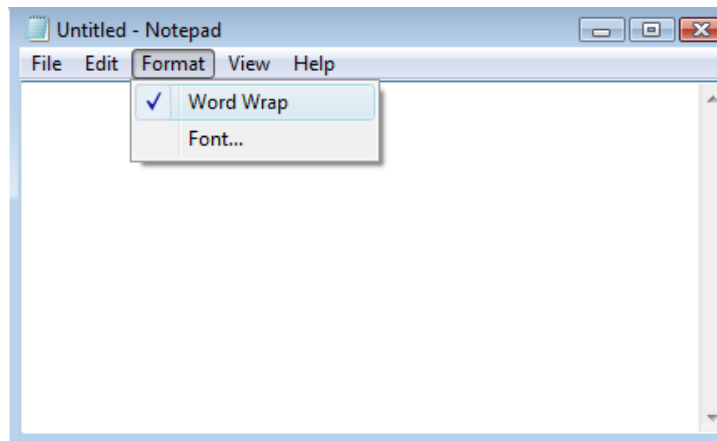
You can easily edit HTML files using a WYSIWYG (what you see is what you get) editor like FrontPage or Dreamweaver, instead of writing your markup tags in a plain text file.

However, if you want to be a skilful Web developer, it is strongly recommended that you use a plain text editor such as Note Pad.

Making Your First Web Page

Are you ready to make a quick web page? You need Notepad or another text editor.

If you're using Notepad, first go to the *Format* menu and put a check mark by *Word Wrap*. This keeps your text from appearing as one long line that you'd have to scroll sideways to see.



As noted earlier, an HTML document starts off with:

<html>

This tells the browser it's an HTML document.

Go ahead and type **<html>** into Notepad now. Next is the:

<head>

...tag. Now that you're an old hand at typing HTML commands, type the **<head>** tag under your **<html>** tag.

Note: Technically, you don't actually have to type the head tag under the html tag, but it will help you to visualize the structure. Start code on new lines because it breaks up the source code with white space, making it much easier to locate the places you want to edit later on. As your page complexity grows with your skill level, this habit can be a valuable time-saver.

There are several items that can go in between the opening head tag and closing head tag, but for a simple page like the one we're creating, we're only concerned with one, and that is the:

<title>

...tag. The title tag, as you might guess, contains the title of your web page. The title shows up in the title bar at the top of the browser window, as the text when someone bookmarks your page, and as the link text of a search engine query on many search engines. If you look at the top of your browser you should see the words, "Making Your First Web Page" in the title bar because that's the title we gave this page.

You'll also use your first cancel command.

</title>

The *cancel* command, also called the *close* command, means you are cancelling a previously opened HTML element (tag). In this case, it's telling the browser that the TITLE of the page is complete, or cancelled. The forward slash before an HTML element tells the browser that element has been cancelled. Note that an *element* and *command* are the same thing. Go ahead and add:

```
<title>My First Web Page</title>
```

...to your HTML document under the HEAD tag now.

Since that is the only element we're adding to the head section, go ahead and type the **</head>** command under your title line to cancel the HEAD element.

Next is the:

```
<body>
```

...tag. The area between the opening and closing body tags is where you put the content you want to show up on the web page when viewed with a browser. Other HTML elements are used within this area as well, which allow you to add images, create paragraph spaces, and to code other display instructions. These other formatting tags do not show up on the web page, but merely tell the browser how to display the information you want to present. When you've completed this tutorial you can go to the individual tutorials and add whatever elements you wish to play around with to your first web page.

After the BODY tag is where you put the content you want to show on a web page. Your pictures, jokes, links and all else goes here. So for now, just enter the **<body>** tag under the **</head>** tag and type a brief message for demonstration purposes.

The last thing to do is to cancel the BODY and HTML elements. In HTML we always cancel tags in the reverse order they were opened, so type:

```
</body>
```

```
</html>
```

Your test page should now look like this:

```
<html>
```

```
<head>
```

```
<title>My First Web Page</title>
```

```
</head>
```

```
<body>
```

Its fun to know the basics of webpage and to have used the little I have learnt to create a simple but fascinating web page. Thanks to all that have contributed in one way or the other for the development of this material.

```
</body>
```

```
</html>
```

The last thing to do is to save the page you just made. You can call this test page anything, but when you build a web site for real, your home page should always be called **index.html**.

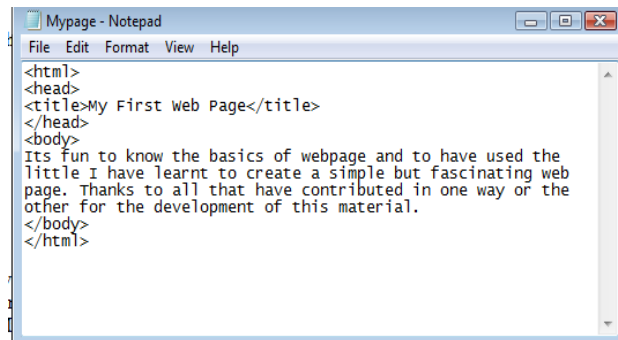
It should be noted that some people name their documents with a *.htm* extension instead of *.html*, but not all servers look for the *.htm*

extension when it can't find .html, so it's wise to get in the habit of saving them as .html right now.

Note: When saving your document, enclose it in quotation marks to prevent your system from adding .txt (for text file) to the end. In the box where you type the name you want to save the file under, type it as: **"Mypage.html"**

...and be sure to include the quotation marks around the name and file extension.

Your codes in the Notepad editor should be of the form below



```

File Edit Format View Help
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
Its fun to know the basics of webpage and to have used the
little I have learnt to create a simple but fascinating web
page. Thanks to all that have contributed in one way or the
other for the development of this material.
</body>
</html>

```

Now that you've saved it, open it in your browser and see what you've made! To do that just hit Control and (the letter) O on your keyboard then hit the Browse (IE). Find the trial file "Mypage.html" page you just made and click it, then click OK. Then just hit the back button to return.

Frequently Asked Questions

Q: After I have edited an HTML file, I cannot view the result in my browser. Why?

A: Make sure that you have saved the file with a proper name and extension like "c:\mypage.html". Also make sure that you use the same name when you open the file in your browser.

Q: I have edited an HTML file, but the changes don't show in the browser. Why?

A: A browser caches pages so it doesn't have to read the same page twice. When you have modified a page, the browser doesn't know that. Use the browser's refresh/reload button to force the browser to reload the page.

Q: What browser should I use?

A: You can do all the training with all of the well-known browsers, like Internet Explorer, Firefox, Netscape, or Opera. However, some of the examples in our advanced classes require the latest versions of the browsers.

Q: Does my computer have to run Windows?

A: You can do all your training on a non-Windows computer like a Mac.

HTML Elements

HTML documents are text files made up of HTML elements. HTML elements are defined using HTML tags.

HTML Tags

- HTML tags are used to mark-up HTML **elements**
 - HTML tags are surrounded by the **two characters < and >**
 - The surrounding characters are called **angle brackets**
 - HTML tags normally **come in pairs** like and
 - The first tag in a pair is the **start tag**, the second tag is the **end tag**
 - The text between the start and end tags is the **element content**
 - HTML tags are **not case sensitive**, means the same as
-

WEEK Three

Basic HTML Tags

The most important tags in HTML are tags that define headings, paragraphs and line breaks.

Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the largest heading. <h6> defines the smallest heading.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

HTML automatically adds an extra blank line before and after a heading.

Paragraphs

Paragraphs are defined with the <p> tag.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

HTML automatically adds an extra blank line before and after a paragraph.

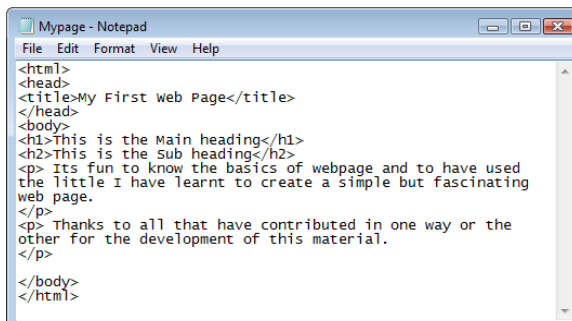
Practice

We may demonstrate the Heading and Paragraph tags by including them in "Mypage" that we designed in Week two.

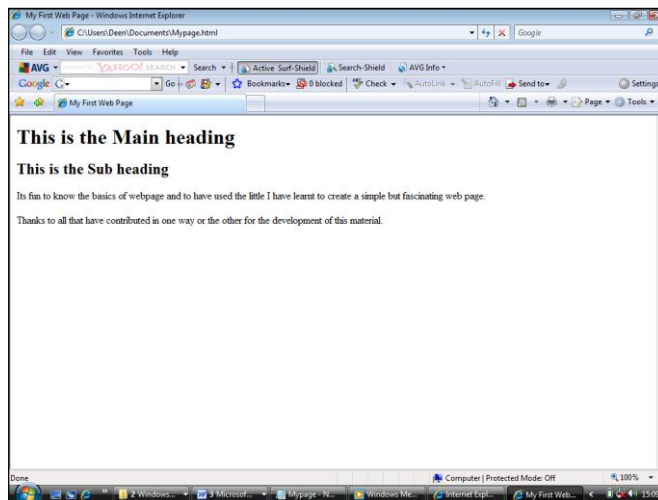
Let us add the followings to the body tag

```
<h1>This is the Main heading</h1>
<h2>This is the Sub heading</h2>
<p> Its fun to know the basics of webpage and to have used the little I have
learnt to create a simple but fascinating web page.
</p>
<p> Thanks to all that have contributed in one way or the other for the
development of this material.
</p>
```

It should look as below in the editor



When run on the browser it appear as below



Don't Forget the Closing Tag

Note that paragraphs can be written without end tags `</p>`:

```
<p>This is a paragraph
<p>This is another paragraph
```

The example above will work in most browsers, but don't rely on it. Future version of HTML will not allow you to skip ANY end tags. Closing all HTML elements with an end tag is a future-proof way of writing HTML. It also makes the code easier to understand (read and browse) when you mark both where an element starts and where it ends.

Line Breaks

The `
` tag is used when you want to break a line, but don't want to start a new paragraph. The `
` tag forces a line break wherever you place it.

```
<p>This <br> is a para<br>graph with line breaks</p>
```

The `
` tag is an empty tag. It has no end tag like `</br>`, since a closing tag doesn't make any sense.

 or

More and more often you will see the
 tag written like this:

Because the
 tag has no end tag (or closing tag), it breaks one of the rules for future HTML (the XML based XHTML), namely that all elements must be closed.

Writing it like
 is a future proof way of closing (or ending) the tag inside the opening tag, accepted by both HTML and XML.

Comments in HTML

The comment tag is used to insert a comment in the HTML source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

Recap on HTML Elements

- Each HTML element has **an element name** (body, h1, p, br)
 - The **start tag is the name** surrounded by angle brackets:
<h1>
 - The **end tag is a slash and the name** surrounded by angle brackets </h1>
 - **The element content** occurs between the start tag and the end tag
 - Some HTML elements have no content
 - Some HTML elements have no end tag
-

Basic Notes - Useful Tips

When you write HTML text, you can never be sure how the text is displayed in another browser. Some people have large computer displays, some have small. The text will be reformatted every time the user resizes his window. Never try to format the text in your editor by adding empty lines and spaces to the text.

HTML will truncate the spaces in your text. Any number of spaces count as one.

In HTML a new line counts as one space.

Using empty paragraphs <p> to insert blank lines is a bad habit. Use the
 tag instead. (But don't use the
 tag to create lists. Wait until you have learned about HTML lists.)

HTML automatically adds an extra blank line before and after some elements, like before and after a paragraph, and before and after a heading.

We use a horizontal rule (the `<hr>` tag), to separate the sections in our tutorials.

Basic HTML Tags

The basic HTML tags in the table below displays the summary of above with additions. You will learn more about HTML tag attributes as we proceeds the tutorial.

Tag	Description
<code><html></code>	Defines an HTML document
<code><body></code>	Defines the document's body
<code><h1></code> to <code><h6></code>	Defines header 1 to header 6
<code><p></code>	Defines a paragraph
<code>
</code>	Inserts a single line break
<code><hr></code>	Defines a horizontal rule
<code><!--></code>	Defines a comment

HTML Attributes

Attributes provide additional information to an HTML element.

HTML Tag Attributes

HTML tags can have attributes.

Attributes provide additional information to an HTML element.

Attributes always come in name/value pairs like this:

name="value".

Attributes are always specified in the start tag of an HTML element.

Use Lowercase Attributes

Attributes and attribute values are case-insensitive. However, the World Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation, and XHTML demands lowercase attributes/attribute values.

Always Quote Attribute Values

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed.

In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:

name= 'NBTE "Unesco" Nigeria'

Practice

We may demonstrate the Attribute Values by reformatting our project i.e. "Mypage" as follows.

```
<h1 Align = "Center" >This is the Main heading</h1>
```

```
<h2>This is the Sub heading</h2>
```

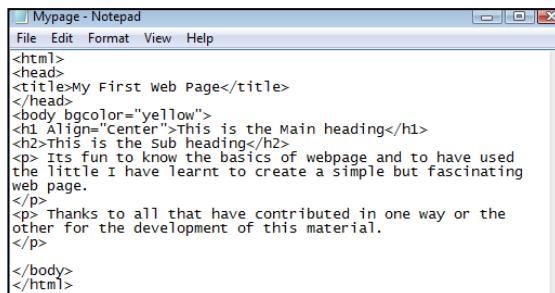
```
<p bgcolor="yellow"> Its fun to know the basics of webpage and to have used the little I have learnt to create a simple but fascinating web page.
```

```
</p>
```

```
<p> Thanks to all that have contributed in one way or the other for the development of this material.
```

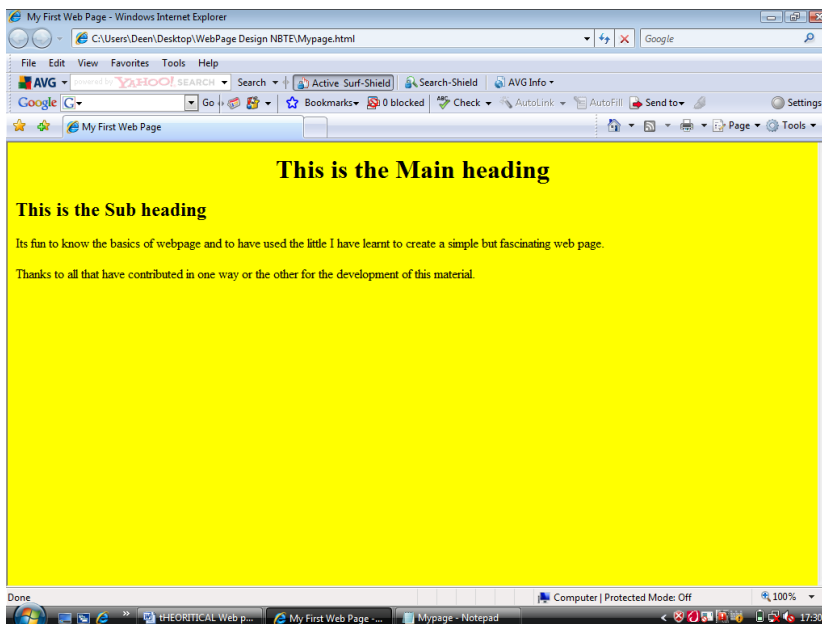
```
</p>
```

It should look as below in the editor



```
<html>
<head>
<title>My First web Page</title>
</head>
<body bgcolor="yellow">
<h1 Align="Center">This is the Main heading</h1>
<h2>This is the Sub heading</h2>
<p> Its fun to know the basics of webpage and to have used
the little I have learnt to create a simple but fascinating
web page.
</p>
<p> Thanks to all that have contributed in one way or the
other for the development of this material.
</p>
</body>
</html>
```

When run on the browser it appear as below



Text can be Aligned Left, Centre or Right. Any of these will take the form of the Centre alignment as presented above/

HTML Text Formatting

HTML defines a lot of elements for formatting output, like bold or italic text.

Below are a lot of examples that you can try out yourself:

Text Formatting Tags

Tag	Description
<u></u>	Defines bold text
<u><big></u>	Defines big text
<u></u>	Defines emphasized text
<u><i></u>	Defines italic text
<u><small></u>	Defines small text
<u></u>	Defines strong text
<u><sub></u>	Defines subscripted text
<u><sup></u>	Defines superscripted text
<u><ins></u>	Defines inserted text
<u></u>	Defines deleted text

"Computer Output" Tags

Tag	Description
<code><code></code>	Defines computer code text
<code><kbd></code>	Defines keyboard text
<code><samp></code>	Defines sample computer code
<code><tt></code>	Defines teletype text
<code><var></code>	Defines a variable
<code><pre></code>	Defines preformatted text

Citations, Quotations, and Definition Tags

Tag	Description
<code><abbr></code>	Defines an abbreviation
<code><acronym></code>	Defines an acronym
<code><address></code>	Defines an address element
<code><bdo></code>	Defines the text direction
<code><blockquote></code>	Defines a long quotation
<code><q></code>	Defines a short quotation
<code><cite></code>	Defines a citation
<code><dfn></code>	Defines a definition term

FONT Tag: Good or Bad?

Explicitly changing font types can also be accomplished through the capabilities of the `` tag. In most web browsers, the default font type for an HTML document is *Times New Roman*. The `` tag can be used with `face=""`, `size=""`, and `color="#hexValue"` attributes to change the font style of characters to be displayed by a web browser. *face* must equal the name of the font to be used, such as `font="arial"`. *size* must equal the size of the characters in points (pt), such as `size="2pt"`, or relative values. *color* must equal the hexadecimal color value of the characters, such as `color="#000000"` (black hexadecimal value). For an example, consider the following:

```
<font face="arial" size="4pt" color="#000080">4PT font size
rendered in Arial Type</font>
```

The above HTML code would produce the following results in a web browser:

4PT font size rendered in Arial Type

The following HTML code segment illustrates the size of possible point values when using the font tag:

```
<font size="1pt" face="arial" color="#FF2400">1 point arial type
text</font>
<font size="2pt" face="arial" color="#FF2400">2 point arial type
text</font>
<font size="3pt" face="arial" color="#FF2400">3 point arial type
text</font>
<font size="4pt" face="arial" color="#FF2400">4 point arial type
text</font>
<font size="5pt" face="arial" color="#FF2400">5 point arial type
text</font>
<font size="6pt" face="arial" color="#FF2400">6 point arial type
text</font>
<font size="7pt" face="arial" color="#FF2400">7 point arial type
text</font>
```

The above HTML code would produce the following results in a web browser:

1 point arial type text
 2 point arial type text
 3 point arial type text
 4 point arial type text
 5 point arial type text
 6 point arial type text
 7 point arial type
 text

The HTML Tag

With HTML code like this, you can specify both the size and the type of the browser output :

```
<p>
<font size="2" face="Verdana">
This is a paragraph.
</font>
</p>
<p>
```



```
<font size="3" face="Times">
This is another paragraph.
</font>
</p>
```

Font Attributes

Attribute	Example	Purpose
size="number"	size="2"	Defines the font size
size="+number"	size="+1"	Increases the font size
size="-number"	size="-1"	Decreases the font size
face="face-name"	face="Times"	Defines the font-name
color="color-value"	color="#eeff00"	Defines the font color
color="color-name"	color="red"	Defines the font color

NOTE: The tag has been described by many web developers to be the most "evil" of all HTML tags. This is mainly due to the fact that **CSS** has emerged as a better way to format and define content. Using the tag religiously will definitely be a concern for the future and should be avoided at all costs. Instead of using the expected to become obsolete tag, a web developer can make use of CSS or simply the *style* attribute of the <div> or tag. For example, consider:

```
<div style="font-family:arial">Here is some text in Arial
type.</div>
```

The above HTML code would produce the following results in a web browser:

Here is some text in Arial type.

It is also possible and very common to combine physical type styles with block level tags or physical type styles with non-block level tags. Care must be taken to ensure that all HTML syntax rules are followed.

How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did they do that?"

To find out, click the VIEW option in your browser's toolbar and select SOURCE or PAGE SOURCE. This will open a window that shows you the HTML code of the page.

HTML Character Entities

Some characters like the < character, have a special meaning in HTML, and therefore cannot be used in the text. To display a less than sign (<) in HTML, we have to use a character entity.

Character Entities

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in the HTML source.

A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).

To display a less than sign in an HTML document we must write:

< or **<**;

The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.

Note that the entities are case sensitive.

Non-breaking Space

The most common character entity in HTML is the non-breaking space.

Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the character entity.

The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space	 	
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
"	quotation mark	"	"
'	apostrophe	' (does not work in IE)	'

Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	¢	¢
£	pound	£	£
¥	yen	¥	¥
€	euro	€	€
§	section	§	§
©	copyright	©	©
®	registered trademark	®	®
×	multiplication	×	×
÷	division	÷	÷

WEEK Four

HTML Links

HTML uses a hyperlink to link to another document on the Web.

A hyperlink is text, an image, or any other object in an HTML document that can be clicked in order to gain access to another web document. Normally, links are distinguished as being external or internal to the current page. An external web site link provides access to another web site that is not part of the current web site, such as going to www.yahoo.com from our current website *mypage.html*. An internal web site link provides access to another web page which is apart of the original web site, such as going to the next article in this guide from this web page. Finally, an internal document section link points to a region within a web page document.

The Anchor Tag and the Href Attribute

HTML uses the <a> (anchor) tag to create a link to another document.

An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc.

The syntax of creating an anchor:

```
<a href="url">Text to be displayed</a>
```

The <a> tag is used to create an anchor to link from, the href attribute is used to address the document to link to, and the words between the open and close of the anchor tag will be displayed as a hyperlink.

For example, the anchor below defines a link to Yahoo page:

```
<a href="http://www.yahoo.com/">Visit Yahoo Site!</a>
```

The line above will look like this in a browser:

[Visit Yahoo Site!](http://www.yahoo.com/)

The Target Attribute

With the target attribute, you can define **where** the linked document will be opened.

The line below will open the document in a new browser window:

```
<a href="http://www.yahoo.com/"
target="_blank">Visit Yahoo site!</a>
```

Getting Out and Around

Hyperlinks are easily noticed in a web page document because by default, they are underlined and painted blue (unless otherwise specified) and a mouse pointer will change to a hand with the index finger pointing to the link indicating that the web user may interact with it. Also, by default, image hyperlinks are given a distinct border color (unless otherwise specified). HTML provides ways to manipulate the color of links, visited links, and active links by using the following attributes of the `<body>` tag:

```
<body link="#hexColor" vlink="#hexColor" alink="#hexColor">
```

link : link color
 vlink : visited link color
 alink : active link color (upon click)

#hexColor must be a valid hexadecimal value representing the color to be used for each attribute as usual. The border surrounding an image hyperlink can be turned off by using the `border="0"` attribute of the image tag such as:

```

```

The following illustrates how to use the *anchor* tag to link to an external web site:

```
<a href="http://www.programmersheaven.com"
target="_blank">Programmer's Heaven</a>
```

The above would produce the following:

Programmer's Heaven

The following illustrates how to link to a web page that is apart of a web site:

```
<a href="/tutorials/cpp.htm">[warebiz] :: C++ Ripped Apart</a>
```

The above would produce the following:

[warebiz] :: C++ Ripped Apart

The following illustrates how to produce internal web document links:

```
<a href="#linkName"> </a> --> use as the physical link
<a name="linkName"> </a> --> use as the anchor
```

When creating a linkable image, simply place the *img src* tag inside the *anchor* tag. Use the following form:

```
<a href="linkURL"></a>
```

Using the above form, a default link border color will surround the image. The border surrounding an image hyperlink can be turned off by using the `border="0"` attribute of the image tag such as:

```
<a href="linkURL"></a>
```

HTML Lists

HTML supports ordered, unordered and definition lists.

HTML Lists - Looking Sharp and Organized

There may be times when web designers need to organize information or images in a neat format resembling a list. This allows the designer to group related pieces of items together so they can be clearly seen increasing readability and importance. HTML has tags for this purpose and supports the following three block-level list types: unordered (ul), ordered (ol), and definition lists (dl).

Unordered Lists

Unnumbered or unordered lists are usually displayed with bullets, which depict each new line of information to be displayed in the list structure. Unordered lists should be used when needing to display a related group of data without stressing the order in which the data is presented. There are four steps needed to create an unordered list:

1. Start with the opening unnumbered list tag ``.
2. Enter the list item tag `` followed by the item or content to

be listed.

3. Continue to list items using the list item tag.

4. End the unnumbered list by using the closing tag ``.

NOTE: The `type=""` attribute can be used in the opening `` tag to specify a *square*, *circle*, or *disc* shaped bullet. If no bullet is specified, the default solid disk shape is used. Also note that the closing `` tag is optional when entering `` items.

The following code illustrates an example of an unordered list:

```
<html>
<head><title>Unordered List Example</title></head>
<body>

<ul>
<li>List Item 1
<li>List Item 2
<li>List Item 3
<li>List Item 4
</ul>

</body>
</html>
```

The above HTML code would produce the following results in a web browser:

- List Item 1
- List Item 2
- List Item 3
- List Item 4

Numbered Lists

Numbered lists are coded identical to an unnumbered list except for the opening and closing tag, which are: `` ``. Numbered lists should be used when needing to display data when order is important.

NOTE: With numbered lists, the `type=""` attribute can be used in the opening `` tag to specify different types of numbering styles. If no value is specified, the default *Arabic* numbering style is used. Also note that the closing `` tag is optional when entering list `` items. The following is a listing of the possible numbering styles for the `type=""` attribute:

<u>VALUE</u>	<u>MEANING</u>
1	Arabic (1, 2, 3, ...) [default]
A	Alphabetic uppercase
a	Alphabetic lowercase
I	Roman numeral uppercase

The following code illustrates an example of a numbered list:

Numbered/Ordered list

```
<html>
<head><title>Numbered List Example</title></head>
<body>

<ol type="A">
<li>List Item 1
<li>List Item 2
<li>List Item 3
<li>List Item 4
</ol>

</body>
</html>
```

The above code would produce the following results:

- A. List Item 1
- B. List Item 2
- C. List Item 3
- D. List Item 4

Definition Lists

Definition lists can be used for two purposes. If needing to list terms with definitions, designers can use alternating definition tags <dt> and definition data tags <dd>. Note that using closing tags when using the <dt> and <dd> tags is optional. Designers can also use a definition list when using a custom bullet image instead of the standard bullet types of numbered and unordered lists. If using a custom bullet, only <dd> opening and closing tags should be used to list the bullet images and items. To begin and end a definition list, use the <dl></dl> start and closing tags.

The following examples illustrate the two types of definition lists:

```
<html>
<head><title>Definition List Example</title></head>
```



```

<body>

<dl>
<dt><b><i>Word 1</i></b></dt><dd>This corresponds to the
meaning of word 1.</dd><br><br>
<dt><b><i>Word 2</i></b></dt><dd>This corresponds to the
meaning of word 2.</dd><br><br>
<dt><b><i>Word 3</i></b></dt><dd>This corresponds to the
meaning of word 3.</dd><br><br>
<dt><b><i>Word 4</i></b></dt><dd>This corresponds to the
meaning of word 4.</dd><br><br>
</dl>

</body>
</html>

```

The above HTML code would produce the following results in a web browser:

Word 1

This corresponds to the meaning of word 1.

Word 2

This corresponds to the meaning of word 2.

Word 3

This corresponds to the meaning of word 3.

Word 4

This corresponds to the meaning of word 4.

Here is an example with custom bullets:

```

<html>
<head><title>Definition List Example</title></head>
<body>

<dl>
<dd> List item 1</dd><br>
<dd> List item 2</dd><br>
<dd> List item 3</dd><br>
<dd> List item 4</dd>
</dl>

</body>
</html>

```

The above HTML code would produce the following results in a web browser assuming that the image is named and is located correctly:

- List Item 1
- List Item 2
- List Item 3

☐ List Item 4

In the next section, we will cover how visitors move from one web page to the next on the WWW

List Tags

Tag	Description
<u></u>	Defines an ordered list
<u></u>	Defines an unordered list
<u></u>	Defines a list item
<u><dl></u>	Defines a definition list
<u><dt></u>	Defines a definition term
<u><dd></u>	Defines a definition description

WEEK Five

HTML Images

If a web page needs a visual appeal, an important step in the design should be the creation of many custom images. Most visitors prefer looking at informative pictures and images rather than reading text, and visitors are also drawn to looking at images before text. Web designers should add relevant images to their web pages to add visual life and substance. Including images should also coincide with the overall layout of the web site.

In HTML, images are defined with the `` tag.

The `` tag is empty, which means that it contains attributes only and it has no closing tag.

To display an image on a page, you need to use the `src` attribute. `Src` stands for "source". The value of the `src` attribute is the URL of the image you want to display on your page.

Most web browsers display images in GIF and JPEG format, but with new browser capabilities, more file formats are capable of being displayed. To include an image in a web page document, use the image search tag:

```

```

When using the image search tag, a web designer will need to specify the name and type of image to be displayed and perhaps the full path (URL) of the image (if not placed in the current directory). If you do not specify the full path, the image must be in the current web directory.

For example, if an image named *logo* of format *gif* is in the current web directory, the following image search specification would be sufficient:

```

```

Attributes

The image search tag comes equip with `width=""` and `height=""` attributes to specify the width and height of the image in pixels. Specifying width and height attributes for an image has been said to allow for faster loading times. It simply lets the browser allocate space on the web page for the image. For an example, if the previous *logo* image had dimensions of 200 by 100, the following image search specification would be sufficient:

```

```

Because some browsers cannot display certain image formats and some visitors choose to turn off image loading, web designers should also use the `alt=""` attribute to inform the visitor of which type of graphic would have been displayed. The text is displayed instead of the image only if the browser refuses to load the image either because of error or an image "block". The `alt=""` attribute will also provide a balloon tip containing the text specified between the quotes when a mouse is hovering over the displayed image. For example, consider the following image search specification with a newly added *alt* attribute to our previous logo image:

```

```

Once you master adding one image to a web page, the same routine is applied for each new image to be added.

HTML Backgrounds

A good background can make a Web site look really great.

Backgrounds

The `<body>` tag has two attributes where you can specify backgrounds. The background can be a color or an image.

Bgcolor

The `bgcolor` attribute specifies a background-color for an HTML page. The value of this attribute can be a hexadecimal number, an RGB value, or a color name:

```
<body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)">
<body bgcolor="black">
```

The lines above all set the background-color to black.

Background

The `background` attribute specifies a background-image for an HTML page. The value of this attribute is the URL of the image you want to use. If the image is smaller than the browser window, the image will repeat itself until it fills the entire browser window.

```
<body background="unesco.jpg">
<body background="C:\Users\Deen\Desktop\WebPage Design
```

```
NBTE\unesco.jpg">
```

The URL can be relative (as in the first line above) or absolute (as in the second line above).

Note: If you want to use a background image, you should keep in mind:

- Will the background image increase the loading time too much?
- Will the background image look good with other images on the page?
- Will the background image look good with the text colors on the page?
- Will the background image look good when it is repeated on the page?
- Will the background image take away the focus from the text?

Basic Notes - Useful Tips

The bgcolor, background, and the text attributes in the <body> tag are deprecated in the latest versions of HTML (HTML 4 and XHTML). The World Wide Web Consortium (W3C) has removed these attributes from its recommendations.

Style sheets (CSS) should be used instead (to define the layout and display properties of HTML elements).

HTML Colors

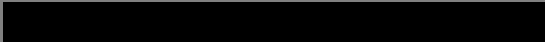






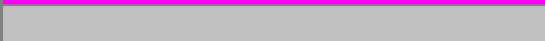

Colors are displayed combining RED, GREEN, and BLUE light sources.

Color Values

HTML colors can be defined as a hexadecimal notation for the combination of Red, Green, and Blue color values (RGB).

The lowest value that can be given to one light source is 0 (hex #00) and the highest value is 255 (hex #FF).

The table below shows the result of combining Red, Green, and Blue light sources:.

Color	Color HEX	Color RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

W3C Standard Color Names

W3C has listed 16 color names that will validate with an HTML validator.

The color names are: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow.

Cross-browser Color Names

A collection of nearly 150 color names are supported by all major browsers.

Cross-browser Color Values

Some years ago, when most computers only supported 256 different colors, a list of 216 Web Safe Colors was suggested as a Web standard. The reason for this was that the Microsoft and Mac operating system used 40 different "reserved" fixed system colors (about 20 each).

We are not sure how important this is now, since most computers today have the ability to display millions of different colors, but the choice is left to you.

The 216 cross-browser color palette was created to ensure that all computers would display the colors correctly when running a 256 color palette:

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

HTML Color Names

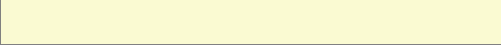
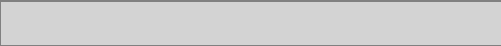
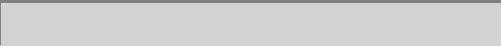
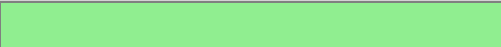
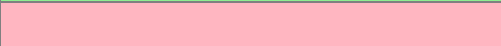


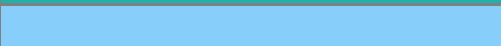






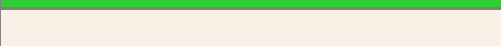









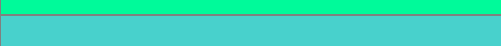




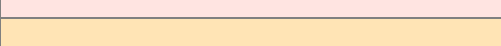


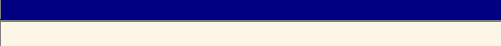






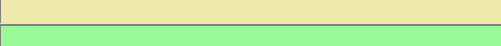
HTML Color Names

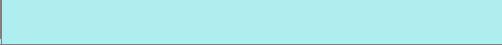

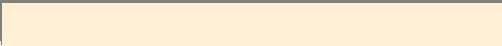


































The table below provides a list of the color names that are supported by all major browsers.

Note: If you want your pages to validate with an HTML or a CSS validator, W3C has listed 16 color names that you can use: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. If you want to use other colors, you must specify their RGB or HEX value.

Color Name	Color HEX	Color
AliceBlue	#F0F8FF	
AntiqueWhite	#FAEBD7	
Aqua	#00FFFF	
Aquamarine	#7FFFD4	
Azure	#F0FFFF	
Beige	#F5F5DC	
Bisque	#FFE4C4	
Black	#000000	
BlanchedAlmond	#FFEBCD	
Blue	#0000FF	
BlueViolet	#8A2BE2	
Brown	#A52A2A	
BurlyWood	#DEB887	
CadetBlue	#5F9EA0	
Chartreuse	#7FFF00	
Chocolate	#D2691E	
Coral	#FF7F50	
CornflowerBlue	#6495ED	
Cornsilk	#FFF8DC	
Crimson	#DC143C	
Cyan	#00FFFF	
DarkBlue	#00008B	
DarkCyan	#008B8B	
DarkGoldenRod	#B8860B	
DarkGray	#A9A9A9	
DarkGrey	#A9A9A9	
DarkGreen	#006400	
DarkKhaki	#BDB76B	
DarkMagenta	#8B008B	
DarkOliveGreen	#556B2F	

Darkorange	#FF8C00	
DarkOrchid	#9932CC	
DarkRed	#8B0000	
DarkSalmon	#E9967A	
DarkSeaGreen	#8FBC8F	
DarkSlateBlue	#483D8B	
DarkSlateGray	#2F4F4F	
DarkSlateGrey	#2F4F4F	
DarkTurquoise	#00CED1	
DarkViolet	#9400D3	
DeepPink	#FF1493	
DeepSkyBlue	#00BFFF	
DimGray	#696969	
DimGrey	#696969	
DodgerBlue	#1E90FF	
FireBrick	#B22222	
FloralWhite	#FFFAF0	
ForestGreen	#228B22	
Fuchsia	#FF00FF	
Gainsboro	#DCDCDC	
GhostWhite	#F8F8FF	
Gold	#FFD700	
GoldenRod	#DAA520	
Gray	#808080	
Grey	#808080	
Green	#008000	
GreenYellow	#ADFF2F	
HoneyDew	#F0FFF0	
HotPink	#FF69B4	
IndianRed	#CD5C5C	
Indigo	#4B0082	
Ivory	#FFFFFF0	
Khaki	#F0E68C	
Lavender	#E6E6FA	
LavenderBlush	#FFF0F5	
LawnGreen	#7CFC00	
LemonChiffon	#FFFACD	
LightBlue	#ADD8E6	
LightCoral	#F08080	
LightCyan	#E0FFFF	

LightGoldenRodYellow	#FAFAD2	
LightGray	#D3D3D3	
LightGrey	#D3D3D3	
LightGreen	#90EE90	
LightPink	#FFB6C1	
LightSalmon	#FFA07A	
LightSeaGreen	#20B2AA	
LightSkyBlue	#87CEFA	
LightSlateGray	#778899	
LightSlateGrey	#778899	
LightSteelBlue	#B0C4DE	
LightYellow	#FFFFE0	
Lime	#00FF00	
LimeGreen	#32CD32	
Linen	#FAF0E6	
Magenta	#FF00FF	
Maroon	#800000	
MediumAquaMarine	#66CDAA	
MediumBlue	#0000CD	
MediumOrchid	#BA55D3	
MediumPurple	#9370D8	
MediumSeaGreen	#3CB371	
MediumSlateBlue	#7B68EE	
MediumSpringGreen	#00FA9A	
MediumTurquoise	#48D1CC	
MediumVioletRed	#C71585	
MidnightBlue	#191970	
MintCream	#F5FFFA	
MistyRose	#FFE4E1	
Moccasin	#FFE4B5	
NavajoWhite	#FFDEAD	
Navy	#000080	
OldLace	#FDF5E6	
Olive	#808000	
OliveDrab	#6B8E23	
Orange	#FFA500	
OrangeRed	#FF4500	
Orchid	#DA70D6	
PaleGoldenRod	#EEE8AA	
PaleGreen	#98FB98	

PaleTurquoise	#AFEEEE	
PaleVioletRed	#D87093	
PapayaWhip	#FFEFD5	
PeachPuff	#FFDAB9	
Peru	#CD853F	
Pink	#FFC0CB	
Plum	#DDA0DD	
PowderBlue	#B0E0E6	
Purple	#800080	
Red	#FF0000	
RosyBrown	#BC8F8F	
RoyalBlue	#4169E1	
SaddleBrown	#8B4513	
Salmon	#FA8072	
SandyBrown	#F4A460	
SeaGreen	#2E8B57	
SeaShell	#FFF5EE	
Sienna	#A0522D	
Silver	#C0C0C0	
SkyBlue	#87CEEB	
SlateBlue	#6A5ACD	
SlateGray	#708090	
SlateGrey	#708090	
Snow	#FFFAFA	
SpringGreen	#00FF7F	
SteelBlue	#4682B4	
Tan	#D2B48C	
Teal	#008080	
Thistle	#D8BFD8	
Tomato	#FF6347	
Turquoise	#40E0D0	
Violet	#EE82EE	
Wheat	#F5DEB3	
White	#FFFFFF	
WhiteSmoke	#F5F5F5	
Yellow	#FFFF00	
YellowGreen	#9ACD32	

WEEK Six

HTML Forms and Input

HTML Forms are used to select different kinds of user input.

An HTML form allows a visitor to input information and possibly send that information to a server on the Internet for processing. Form elements such as text boxes, radio buttons, checkboxes, and text fields provide a graphical interface so visitors can specify data very easily. Creating the HTML code to devise a web page form is not difficult, but creating the scripting code that receives and processes the form data can be complex. Constructing an HTML form for a web page is a two-part process. The first part being the actual creation of a form in an HTML document, which is covered in this week's activities. The second part is the creation of a CGI (common gateway interface) program that will receive the form data and implement the action specified within the program located on a web server. The creation of a CGI program requires knowledge of a high-level programming language such as Perl, C, C++, ASP, PHP, etc.

A form is generated in HTML using the `<form>` element. All other form elements such as `<input>`, `<select>`, and `<textarea>` must be placed within the opening and closing `<form>` tags.

Form Element

Form elements are elements that allow the user to enter information (like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.) in a form.

A form is defined with the `<form>` tag.

```
<form>  
  <input>  
  <input>  
</form>
```

The `<form>` element has the following standard attributes:

<code>action=""</code>	- specifies the URL of the CGI program located on a web server which will process all active form data elements.
------------------------	--

method="" - specifies the way in which the form data will be transferred to the HTTP server. Method options are "get" and "post" with "get" being the default method type.

enctype="" - specifies the way in which the form data will be encoded before it is transferred to the HTTP server.

Input

The most used form tag is the <input> tag. The type of input is specified with the type attribute.

The **input** element is the basis for various button types and text types. <input> attributes are as follows:

*** *Attributes* ***

type="text"	- specifies the data entry field property to use as input.
type="button"	
type="checkbox"	
type="file"	
type="hidden"	
type="image"	
type="password"	
type="radio"	
type="reset"	
type="submit"	
name=""	- gives the input type a name for CGI processing purposes. Required for all form types except submit, reset, and button.
value=""	- either gives a form field an initial value or gives a label to a button. Required for radio and checkbox form types.
align="top"	- provides an alignment for an image form type.
align="middle"	
align="left"	
align="right"	
checked	- specifies the initial state of a radio or checkbox form type to be selected.
maxlength="#"	- specifies the maximum length of form textfield characters.
size="#"	- specifies the visual number of textfield characters in a textfield.

`src="imageURL"` - specifies the URL of the image for an input image type.

The most commonly used input types are explained below.

Text Fields

Text fields are used when you want the user to type letters, numbers, etc. in a form.

```
<form>
First name:
<input type="text" name="firstname">
<br>
Last name:
<input type="text" name="lastname">
</form>
```

How it looks in a browser:

First name:

Last name:

Note that the form itself is not visible. Also note that in most browsers, the width of the text field is 20 characters by default.

input Examples

The `type=""` attribute is used to specify which input form type to use. Study the examples that follow to see which attributes may be used for each type.

The following HTML code segment is needed to produce a text box:

```
<form>
Enter name: <input type="text" size="25" name="name">
</form>
```

The above HTML code segment would produce the following results in a web browser:

Enter name:

Radio Buttons

Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
<input type="radio" name="sex" value="male"> Male
<br>
```

```
<input type="radio" name="sex" value="female"> Female
</form>
```

How it looks in a browser:

- ☐ Male
☐ Female

Note that only one option can be chosen.

The following HTML code segment is needed to produce a set of radio buttons:

```
<form>
<input type="radio" name="size" value="small">Small
<input type="radio" name="size" value="medium">Medium
<input type="radio" name="size" value="large">Large
</form>
```

The above HTML code segment would produce the following results in a web browser:

- ☐ Small ☐ Medium ☐ Large

Checkboxes

Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
I have a bike:
<input type="checkbox" name="vehicle" value="Bike">
<br>
I have a car:
<input type="checkbox" name="vehicle" value="Car">
<br>
I have an airplane:
<input type="checkbox" name="vehicle" value="Airplane">
</form>
```

How it looks in a browser:

I have a bike: ☐

I have a car: ☐

I have an airplane: ☐

The following HTML code segment is needed to produce a set of checkboxes:

```
<form>
<input type="checkbox" name="topping"
```

```
value="pepperoni">Pepperoni
<input type="checkbox" name="topping"
value="sausage">Sausage
<input type="checkbox" name="topping"
value="mushrooms">Mushrooms
<input type="checkbox" name="topping" value="black
olives">Black Olives
</form>
```

The above HTML code segment would produce the following results in a web browser:

☐ Pepperoni ☐ Sausage ☐ Mushrooms ☐ Black Olives

The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to another file. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_action.asp"
method="get">
Username:
<input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

How it looks in a browser:

Username:

If you type some characters in the text field above, and click the "Submit" button, you will send your input to a page called "html_form_action.asp". That page will show you the received input.

The following HTML code segment is needed to produce a reset and submit button:

```
<form>
<input type="reset" value="Reset">
<input type="submit" value="Submit">
</form>
```

The above HTML code segment would produce the following results in a web browser:

TAG NAME: select

PURPOSE: This element creates a pull-down menu. The <option> element is included inside the <select> element in order to define menu choices. The selected attribute in the <option> element specifies the default menu option to be selected.

***** Attributes *****

name=""	- gives the <select> element a name for CGI processing purposes. (required)
size="#"	- specifies the number of visible rows.
multiple	- specifies whether multiple <select> entries may be selected simultaneously.

TAG NAME: option

PURPOSE: Used to specify the menu options of a <select> form type.

***** Attributes *****

value=""	- specifies the value to be associated with the <select> form type name if the menu option is selected.
selected	- specifies that the menu option is selected or shown by default.

The following HTML code segment is needed to produce a pull-down menu:

```
<form>
What is your favorite color
<select name="favorite color">
<option selected value="red">Red
<option value="orange">Orange
<option value="yellow">Yellow
<option value="green">Green
<option value="blue">Blue
<option value="indigo">Indigo
<option value="violet">Violet
</select>
</form>
```

The above HTML code segment would produce the following results in a web browser:

What is your favorite color?

TAG NAME: `textarea`

PURPOSE: This element specifies a multi-line text area. The columns and rows attributes in the `<textarea>` specify the default dimensions of the field.

***** Attributes *****

<code>name=""</code>	- gives the <code><textarea></code> element a name for CGI processing purposes.
<code>rows="#"</code>	- specifies the number of visible <code><textarea></code> row dimensions.
<code>cols="#"</code>	- specifies the number of visible <code><textarea></code> column dimensions.

The following HTML code segment is needed to produce a multi-line text field:

```
<form>
Enter comments:
<textarea name="words" cols="45" rows="8">
Start...
</textarea>
</form>
```

The above HTML code segment would produce the following results in a web browser:

Enter comments:


Overview

When the "submit" button is clicked, the form data is sent to a CGI program (specified in the action attribute of `<form>` tag which should be running on a web server. The CGI program receives the form data and processes the data. If the CGI program is not created, the HTML form basically serves only a visual appeal with no interaction or processing capabilities. It will only look like an online form is on a web page until a CGI program is implemented to process the active form data.

Form Tags

Tag	Description
<code><form></code>	Defines a form for user input
<code><input></code>	Defines an input field
<code><textarea></code>	Defines a text-area (a multi-line text input control)
<code><label></code>	Defines a label to a control
<code><fieldset></code>	Defines a fieldset
<code><legend></code>	Defines a caption for a fieldset
<code><select></code>	Defines a selectable list (a drop-down box)
<code><optgroup></code>	Defines an option group
<code><option></code>	Defines an option in the drop-down box
<code><button></code>	Defines a push button

WEEK Seven

HTML Frames

With frames, you can display more than one Web page in the same browser window.

Frames

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- The web developer must keep track of more HTML documents
 - It is difficult to print the entire page
-

The Frameset Tag

- The <frameset> tag defines how to divide the window into frames
 - Each frameset defines a set of rows **or** columns
 - The values of the rows/columns indicate the amount of screen area each row/column will occupy
-

The Frame Tag

- The <frame> tag defines what HTML document to put into each frame

In the example below we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document "frame_a.html" is put into the first column, and the HTML document "frame_b.html" is put into the second column:

```
<frameset cols="25%,75%">
  <frame src="frame_a.html">
  <frame src="frame_b.html">
</frameset>
```

Note: The frameset column size value can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space (cols="25%,*").

Basic Notes - Useful Tips

If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add `noresize="noresize"` to the `<frame>` tag.

Add the `<noframes>` tag for browsers that do not support frames.

Important: You cannot use the `<body></body>` tags together with the `<frameset></frameset>` tags! However, if you add a `<noframes>` tag containing some text for browsers that do not support frames, you will have to enclose the text in `<body></body>` tags!

The Web according to the dictates of the "Main.html"

Frame Tags

Tag	Description
<code><frameset></code>	Defines a set of frames
<code><frame></code>	Defines a sub window (a frame)
<code><noframes></code>	Defines a noframe section for browsers that do not handle frames
<code><iframe></code>	Defines an inline sub window (frame)

HTML Tables

With HTML you can create tables.

In most applications, a table is commonly used to present tabular information such as schedules, statistics, calendars, etc. In the world of HTML, a table is also used for these purposes, but it is more commonly used for organizing the content of complex web pages. Many web designers prefer to create an overall web site layout through the use of a single table. A table lets web designers make use of rows and columns so they can specify precisely where text, images, and objects are to be placed within their documents. This is what makes the `<table>` element one of the most powerful and versatile of all HTML tags. The following gives an overview of the elements and attributes that may be used with the `<table>` tag:

Table Elements and Corresponding Attributes

<table> </table>

PURPOSE: Used to define a table.

*** *Attributes* ***

<code>align=""</code> [left, center, right]	- specifies the horizontal alignment of the table.
<code>border="#"</code>	- specifies the thickness of the table border in pixels represented by #. 0 = no border; 1 = default; the higher the number, the thicker the border.
<code>cellpadding="#"</code>	- specifies the thickness of the area inside the contents of a cell in pixels represented by #; the higher the number, the thicker the area.
<code>cellspacing="#"</code>	- specifies the thickness of the area outside the contents of a cell in pixels represented by #; the higher the number, the thicker the area.
<code>width="#"</code>	- specifies the width of the table in pixels represented by #. Use pixel values such as <code>width="500"</code> or use percentage values such as <code>width="100%"</code> . Percentages are based on the screen resolution of the visitor's display.
<code>bgcolor="#hexValue"</code>	- specifies the color of the background of the table; must use hexadecimal HTML color values as usual.
<code>background="imageURL"</code>	- specifies an image to be used as the background of the table; image must be present in current directory as usual unless full path of image is provided.

The following tags must be placed within the beginning and closing `<table> </table>` tag in order for them to be associated with the table.

Tables are defined with the `<table>` tag. A table is divided into rows (with the `<tr>` tag), and each row is divided into data cells (with the `<td>` tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

```
<table border="1">
<tr>
```

```

<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>

```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Tables and the Border Attribute

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show.

To display a table with borders, you will have to use the border attribute:

```

<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>

```

<caption></caption>

PURPOSE: Defines a title to be used for the table positioned above the first row, by default. Use the attribute align="bottom" to place the caption below the last row.

<th></th>

PURPOSE: Defines a table header cell. By default, the text is centered and is of bold type.

*** Attributes ***

align="" - [left, center, right]	- specifies the horizontal alignment of the contents of a cell.
valign="" - [top, middle, bottom]	- specifies the vertical alignment of the contents of a cell.
colspan="#"	- specifies the number of columns a cell will expand represented by #.

<code>rowspan="#"</code>	- specifies the number of rows a cell will expand represented by #.
<code>nowrap</code>	- eliminates word wrapping in a cell.
<code>width="#"</code>	- specifies the width of the cell. (pixel or percentage values) represented by #.
<code>height="#"</code>	- specifies the height of the cell. (pixel or percentage values) represented by #.
<code>bgcolor="#hexValue"</code>	- specifies the color of the background of the cell; must use hexadecimal HTML color values as usual.
<code>background="imageURL"</code>	- specifies an image to be used as the background of the cell.

Headings in a Table

Headings in a table are defined with the `<th>` tag.

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

`<tr></tr>`

PURPOSE: Defines a table row within the table.

*** *Attributes* ***

<code>align=""</code> - [left, center, right]	- specifies the horizontal alignment of the contents of a cell.
<code>valign=""</code> - [top, middle, bottom]	- specifies the vertical alignment of the contents of a cell.
<code>bgcolor="#hexValue"</code>	- specifies the color of the background of the cell; must use hexadecimal HTML color values as usual.
<code>background="imageUrl"</code>	- specifies an image to be used as the background of the cell.

<td></td>

PURPOSE: Defines a table data cell within the table row. NOTE: A data cell must be placed inside a table row.

*** *Attributes* ***

<code>align=""</code> - [left, center, right]	- specifies the horizontal alignment of the contents of a cell.
<code>valign=""</code> - [top, middle, bottom]	- specifies the vertical alignment of the contents of a cell.
<code>colspan="#"</code>	- specifies the number of columns a cell will expand represented by #.
<code>rowspan="#"</code>	- specifies the number of rows a cell will expand represented by #.
<code>nowrap</code>	- eliminates word wrapping in a cell.
<code>width="#"</code>	- specifies the width of the cell. (pixel or percentage values) represented by #.
<code>height="#"</code>	- specifies the height of the cell. (pixel or percentage values) represented by #.
<code>bgcolor="#hexValue"</code>	- specifies the color of the background of the cell; must use hexadecimal HTML color values as usual.
<code>background="imageUrl"</code>	- specifies an image to be used as the background of the cell.

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
```

```

<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>

```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing (NB! Mozilla Firefox displays the border).

To avoid this, add a non-breaking space () to empty data cells, to make the borders visible:

```

<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>

```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

The followings are examples illustrating how tables can be used to provide structure to web documents. Tabular structure, complex web page structure, border tricks, and complex web page structure with a navigational menu are emphasized.

Example 1: (simple table display for tabular information)

```

<html>
<head><title></title></head>
<body>

<table border="1" width="300">
<caption><font size="2pt" face="arial">Example Table</font></caption>
<tr>
<th bgcolor="#114E86"><font size="2pt" face="arial"
color="#FFFFFF">Heading 1</font></th>
<th bgcolor="#114E86"><font size="2pt" face="arial"

```

```

color="#FFFFFF">Heading 2</font></th>
</tr>
<tr>
<td align="left"><font size="2pt" face="arial">Text
A</font></td>
<td align="right"><font size="2pt" face="arial">Text
B</font></td>
</tr>
<tr>
<td align="center"><font size="2pt" face="arial">Text
C</font></td>
<td align="center"><font size="2pt" face="arial">Text
D</font></td>
</tr>
<tr>
<td align="left"><font size="2pt" face="arial">Text
E</font></td>
<td align="right"><font size="2pt" face="arial">Text
F</font></td>
</tr>
<td align="center" colspan="2"><font size="2pt" face="arial">Text
G</font></td>
</tr>
</table>

</body>
</html>

```

The above HTML code would produce the following results in a web browser:

Example Table	
Heading 1	Heading 2
Text A	Text B
Text C	Text D
Text E	Text F
Text G	

Example 2: (complex web page for entire web page layout)

```

<table cellpadding="5" cellspacing="0" border="0" width="100%"
height="300">
<tr>
<td width="25%" bgcolor="#114E86" align="left" valign="top">
<font size="2pt" face="arial" color="#FFFFFF">This text
corresponds to the left content panel of the overall layout of the

```

```

web page document.</font>
</td>
<td width="50%" bgcolor="#FFFFFF" align="center" height="300"
valign="middle">
<font size="2pt" face="arial">This text corresponds to the center
content panel of the overall layout of the web page
document.</font>
</td>
<td width="25%" bgcolor="#E8E8E8" align="right"
valign="bottom">
<font size="2pt" face="arial" color="#114E86">This text
corresponds to the right content panel of the overall layout of the
web page document.</font>
</td>
</tr>
</table>

```

The above HTML code segment would produce the following results in a web browser:

This text
corresponds to
the left content
panel of the
overall layout of
the web page
document.

This text corresponds to the
center content panel of the
overall layout of the web page
document.

This text
corresponds to
the right
content panel of
the overall
layout of the
web page
document.

Example 3: (border tricks)

```

<table cellpadding="1" cellspacing="0" width="250" border="0"
bgcolor="#114E86"><tr><td>
<table cellpadding="2" cellspacing="0" width="100%" border="0"
bgcolor="#FFFFFF"><tr><td>
<table cellpadding="1" cellspacing="0" width="100%" border="0"
bgcolor="#C0C0C0"><tr><td>
<table cellpadding="2" cellspacing="0" width="100%" border="0"
bgcolor="#FFFFFF"><tr><td>
<table cellpadding="1" cellspacing="0" width="100%" border="0"
bgcolor="#114E86"><tr><td>
<table cellpadding="2" cellspacing="0" width="100%" border="0"
bgcolor="#C0C0C0"><tr><td height="100" align="center"
valign="middle">
<b>Text Here</b>
</td></tr></table>
</td></tr></table>
</td></tr></table>

```

```

</td></tr></table>
</td></tr></table>
</td></tr></table>
</td></tr></table>

```

The above HTML code segment would produce the following results in a web browser:



Example 4: (more border tricks)

```

<table cellpadding="0" cellspacing="0" width="450"
align="center">
<tr>
<td width="220">

<table cellpadding="1" cellspacing="0" width="100%" border="0"
bgcolor="#C0C0C0"><tr><td>
<table cellpadding="1" cellspacing="0" width="100%" border="0"
bgcolor="#E8E8E8"><tr><td align="center">
<font size="2pt" face="arial">Heading 1</font>
</td></tr></table>
</td></tr></table>

</td>
<td width="10"></td>
<td width="220">

<table cellpadding="1" cellspacing="0" width="100%" border="0"
bgcolor="#C0C0C0"><tr><td>
<table cellpadding="1" cellspacing="0" width="100%" border="0"
bgcolor="#E8E8E8"><tr><td align="center">
<font size="2pt" face="arial">Heading 2</font>
</td></tr></table>
</td></tr></table>

</td>
</tr>
<tr>
<td width="220" valign="top">

<table cellpadding="8" cellspacing="0" width="100%"

```

```
border="0"><tr><td>
<font size="2pt" face="arial">Put some text in this
location...</font>
</td></tr></table>

</td>
<td width="10"></td>
<td width="220">

<table cellpadding="8" cellspacing="0" width="100%"
border="0"><tr><td>
<font size="2pt" face="arial">Put more text in this
location...</font>
</td></tr></table>

</td>
</tr>
</table>
```

The above HTML code segment would produce the following results in a web browser:

Heading 1	Heading 2
Put some text in this location...	Put more text in this location...

Example 5: (complex web page using a navigational left pane vertical bar)

```
<center>
<table cellpadding="1" cellspacing="0" border="0"
bgcolor="#000000" width="100%"><tr><td>
<table cellpadding="0" cellspacing="0" border="0" width="100%">
<tr>

<!-- begin left pane of content area -->
<td width="20%" align="center" valign="top" bgcolor="#C0C0C0"
height="350">
<br><br>
<table cellpadding="0" cellspacing="0" border="0" width="100%">
<tr><td align="center"><font size="2pt" face="arial"><b>.:</b>
<a href="/">Nav Link 1</a></td></tr>
<tr><td align="center"><font size="2pt" face="arial"><b>.:</b>
<a href="/">Nav Link 2</a></td></tr>
```

```

<tr><td align="center"><font size="2pt" face="arial"><b>.:</b>
<a href="/">Nav Link 3</a></td></tr>
<tr><td align="center"><font size="2pt" face="arial"><b>.:</b>
<a href="/">Nav Link 4</a></td></tr>
<tr><td align="center"><font size="2pt" face="arial"><b>.:</b>
<a href="/">Nav Link 5</a></td></tr>
</table>
</td>
<!-- end left pane of content area -->

<!-- begin right pane of content area -->
<td width="85%" align="center" valign="top" bgcolor="#FFFFFF">
<br><h3><b>Welcome, to my web page!</b></h3>
</td>
<!-- end right pane of content area -->

</tr>
</table>
</td></tr>
</table>
</center>

```

The above HTML code segment would produce the following results in a web browser:

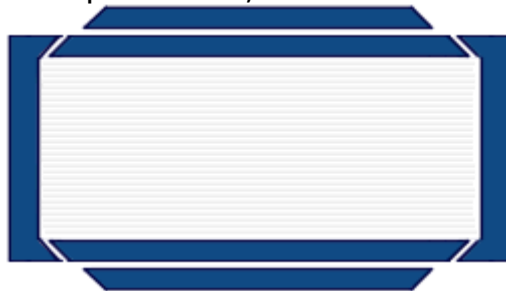
.: Nav Link 1 .: Nav Link 2 .: Nav Link 3 .: Nav Link 4 .: Nav Link 5	Welcome, to my web page!
---	---------------------------------

Tables are used when developing structural table formats within a web document. Because tables can be created using pixel dimensions or percentage values, they can play a crucial role in

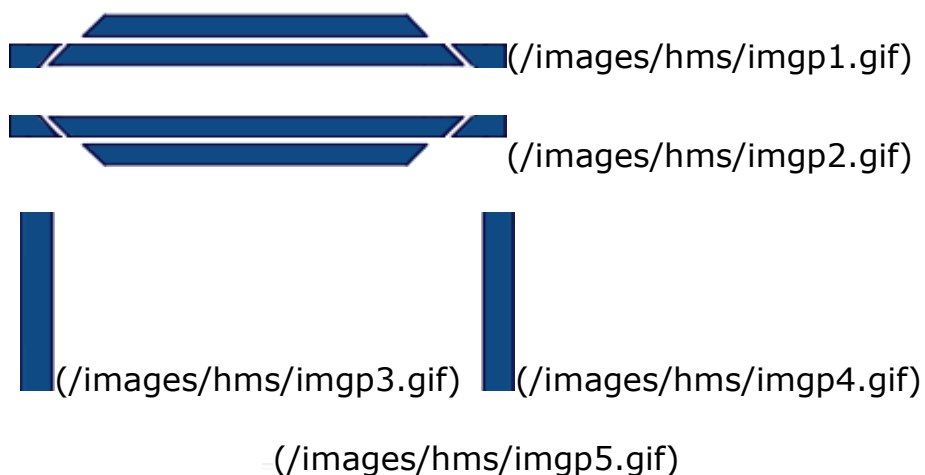
organizing web document content when creating complex web page layouts. The table `<table>` tag is one of the most convenient and popular HTML element because of its powerful and flexible capabilities.

Image Alignment Using Tables

When displaying multiple images, it is recommended to place the images in a table structure using the `<table>` tag so the images will appear in a precise region. Tables are commonly used when designers need to display multiple, smaller portions of an image (which all together represent a larger image) as a "whole" in the web page. For an example of this, consider the following image:



Let's say that for some reason I decide I don't want to display this image as one object. Instead, I want to divide the image into separate portions, and then piece it back together when displaying it on a web page. There are many reasons for this. One reason is to decrease the size of the image. Decreasing size makes loading times faster. I decide to divide the image into the following five pieces:



Four of these pieces are easily seen. The fifth image is extremely small and is located below all other pieces. You may have to highlight the entire region to observe the small image. It is used as a background.

Using a table, I can then piece the image back together. I must be careful to place the images precisely in the correct table row and column. It is best to structure the table using the size (width and height) of the images to be placed within it. The following piece of code would handle our situation:

```
<table cellpadding="0" cellspacing="0" border="0" align="center">
<tr><td colspan="3" width="251"></td></tr>
<tr><td width="17"></td>
<td width="217" background="/images/hms/imgp5.gif"></td>
<td width="17"></td></tr>
<tr><td colspan="3" width="251"></td>
</tr>
</table>
```

The above code would produce the following results:



Since our original image was fairly small in size, there would really be no great benefit of slicing the image into portions and then piecing it back together. I used this example to simply demonstrate how this is done. In reality, structuring images with a table is an advantage for images of great size.

The Alt Attribute

The alt attribute is used to define an "alternate text" for an image. The value of the alt attribute is an author-defined text:

```
<img src= C:\Users\Deen\Desktop\WebPage Design NBTE\nbte.jpg alt="NBTE Logo">
```

The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers.

Basic Notes - Useful Tips

If an HTML file contains ten images - eleven files are required to display the page right. Loading images take time, so it is advised to use images carefully.

Image Tags

Tag	Description
<u></u>	Defines an image
<u><map></u>	Defines an image map
<u><area></u>	Defines a clickable area inside an image map

WEEK Eight

Introduction to CSS

What is CSS?

- **CSS** stands for **Cascading Style Sheets**
 - Styles define **how to display** HTML elements
 - Styles are normally stored in **Style Sheets**
 - Styles were added to HTML 4.0 **to solve a problem**
 - **External Style Sheets** can save you a lot of work
 - External Style Sheets are stored in **CSS files**
 - Multiple style definitions will **cascade** into one
-

CSS Demo

With CSS, your HTML documents can be displayed using different output styles:

Styles Solve a Common Problem

HTML tags were originally designed to define the content of a document. They were supposed to say "This is a header", "This is a paragraph", "This is a table", by using tags like <h1>, <p>, <table>, and so on. The layout of the document was supposed to be taken care of by the browser, without using any formatting tags.

As the two major browsers - Netscape and Internet Explorer - continued to add new HTML tags and attributes (like the tag and the color attribute) to the original HTML specification, it became more and more difficult to create Web sites where the content of HTML documents was clearly separated from the document's presentation layout.

To solve this problem, the World Wide Web Consortium (W3C) - the non profit, standard setting consortium, responsible for standardizing HTML - created STYLES in addition to HTML 4.0.

All major browsers support Cascading Style Sheets.

Style Sheets Can Save a Lot of Work

Styles sheets define HOW HTML elements are to be displayed, just like the font tag and the color attribute in HTML 3.2. Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in your Web, just by editing one single CSS document!

CSS is a breakthrough in Web design because it allows developers to control the style and layout of multiple Web pages all at once. As a Web developer you can define a style for each HTML element and apply it to as many Web pages as you want. To make a global change, simply change the style, and all elements in the Web are updated automatically.

Multiple Styles Will Cascade into One

Style sheets allow style information to be specified in many ways. Styles can be specified inside a single HTML element, inside the <head> element of an HTML page, or in an external CSS file. Even multiple external style sheets can be referenced inside a single HTML document.

Cascading Order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (inside the <head> tag)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style declared inside the <head> tag, in an external style sheet, or in a browser (a default value).

CSS Syntax

The CSS syntax is made up of three parts: a selector, a property and a value:

```
selector {property: value}
```

The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon, and surrounded by curly braces:

```
body {color: black}
```

Note: If the value is multiple words, put quotes around the value:

```
p {font-family: "sans serif"}
```

Note: If you wish to specify more than one property, you must separate each property with a semicolon. The example below shows how to define a center aligned paragraph, with a red text color:

```
p {text-align:center;color:red}
```

To make the style definitions more readable, you can describe one property on each line, like this:

```
p
{
text-align: center;
color: black;
font-family: arial
}
```

Grouping

You can group selectors. Separate each selector with a comma. In the example below we have grouped all the header elements. All header elements will be displayed in green text color:

```
h1,h2,h3,h4,h5,h6
{
color: green
}
```

The class Selector

With the class selector you can define different styles for the same type of HTML element.

Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles:

```
p.right {text-align: right}
p.center {text-align: center}
```

You have to use the class attribute in your HTML document:

```
<p class="right">
This paragraph will be right-aligned.
</p>
<p class="center">
This paragraph will be center-aligned.
</p>
```

Note: To apply more than one class per given element, the syntax is:

```
<p class="center bold">
This is a paragraph.
</p>
```

The paragraph above will be styled by the class "center" AND the class "bold".

You can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class. In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align: center}
```

In the code below both the h1 element and the p element have class="center". This means that both elements will follow the rules in the ".center" selector:

```
<h1 class="center">
This heading will be center-aligned
</h1>
<p class="center">
This paragraph will also be center-aligned.
</p>
```

💡 Do **NOT** start a class name with a number! It will not work in Mozilla/Firefox.

Add Styles to Elements with Particular Attributes

You can also apply styles to HTML elements with particular attributes.

The style rule below will match all input elements that have a type attribute with a value of "text":

```
input[type="text"] {background-color: blue}
```

The id Selector

You can also define styles for HTML elements with the id selector. The id selector is defined as a #.

The style rule below will match the element that has an id attribute with a value of "green":

```
#green {color: green}
```

The style rule below will match the p element that has an id with a value of "para1":

```
p#para1
{
text-align: center;
color: red
}
```

💡Do **NOT** start an ID name with a number! It will not work in Mozilla/Firefox.

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. A comment will be ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/* This is a comment */
p
{
text-align: center;
```

```
/* This is another comment */
color: black;
font-family: arial
}
```

How to Insert a Style Sheet

When a browser reads a style sheet, it will format the document according to it. There are three ways of inserting a style sheet:

External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

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

The browser will read the style definitions from the file mystyle.css, and format the document according to it.

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
```

Do **NOT** leave spaces between the property value and the units! If you use "margin-left: 20 px" instead of "margin-left: 20px" it will only work properly in IE6 but it will not work in Mozilla/Firefox or Netscape.

Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag, like this:

```
<head>
<style type="text/css">
hr {color: sienna}
p {margin-left: 20px}
```



```
body {background-image: url("images/back40.gif")}
</style>
</head>
```

The browser will now read the style definitions, and format the document according to it.

Note: A browser normally ignores unknown tags. This means that an old browser that does not support styles, will ignore the <style> tag, but the content of the <style> tag will be displayed on the page. It is possible to prevent an old browser from displaying the content by hiding it in the HTML comment element:

```
<head>
<style type="text/css">
<!--
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
-->
</style>
</head>
```

Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color: sienna; margin-left: 20px">
This is a paragraph
</p>
```

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
h3
{
color: red;
```

```
text-align: left;  
font-size: 8pt  
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3  
{  
text-align: right;  
font-size: 20pt  
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color: red;  
text-align: right;  
font-size: 20pt
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

WEEK Nine

CSS Background

The CSS background properties define the background effects of an element.

The CSS background properties allow you to control the background color of an element, set an image as the background, repeat a background image vertically or horizontally, and position an image on a page.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
background	A shorthand property for setting all background properties in one declaration	<i>background-color</i> <i>background-image</i> <i>background-repeat</i> <i>background-attachment</i> <i>background-position</i>	4	1	6	1
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	scroll fixed	4	1	6	1
background-color	Sets the background color of an element	<i>color-rgb</i> <i>color-hex</i> <i>color-name</i> transparent	4	1	4	1
background-image	Sets an image as the background	url(URL) none	4	1	4	1

background-position	Sets the starting position of a background image	top left top center top right center left center center center right bottom left bottom center bottom right <i>x% y%</i> <i>xpos ypos</i>	4	1	6	1
background-repeat	Sets if/how a background image will be repeated	repeat repeat-x repeat-y no-repeat	4	1	4	1

CSS Text

The CSS text properties define the appearance of text.

CSS Text Properties

The CSS text properties allow you to control the appearance of text. It is possible to change the color of a text, increase or decrease the space between characters in a text, align a text, decorate a text, indent the first line in a text, and more.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
color	Sets the color of a text	<i>color</i>	3	1	4	1
direction	Sets the text direction	ltr rtl	6	1	6	2
line-height	Sets the distance between lines	normal <i>number</i>	4	1	4	1

		<i>length</i> %				
letter-spacing	Increase or decrease the space between characters	normal <i>length</i>	4	1	6	1
text-align	Aligns the text in an element	left right center justify	4	1	4	1
text-decoration	Adds decoration to text	none underline overline line-through blink	4	1	4	1
text-indent	Indents the first line of text in an element	<i>length</i> %	4	1	4	1
text-shadow		none <i>color</i> <i>length</i>				
text-transform	Controls the letters in an element	none capitalize uppercase lowercase	4	1	4	1
unicode-bidi		normal embed bidi-override	5			2
white-space	Sets how white space inside an element is handled	normal pre nowrap	5	1	4	1
word-spacing	Increase or decrease the space between words	normal <i>length</i>	6	1	6	1

CSS Font

The CSS font properties define the font in text.

Examples

This example demonstrates how to set a font of a text.

```
<html>
<head>
<style type="text/css">
h3 {font-family: times}
p {font-family: courier}
p.sansserif {font-family: sans-serif}
</style>
</head>

<body>
<h3>This is header 3</h3>
<p>This is a paragraph</p>
<p class="sansserif">This is a paragraph</p>
</body>

</html>
```

CSS Font Properties

The CSS font properties allow you to change the font family, boldness, size, and the style of a text.

Note: In CSS1 fonts are identified by a font name. If a browser does not support the specified font, it will use a default font.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
font	A shorthand property for setting all of the properties for a font in one declaration	<i>font-style</i> <i>font-variant</i> <i>font-weight</i> <i>font-size/line-height</i> <i>font-family</i> caption icon menu message-box small-caption status-bar	4	1	4	1

font-family	A prioritized list of font family names and/or generic family names for an element	<i>family-name</i> <i>generic-family</i>	3	1	4	1
font-size	Sets the size of a font	xx-small x-small small medium large x-large xx-large smaller larger <i>length</i> %	3	1	4	1
font-size-adjust	Specifies an aspect value for an element that will preserve the x-height of the first-choice font	none <i>number</i>	-	-	-	2
font-stretch	Condenses or expands the current font-family	normal wider narrower ultra-condensed extra-condensed condensed semi-condensed semi-expanded expanded extra-expanded ultra-expanded	-	-	-	2
font-style	Sets the style of the font	normal italic oblique	4	1	4	1
font-variant	Displays text in a small-caps font or a normal font	normal small-caps	4	1	6	1
font-weight	Sets the weight of a font	normal bold bolder lighter 100 200 300	4	1	4	1

		400				
		500				
		600				
		700				
		800				
		900				

CSS Border

The CSS border properties define the borders around an element.

CSS Border Properties

The CSS border properties allow you to specify the style and color of an element's border. In HTML we use tables to create borders around a text, but with the CSS border properties we can create borders with nice effects, and it can be applied to any element.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
border	A shorthand property for setting all of the properties for the four borders in one declaration	<i>border-width</i> <i>border-style</i> <i>border-color</i>	4	1	4	1
border-bottom	A shorthand property for setting all of the properties for the bottom border in one declaration	<i>border-bottom-width</i> <i>border-style</i> <i>border-color</i>	4	1	6	1
border-bottom-color	Sets the color of the bottom border	<i>border-color</i>	4	1	6	2
border-bottom-style	Sets the style of the bottom border	<i>border-style</i>	4	1	6	2
border-bottom-width	Sets the width of	thin	4	1	4	1

width	the bottom border	medium thick <i>length</i>				
border-color	Sets the color of the four borders, can have from one to four colors	<i>color</i>	4	1	6	1
border-left	A shorthand property for setting all of the properties for the left border in one declaration	<i>border-left-width</i> <i>border-style</i> <i>border-color</i>	4	1	6	1
border-left-color	Sets the color of the left border	<i>border-color</i>	4	1	6	2
border-left-style	Sets the style of the left border	<i>border-style</i>	4	1	6	2
border-left-width	Sets the width of the left border	thin medium thick <i>length</i>	4	1	4	1
border-right	A shorthand property for setting all of the properties for the right border in one declaration	<i>border-right-width</i> <i>border-style</i> <i>border-color</i>	4	1	6	1
border-right-color	Sets the color of the right border	<i>border-color</i>	4	1	6	2
border-right-style	Sets the style of the right border	<i>border-style</i>	4	1	6	2
border-right-width	Sets the width of the right border	thin medium thick <i>length</i>	4	1	4	1
border-style	Sets the style of the four borders, can have from one to four styles	none hidden dotted dashed solid double groove ridge inset	4	1	6	1

		outset				
border-top	A shorthand property for setting all of the properties for the top border in one declaration	<i>border-top-width</i> <i>border-style</i> <i>border-color</i>	4	1	6	1
border-top-color	Sets the color of the top border	<i>border-color</i>	4	1	6	2
border-top-style	Sets the style of the top border	<i>border-style</i>	4	1	6	2
border-top-width	Sets the width of the top border	thin medium thick <i>length</i>	4	1	4	1
border-width	A shorthand property for setting the width of the four borders in one declaration, can have from one to four values	thin medium thick <i>length</i>	4	1	4	1

CSS Margin

The CSS margin properties define the space around elements.

CSS Margin Properties

The CSS margin properties define the space around elements. It is possible to use negative values to overlap content. The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used to change all of the margins at once.

Note: Netscape and IE give the body tag a default margin of 8px. Opera does not! Instead, Opera applies a default padding of 8px, so if one wants to adjust the margin for an entire page and have it display correctly in Opera, the body padding must be set as well!

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
margin	A shorthand property for setting the margin properties in one declaration	<i>margin-top</i> <i>margin-right</i> <i>margin-bottom</i> <i>margin-left</i>	4	1	4	1
margin-bottom	Sets the bottom margin of an element	auto <i>length</i> %	4	1	4	1
margin-left	Sets the left margin of an element	auto <i>length</i> %	3	1	4	1
margin-right	Sets the right margin of an element	auto <i>length</i> %	3	1	4	1
margin-top	Sets the top margin of an element	auto <i>length</i> %	3	1	4	1

CSS Padding

The CSS padding properties define the space between the element border and the element content.

CSS Padding Properties

The CSS padding properties define the space between the element border and the element content. Negative values are not allowed. The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property is also created to control multiple sides at once.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
padding	A shorthand property for setting all of the padding properties in one declaration	<i>padding-top</i> <i>padding-right</i> <i>padding-bottom</i> <i>padding-left</i>	4	1	4	1
padding-bottom	Sets the bottom padding of an element	<i>length</i> <i>%</i>	4	1	4	1
padding-left	Sets the left padding of an element	<i>length</i> <i>%</i>	4	1	4	1
padding-right	Sets the right padding of an element	<i>length</i> <i>%</i>	4	1	4	1
padding-top	Sets the top padding of an element	<i>length</i> <i>%</i>	4	1	4	1

CSS List

The CSS list properties allow you to place the list-item marker, change between different list-item markers, or set an image as the list-item marker.

CSS List Properties

The CSS list properties allow you to place the list-item marker, change between different list-item markers, or set an image as the list-item marker.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
list-style	A shorthand property for setting all of the properties	<i>list-style-type</i> <i>list-style-position</i>	4	1	6	1

	for a list in one declaration	<i>list-style-image</i>				
list-style-image	Sets an image as the list-item marker	none <i>url</i>	4	1	6	1
list-style-position	Sets where the list-item marker is placed in the list	inside outside	4	1	6	1
list-style-type	Sets the type of the list-item marker	none disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-alpha upper-alpha lower-greek lower-latin upper-latin hebrew armenian georgian cjk-ideographic hiragana katakana hiragana-iroha katakana-iroha	4	1	4	1
marker-offset		auto <i>length</i>		1	7	2

CSS Table

The CSS table properties allow you to set the layout of a table.

CSS Table Properties

The CSS table properties allow you to set the layout of a table.

Browser support: IE: Internet Explorer, M: Mac IE only, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
border-collapse	Sets whether the table borders are collapsed into a single border or detached as in standard HTML	collapse separate	5	1	7	2
border-spacing	Sets the distance that separates cell borders (only for the "separated borders" model)	<i>length length</i>	5M	1	6	2
caption-side	Sets the position of the table caption	top bottom left right	5M	1	6	2
empty-cells	Sets whether or not to show empty cells in a table (only for the "separated borders" model)	show hide	5M	1	6	2
table-layout	Sets the algorithm used to display the table cells, rows, and columns	auto fixed	5	1	6	2

WEEK Ten

HTML Scripts

A *script* is a small, embedded program that can add interactivity to your website. For example, a script could generate a pop-up alert box message, or provide a dropdown menu.

Scripts offer authors a means to extend HTML documents in highly active and interactive ways. For example:

- Scripts may be evaluated as a document loads to modify the contents of the document dynamically.
- Scripts may accompany a form to process input as it is entered. Designers may dynamically fill out parts of a form based on the values of other fields. They may also ensure that input data conforms to predetermined ranges of values, that fields are mutually consistent, etc.
- Scripts may be triggered by events that affect the document, such as loading, unloading, element focus, mouse movement, etc.
- Scripts may be linked to form controls (e.g., buttons) to produce graphical user interface elements.

There are two types of scripts authors may attach to an HTML document:

- Those that are executed one time when the document is loaded by the user agent. Scripts that appear within a SCRIPT element are executed when the document is loaded. For user agents that cannot or will not handle scripts, authors may include alternate content via the NOSCRIPT element.
- Those that are executed every time a specific event occurs. These scripts may be assigned to a number of elements via the [intrinsic event](#) attributes.

Because HTML doesn't actually have scripting capability, you need to use the `<script>` tag to generate a script, using a scripting language.

The `<script>` tags tell the browser to expect a script in between them. You specify the language using the `type` attribute. The most popular scripting language on the web is JavaScript.

Adding a Script

You can specify whether to make a script run automatically (as soon as the page loads), or after the user has done something (like click on a link).

In either case, a generally accepted convention is to place your scripts between the `<head></head>` tags. This ensures that the script is ready to run when it is called.

HTML Code:

```
<script type="text/javascript">  
  alert("I am a script. I ran first!")  
</script>
```

This would open a JavaScript alert as soon as the page loads.

Triggering a Script

In many cases, you won't want the script to run automatically. You might only want the script to run if the user does something (like hover over a link), or once the page has finished loading.

These actions are called *intrinsic events* (*events* for short). There are 18 pre-defined intrinsic events that can trigger a script to run. You use *event handlers* to tell the browser which event should trigger which script. These are specified as an attribute within the HTML tag.

Lets say you want a message to display in the status bar whenever the user hovers over a link. The act of hovering over the link is an

event which is handled by the onmouseover event handler. You add the onmouseover attribute to the HTML tag to tell the browser what to do next.

HTML Code:

```
Treat yourself to a <a href="http://www.great-workout.com/killer-ab-workout.cfm" onMouseover="window.status='Go on, you know you want to'; return true">Killer Ab Workout</a>
```

This results in:

Treat yourself to a Killer Ab Workout

Status bar messages aren't supported by all browsers. If you see no change to the status bar, it's likely that your browser doesn't support this piece of JavaScript.

Attribute definitions

onload

The onload event occurs when the user agent finishes loading a window or all frames within a FRAMESET. This attribute may be used with BODY and FRAMESET elements.

onunload

The onunload event occurs when the user agent removes a document from a window or frame. This attribute may be used with BODY and FRAMESET elements.

onclick

The onclick event occurs when the pointing device button is clicked over an element. This attribute may be used with most elements.

ondblclick

The ondblclick event occurs when the pointing device button is double clicked over an element. This attribute may be used with most elements.

onmousedown

The onmousedown event occurs when the pointing device button is pressed over an element. This attribute may be used with most elements.

onmouseup

The `onmouseup` event occurs when the pointing device button is released over an element. This attribute may be used with most elements.

`onmouseover`

The `onmouseover` event occurs when the pointing device is moved onto an element. This attribute may be used with most elements.

`onmousemove`

The `onmousemove` event occurs when the pointing device is moved while it is over an element. This attribute may be used with most elements.

`onmouseout`

The `onmouseout` event occurs when the pointing device is moved away from an element. This attribute may be used with most elements.

`onfocus`

The `onfocus` event occurs when an element receives focus either by the pointing device or by tabbing navigation. This attribute may be used with the following elements: `A`, `AREA`, `LABEL`, `INPUT`, `SELECT`, `TEXTAREA`, and `BUTTON`.

`onblur`

The `onblur` event occurs when an element loses focus either by the pointing device or by tabbing navigation. It may be used with the same elements as `onfocus`.

`onkeypress`

The `onkeypress` event occurs when a key is pressed and released over an element. This attribute may be used with most elements.

`onkeydown`

The `onkeydown` event occurs when a key is pressed down over an element. This attribute may be used with most elements.

`onkeyup`

The `onkeyup` event occurs when a key is released over an element. This attribute may be used with most elements.

`onsubmit`

The `onsubmit` event occurs when a form is submitted. It only applies to the `FORM` element.

`onreset`

The `onreset` event occurs when a form is reset. It only applies to the `FORM` element.

`onselect`

The `onselect` event occurs when a user selects some text in a text field. This attribute may be used with the `INPUT` and `TEXTAREA` elements.

`onchange`

The onchange event occurs when a control loses the input focus *and* its value has been modified since gaining focus. This attribute applies to the following elements: INPUT, SELECT, and TEXTAREA.

It is possible to associate an action with a certain number of events that occur when a user interacts with a user agent. Each of the "intrinsic events" listed above takes a value that is a script. The script is executed whenever the event occurs for that element. The [syntax of script data](#) depends on the scripting language.

Control elements such as INPUT, SELECT, BUTTON, TEXTAREA, and LABEL all respond to certain intrinsic events. When these elements do not appear within a form, they may be used to augment the graphical user interface of the document.

For instance, authors may want to include press buttons in their documents that do not submit a form but still communicate with a server when they are activated.

The following examples show some possible control and user interface behavior based on intrinsic events.

In the following example, userName is a required text field. When a user attempts to leave the field, the onblur event calls a JavaScript function to confirm that userName has an acceptable value.

```
<INPUT NAME="userName"
onblur="validUserName(this.value)">
```

Here is another JavaScript example:

```
<INPUT NAME="num"
onchange="if (!checkNum(this.value, 1, 10))
{this.focus();this.select();} else {thanks()}"
VALUE="0">
```

Here is a VBScript example of an event handler for a text field:

```
<INPUT name="edit1" size="50">
<SCRIPT type="text/vbscript">
Sub edit1_changed()
If edit1.value = "abc" Then
button1.enabled = True
Else
button1.enabled = False
End If
End Sub
```

```
</SCRIPT>
```

Here is the same example using Tcl:

```
<INPUT name="edit1" size="50">
<SCRIPT type="text/tcl">
  proc edit1_changed {} {
    if {[edit value] == abc} {
      button1 enable 1
    } else {
      button1 enable 0
    }
  }
  edit1 onChange edit1_changed
</SCRIPT>
```

Here is a JavaScript example for event binding within a script. First, here's a simple click handler:

```
<BUTTON type="button" name="mybutton" value="10">
<SCRIPT type="text/javascript">
  function my_onclick() {
    . . .
  }
  document.form.mybutton.onclick = my_onclick
</SCRIPT>
</BUTTON>
```

Here's a more interesting window handler:

```
<SCRIPT type="text/javascript">
  function my_onload() {
    . . .
  }

  var win = window.open("some/other/URI")
  if (win) win.onload = my_onload
</SCRIPT>
```

In Tcl this looks like:

```
<SCRIPT type="text/tcl">
  proc my_onload {} {
    . . .
  }
  set win [window open "some/other/URI"]
```

```

    if {$win != ""} {
        $win onload my_onload
    }
</SCRIPT>

```

Note that "document.write" or equivalent statements in intrinsic event handlers create and write to a new document rather than modifying the current one.

Dynamic modification of documents

Scripts that are executed when a document is loaded may be able to modify the document's contents dynamically. The ability to do so depends on the scripting language itself (e.g., the "document.write" statement in the HTML object model supported by some vendors).

The dynamic modification of a document may be modeled as follows:

1. All SCRIPT elements are evaluated in order as the document is loaded.
2. All script constructs within a given SCRIPT element that generate SGML CDATA are evaluated. Their combined generated text is inserted in the document in place of the SCRIPT element.
3. The generated CDATA is re-evaluated.

HTML documents are constrained to conform to the HTML DTD both before and after processing any SCRIPT elements.

The following example illustrates how scripts may modify a document dynamically. The following script:

```

<TITLE>Test Document</TITLE>
<SCRIPT type="text/javascript">
    document.write("<p><b>Hello World!</b>")
</SCRIPT>

```

Has the same effect as this HTML markup:

```

<TITLE>Test Document</TITLE>
<P><B>Hello World!</B>

```

Calling an External Script

You can also place your scripts into their own file, then call that file from your HTML document. This is useful if you want the same scripts available to multiple HTML documents - it saves you from having to "copy and paste" the scripts into each HTML document. This makes it much easier to maintain your website.

HTML Code:

```
<script type="text/javascript" src="external_scripts.js"></script>
```

Hide Scripts from Older Browsers

Although most (if not all) browsers these days support scripts, some older browsers don't. If a browser doesn't support JavaScript, instead of running your script, it would display the code to the user. To prevent this from happening, you can simply place HTML comments around the script. Older browsers will ignore the script, while newer browsers will run it.

HTML Code:

```
<script type="text/javascript">  
  <-- Hide from older browsers  
    alert("I am a script. I ran first!")  
  Unhide -->  
</script>
```

Alternate Information for Older Browsers

You can also provide alternative info for users whose browsers don't support scripts (and for users who have disabled scripts). You do this using the `<noscript>` tag.

HTML Code:

```
<script type="text/javascript">
  <-- Hide from older browsers
    alert("I am a script. I ran first!")
  Unhide -->
</script>

<noscript>
You need JavaScript enabled to view this page.
</noscript>
```

Set a Default Scripting Language

You can specify a default scripting language for all your script tags to use. This saves you from having to specify the language everytime you use a script tag within the page.

HTML Code:

```
<meta http-equiv="Content-Script-Type" content="text/JavaScript"
/>
```

Note that you can still override the default by specifying a language within the script tag.

WEEK Eleven

Introduction to JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, and Opera.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML / XHTML
-

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
 - JavaScript is a scripting language
 - A scripting language is a lightweight programming language
 - JavaScript is usually embedded directly into HTML pages
 - JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
 - Everyone can use JavaScript without purchasing a license
-

Are Java and JavaScript the Same?

NO!

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do?

- **JavaScript gives HTML designers a programming tool** - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
 - **JavaScript can put dynamic text into an HTML page** - A JavaScript statement like this: `document.write("<h1>" + name + "</h1>")` can write a variable text into an HTML page
 - **JavaScript can react to events** - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
 - **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element
 - **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
 - **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
 - **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer
-

The Real Name is ECMAScript

JavaScript's official name is "ECMAScript". The standard is developed and maintained by the [ECMA organisation](#).

ECMA-262 is the official JavaScript standard. The standard is based on JavaScript (Netscape) and JScript (Microsoft).

The language was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all Netscape and Microsoft browsers since 1996.

The development of ECMA-262 started in 1996, and the first edition was adopted by the ECMA General Assembly in June 1997.

The standard was approved as an international ISO (ISO/IEC 16262) standard in 1998.

JavaScript Designs

The HTML `<script>` tag is used to insert a JavaScript into an HTML page.

How to Put a JavaScript Into an HTML Page

```
<html>
<body>
  <script type="text/javascript">
    document.write("Hello World!");
  </script>
</body>
</html>
```

The code above will produce this output on an HTML page:

Hello World!

Example Explained

To insert a JavaScript into an HTML page, we use the `<script>` tag. Inside the `<script>` tag we use the `type` attribute to define the scripting language.

So, the `<script type="text/javascript">` and `</script>` tells where the JavaScript starts and ends:

```
<html>
<body>
  <script type="text/javascript">
    ...
  </script>
</body>
</html>
```

The word **document.write** is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
</html>
```

http://www.w3schools.com/js/tryit.asp?filename=tryjs_text

Note: If we had not entered the <script> tag, the browser would have treated the document.write("Hello World!") command as pure text, and just write the entire line on the page.

HTML Comments to Handle Simple Browsers

Browsers that do not support JavaScript will display JavaScript as page content.

To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag can be used to "hide" the JavaScript. Just add an HTML comment tag <!-- before the first JavaScript statement, and a --> (end of comment) after the last JavaScript statement.

```
<html>
<body>
<script type="text/javascript">
<!--
document.write("Hello World!");
//-->
</script>
</body>
</html>
```

The two forward slashes at the end of comment line (//) is the JavaScript comment symbol. This prevents JavaScript from executing the --> tag.

Where to Put the JavaScript

JavaScripts in the body section will be executed WHILE the page loads.

JavaScripts in the head section will be executed when CALLED.

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

Scripts in the head section: Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
```

Scripts in the body section: Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
```

Scripts in both the body and the head section: You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
<script type="text/javascript">
....
</script>
```

```
</body>
```

Using an External JavaScript

Sometimes you might want to run the same JavaScript on several pages, without having to write the same script on every page.

To simplify this, you can write a JavaScript in an external file. Save the external JavaScript file with a .js file extension.

Note: The external script cannot contain the `<script>` tag!

To use the external script, point to the .js file in the "src" attribute of the `<script>` tag:

```
<html>
<head>
  <script src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

Note: Remember to place the script exactly where you normally would write the script!

JavaScript Statements

JavaScript is a sequence of statements to be executed by the browser.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Statements

A JavaScript statement is a command to the browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

```
document.write("Hello Dolly");
```

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript Code

JavaScript code (or just JavaScript) is a sequence of JavaScript statements.

Each statement is executed by the browser in the sequence they are written.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
```

```
document.write("<p>This is another paragraph</p>");  
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks.

Blocks start with a left curly bracket {, and ends with a right curly bracket }.

The purpose of a block is to make the sequence of statements execute together.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">  
{  
  document.write("<h1>This is a header</h1>");  
  document.write("<p>This is a paragraph</p>");  
  document.write("<p>This is another paragraph</p>");  
}  
</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

JavaScript Comments

JavaScript comments can be used to make the code more readable.

Comments can be added to explain the JavaScript, or to make it more readable.

Single line comments start with //.

This example uses single line comments to explain the code:

```
<script type="text/javascript">  
// This will write a header:  
document.write("<h1>This is a header</h1>");  
// This will write two paragraphs:  
document.write("<p>This is a paragraph</p>");  
document.write("<p>This is another paragraph</p>");  
</script>
```

JavaScript Multi-Line Comments

Multi line comments start with `/*` and end with `*/`.

This example uses a multi line comment to explain the code:

```
<script type="text/javascript">
/*
The code below will write
one header and two paragraphs
*/
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

Using Comments to Prevent Execution

In this example the comment is used to prevent the execution of a single code line:

```
<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
//document.write("<p>This is another paragraph</p>");
</script>
```

In this example the comments is used to prevent the execution of multiple code lines:

```
<script type="text/javascript">
/*
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
*/
</script>
```

Using Comments at the End of a Line

In this example the comment is placed at the end of a line:

```
<script type="text/javascript">
document.write("Hello"); // This will write "Hello"
document.write("Dolly"); // This will write "Dolly"
</script>
```

JavaScript Variables

Variables are "containers" for storing information.

Do you remember algebra from school? $x=5$, $y=6$, $z=x+y$

Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?

These letters are called **variables**, and variables can be used to hold values ($x=5$) or expressions ($z=x+y$).

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like x , or a more descriptive name, like `carname`.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Example

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

[This example will show you how](#)

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var statement**:

```
var x;
```

```
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;  
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared.

These statements:

```
x=5;  
carname="Volvo";
```

have the same effect as:

```
var x=5;  
var carname="Volvo";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;  
var x;
```

After the execution of the statements above, the variable **x** will still have the value of **5**. The value of **x** is not reset (or cleared) when you redeclare it.

JavaScript Arithmetic

As with algebra, you can do arithmetic operations with JavaScript variables:

```
y=x-5;
z=y+5;
```

You will learn more about the operators that can be used in the next chapter of this tutorial.

JavaScript Operators

The operator = is used to assign values.

The operator + is used to add values.

The assignment operator = is used to assign values to JavaScript variables.

The arithmetic operator + is used to add values together.

```
y=5;
z=2;
x=y+z;
```

The value of x, after the execution of the statements above is 7.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

Look at these examples:

```
x=5+5;  
document.write(x);  
  
x="5"+"5";  
document.write(x);  
  
x=5+"5";  
document.write(x);  
  
x="5"+5;  
document.write(x);
```

The rule is:

If you add a number and a string, the result will be a string.

WEEK Twelve

What is DHTML?

DHTML is the combination of HTML and JavaScript. It is the combination of several built-in browser features in fourth generation browsers that enable a web page to be more dynamic

DHTML is NOT a scripting language (like JavaScript), but merely a browser feature- or enhancement- that gives your browser the ability to be dynamic. What you really want to learn is not DHTML itself, but rather, the syntax needed to use DHTML. Before anything else, you may want to quickly visit [Dynamic Drive](#) to see what this language is capable of.

Like I said, DHTML is a collection of features that together, enable your web page to be dynamic. I think its important now to define just what the creators of DHTML meant when they say "dynamic". "Dynamic" is defined as the ability of the browser to alter a web page's look and style *after* the document has loaded. I remember when I was learning JavaScript, I was taught that you could use the document.write() method of JavaScript to create webpages on the fly. For example:

```
<script>
document.write("This is text created on the fly!")
</script>
```

"Not bad", I remember saying to myself. But what if I wanted to create content not only on the fly, but on *demand*? Naive I was then, I tried doing just that, by nesting the above code in a function, and calling it via a form button:

```
<input type="button"
onClick="writeconent()" value="text">
```

Pressing the button was nothing short of a big disappointment. My entire web page was erased, and I was left with only the text the function produced.

That was back then. Now, with the introduction of DHTML, I can alter content on a web page on demand, whenever I bloody feel like it, without having the browser erase everything else. That's what DHTML is all about. The ability of the browser to change look and style even after the document has loaded.

Now that I've got you all excited, I think it's only fair that I put a damper on it. The technology of DHTML is currently at its development stage, with NS 4 and IE 4 differing quite greatly in their implementation of this great technology. Its currently not possible to write one DHTML code and expect it to function in both browsers

properly. Furthermore, the two browsers are at different stages in their development of DHTML; from my own knowledge and what I've heard, DHTML in IE 4 is far more powerful and versatile than NS 4's. I don't want to help spread any propaganda, so I'll leave it at that

DHTML in IE 4

As I began taking on and learning IE 4's implementation of DHTML, and realized that there's a lot more to it than its NS counterpart. DHTML in IE does not rely on any one tag, but rather, new objects and properties that stem out of the usual HTML tags you're used to working with, such as <div> and <table>. It's a lot more powerful, but at the same time, and lot more complicated to grasp.

The style object of IE 4

HTML elements in IE 4 now all support a style object, which is essentially the "dynamic" object used to manipulate the look and "feel" of that element. Like the <layer> tag, elements can also be assigned an "id" attribute, which can then be used to identify it during scripting. For example:

```
<div id="adiv"></div>
```

In your script, the syntax required to access the style object of "adiv" would look like this: `adiv.style`

The style object contains many properties, and by manipulating these properties, you can alter the look of an element, dynamically. I'll show some of these properties now:

Important properties of the style object	
backgroundColor	The background color of the element
backgroundImage	The background image of the element
color	The color of the element
position	The position type of the element. Accepted values are "absolute" and "relative"
pixelWidth	The width of the element
pixelHeight	The height of the element
pixelLeft	The position of the element in relation to the x coordinates
pixelTop	The position of the element in relation to the y coordinates

The properties above only represent a subset of the total supported properties, but are the most commonly used ones. The basic syntax to manipulating any style property is the same, which I'll show in a minute. By accessing these properties, we can change the look and style of most HTML elements (as opposed to just the <layer> tag in Netscape)!

Here's a simple demonstration. The below text changes color when the mouse moves over it:

Move your mouse here

Here's the source code to the above text:

```
<span id="sometext" onmouseover="sometext.style.color='red'"
onmouseout="sometext.style.color='black'">Move your mouse here</span>
```

Notice how I changed the text's color:

```
sometext.style.color='red'
```

I first used the element's id to gain access to it, then, through the style object and finally the style's color property, I was able to easily change the color of the text on demand!

All style properties are read/write, and are accessed in a similar manner: element id->style object->property name.

Here is another example that expands an image when the mouse is over it, and reverts it back to its original size when the mouse moves out:



```


<script language="JavaScript1.2"><!--
function enlarge() {
    aimage.style.pixelWidth=164
    aimage.style.pixelHeight=202
}

function revertback() {
    aimage.style.pixelWidth=82
    aimage.style.pixelHeight=101
}
//--></script>
```

Yes, I know its not exactly the most practical example in the world, but it does illustrate DHTML at work quite well. The image changes dimensions on demand, without the need to reload the document. That's something JavaScript alone can never do.

Dynamic content

If you're not yet scared off by all the differences in syntax and functionality between DHTML in NS 4 and IE 4, you're ready to learn how to make content on your page dynamic, or change on demand!

Dynamic content in IE 4

In IE 4, dynamic content is realized through a special property called `innerHTML` that exists on the `` and `<div>` tag. Just set this property to a new HTML value, and the contents inside that span or div is instantly updated to the new value! I'll illustrate how it's done by modifying the above example to create dynamic content for IE 4 users:

```
<div id="mydiv"></div>

<script language="JavaScript1.2">
var thecontents=new Array()
thecontents[0]="How are you today?"
thecontents[1]="I am fine, thank you."
thecontents[2]="Well, nice talking to you!"
var current=0

function changecontent() {
  mydiv.innerHTML=thecontents[current]
  if (current==2) current=0
  else current++
  setTimeout("changecontent()",3000)
}

window.onload=changecontent
//--></script>
```

How are you today?

Same results, just a different way to get there!

Moving elements around in the document

If you like working with animations, you'll be glad to know that with DHTML, the entire web page is now your drawing board! You can create content that fly all over the screen freely. In Netscape, this is done by manipulating the `left` and `top` attributes of the `<layer>` tag. In IE 4, the same thing is accomplished by altering the `pixelLeft` and `pixelTop` properties of the `style` object.

Moving elements in IE 4

By the way, the day when NS and IE agree upon one implementation of DHTML is the day I can stop writing two versions of everything (just letting out a little frustration). Moving an element in IE 4 involves basically first wrapping that element either inside a positioned span or div, then changing the span or div's `pixelLeft` and `pixelTop` properties. It sounds complicated, but is actually very simple:



```
<div id="spaceship" style="position:relative">

</div>

<script><!--
function moving2() {
  if (spaceship.style.pixelLeft<1000)
    spaceship.style.pixelLeft+=5
  moveid2=setTimeout("moving2()",50)
}

function come_back2() {
  clearTimeout(moveid2)
  spaceship.style.pixelLeft=0
}
//--></script>

<form>
<input type="button" value="Move" onClick="moving2()">
<input type="button" value="Come back" onClick="come_back2()">
</form>
```

What I did first was set the outside `<div>` called "spaceship" to a position of relative, which is necessary to make the element movable (you could also set it to a value of "absolute"). Then, by manipulating the `pixelWidth` property of it's style object, the element moves.

Creating cross-browser DHTML

Before "true" cross-browser DHTML becomes available (in other words, when NS and IE comes to their senses), cross-browser DHTML basically means using various scripting techniques you picked during those JavaScript years to sniff out which browser the user is using, and execute the code intended for that browser. In this lesson, I'll first illustrate a way of creating a "cross-browser" layer, then show you a scripting technique I recently learned that allows you to easily sniff out the browser type of the surfer.

Creating a "cross-browser" layer

Ok, so we've got NS that understands the `<layer>` tag, and IE that understands the `` and `<div>`. If we wanted to create a simple DHTML effect such as a moving image, we would usually need to use two tags- A layer tag for NS 4, and either a div or span tag for IE 4. Not exactly pretty, uh? Well, I recently learned that there is actually a way to create a "cross-browser" layer that uses only one tag, although its a little buggy on the NS side. Apparently NS 4 treats an absolutely positioned div the same as a layer. So, without any further delay, here's an example of a cross browser layer:

```
<div id="crosslayer" style="position:absolute"></div>
```

NS 4 treats the above div exactly the same as it would with a layer. Like any other layer, to access it, we would first go through the document object, then the layer's id: document.crosslayer

In IE 4, we would simply use the div's id: crosslayer

I found that in NS, specifying a layer this way, while convenient in terms of cross-browser compatibility, has one major drawback. Such a layer doesn't always behave the way a normal layer should, and can sometimes actually crash the browser. Just be prepared to expect the unexpected!

Browser sniffing- object detection

Up until recently, whenever I wished to determine the browser type of my surfers, I would use the navigator object, like most JavaScript programmers would. The below illustrates using this object to sniff out both NS 4 and IE 4:

```
var ns4= (navigator.appName=="Netscape"&&navigator.appVersion>=4)
var ie4= (navigator.appName=="Microsoft Internet
Explorer"&&navigator.appVersion>=4)
```

Personally, I hate using the navigator object- its so complicated to use (just look at the above mess!). Well, I have good news to bring to you. There is actually a lot quicker way to sniff out various browsers, and its called object detection.

The idea is based on the way JavaScript works. If the browser does NOT support a particular object, JavaScript returns null when you reference it. Knowing this fact, we can use an object reference in your if statement (in place of the navigator object) to determine the browser of the user.

Let's do an example. We know that NS 3+ and IE 4+ support the document.images object. If we wanted to sniff out these browsers, we would do this:

```
if (document.images)
    alert("You are using NS 3+ or IE 4+")
```

Translating the above into English, it reads: "If the browser supports the images object (which only NS 3+ and IE 4+ do), alert a message.

Think of object detection as an indirect way of determining the browser type of the user. Instead of directly determining the name and version of the user's browser (through the navigator object), object detection is a more generic, less hassling browser sniffing technique.

So, how can we use object detection to sniff out NS 4 and IE 4? Well, only NS 4 supports the document.layers object, and only IE 4 supports document.all. We can use this knowledge to easily determine whether the user is using NS 4, IE 4, or both:

```
if (document.layers)
  alert("You are using NS 4+")
if (document.all)
  alert("You are using IE 4+")
if (document.layers||document.all)
  alert("You are using either NS 4 or IE 4+")
```

Now you never have to return to the messy navigator object to do your browser sniffings!

WEEK Thirteen

Introduction to Web Multimedia

Multimedia is pictures, sounds, music, animations and videos.

Modern web browsers have support for many multimedia formats.

What is Multimedia?

Multimedia is everything you can hear or see: texts, books, pictures, music, sounds, CDs, videos, DVDs, Records, Films, and more.

Multimedia comes in many different formats. On the Internet you will find many of these elements embedded in web pages, and today's web browsers have support for a number of multimedia formats.

In this tutorial you will learn about different multimedia formats and how to use them in your web pages.

Browser Support

The first Internet browsers had support for text only, and even the text support was limited to a single font in a single color, and little or nothing else.

Then came web browsers with support for colors, fonts and text styles, and the support for pictures was added.

The support for sounds, animations and videos is handled in different ways by different browsers. Some elements can be handled inline, some requires a plug-in and some requires an ActiveX control.

You will learn more about this in the next chapters.

Multimedia Formats

Multimedia elements (like sounds or videos) are stored in media files.

The most common way to discover the media type is to look at the file extension.

When a browser sees the file extensions .htm or .html, it will assume that the file is an HTML page. The .xml extension indicates an XML file, and the .css extension indicates a style sheet.

Picture formats are recognized by extensions like .gif and .jpg.

Multimedia elements also have their own file formats with different extensions.

You will learn more about media file extensions in the next chapters.

Multimedia Sound Formats

Sound can be stored in many different formats.

The MIDI Format

The MIDI (Musical Instrument Digital Interface) is a format for sending music information between electronic music devices like synthesizers and PC sound cards.

The MIDI format was developed in 1982 by the music industry. The MIDI format is very flexible and can be used for everything from very simple to real professional music making.

MIDI files do not contain sampled sound, but a set of digital musical instructions (musical notes) that can be interpreted by your PC's sound card.

The downside of MIDI is that it cannot record sounds (only notes). Or, to put it another way: It cannot store songs, only tunes.

The upside of the MIDI format is that since it contains only instructions (notes), MIDI files can be extremely small. The example above is only 23K in size but it plays for nearly 5 minutes.

The MIDI format is supported by many different software systems over a large range of platforms. MIDI files are supported by all the most popular Internet browsers.

Sounds stored in the MIDI format have the extension .mid or .midi.

The RealAudio Format

The RealAudio format was developed for the Internet by Real Media. The format also supports video.

The format allows streaming of audio (on-line music, Internet radio) with low bandwidths. Because of the low bandwidth priority, quality is often reduced.

Sounds stored in the RealAudio format have the extension .rm or .ram.

The AU Format

The AU format is supported by many different software systems over a large range of platforms.

Sounds stored in the AU format have the extension .au.

The AIFF Format

The AIFF (Audio Interchange File Format) was developed by Apple.

AIFF files are not cross-platform and the format is not supported by all web browsers.

Sounds stored in the AIFF format have the extension .aif or .aiff.

The SND Format

The SND (Sound) was developed by Apple.

SND files are not cross-platform and the format is not supported by all web browsers.

Sounds stored in the SND format have the extension .snd.

The WAVE Format

The WAVE (waveform) format is developed by IBM and Microsoft.

It is supported by all computers running Windows, and by all the most popular web browsers.

Sounds stored in the WAVE format have the extension .wav.

The MP3 Format (MPEG)

MP3 files are actually MPEG files. But the MPEG format was originally developed for video by the Moving Pictures Experts Group. We can say that MP3 files are the sound part of the MPEG video format.

MP3 is one of the most popular sound formats for music recording. The MP3 encoding system combines good compression (small files) with high quality. Expect all your future software systems to support it.

Sounds stored in the MP3 format have the extension .mp3, or .mpga (for MPG Audio).

What Format To Use?

The WAVE format is one of the most popular sound format on the Internet, and it is supported by all popular browsers. If you want recorded sound (music or speech) to be available to all your visitors, you should use the WAVE format.

The MP3 format is the new and upcoming format for recorded music. If your website is about recorded music, the MP3 format is the choice of the future.

Multimedia Video Formats

Video can be stored in many different formats.

The AVI Format

The AVI (Audio Video Interleave) format was developed by Microsoft.

The AVI format is supported by all computers running Windows, and by all the most popular web browsers. It is a very common format on the Internet, but not always possible to play on non-Windows computers.

Videos stored in the AVI format have the extension .avi.

The Windows Media Format

The Windows Media format is developed by Microsoft.

Windows Media is a common format on the Internet, but Windows Media movies cannot be played on non-Windows computer without an extra (free) component installed. Some later Windows Media movies cannot play at all on non-Windows computers because no player is available.

Videos stored in the Windows Media format have the extension .wmv.

The MPEG Format

The MPEG (Moving Pictures Expert Group) format is the most popular format on the Internet. It is cross-platform, and supported by all the most popular web browsers.

Videos stored in the MPEG format have the extension .mpg or .mpeg.

The QuickTime Format

The QuickTime format is developed by Apple.

QuickTime is a common format on the Internet, but QuickTime movies cannot be played on a Windows computer without an extra (free) component installed.

Videos stored in the QuickTime format have the extension .mov.

The RealVideo Format

The RealVideo format was developed for the Internet by Real Media.

The format allows streaming of video (on-line video, Internet TV) with low bandwidths. Because of the low bandwidth priority, quality is often reduced.

Videos stored in the RealVideo format have the extension .rm or .ram.

The Shockwave (Flash) Format

The Shockwave format was developed by Macromedia.

The Shockwave format requires an extra component to play. This component comes preinstalled with the latest versions of Netscape and Internet Explorer.

Videos stored in the Shockwave format have the extension .swf.

Playing Sounds On The Web

Sounds can be played "inline" or by a "helper", depending on the HTML element you use.

Inline Sound

When sound is included in a web page, or as part of a web page, it is called inline sound.

Inline sound can be added to a web page by using the `<bgsound>` element or the `` element.

If you plan to use inline sounds in your web applications, be aware that many people find inline sound annoying. Also note that some users might have turned off the inline sound option in their browser.

Our best advice is to include inline sound only in web pages where the user expects to hear the sound. An example of this is a page which opens after the user has clicked on a link to hear a recording.

Using A Helper (Plug-In)

A helper application is a program that can be launched by the browser to "help" playing sound. Helper applications are also called Plug-Ins.

Helper applications can be launched using the `<embed>` element, the `<applet>` element, or the `<object>` element.

One great advantage of using a helper application is that you can let some (or all) of the player settings be controlled by the user.

Most helper applications allows manually (or programmed) control over the volume settings and play functions like rewind, pause, stop and play.

Using The `<bgsound>` Element

Internet Explorer supports an element called `<bgsound>`.

The purpose of this element is to provide a background sound for a web page:

```
<bgsound src="beatles.mid" />
```

The code fragment above displays a MIDI file as background music for a web page.

A list of attributes for the `<bgsound>` element can be found in a later chapter of this tutorial.

Note: The `<bgsound>` element is not a standard HTML or XHTML element. It is supported by Internet Explorer only.

Using The `` Element

Internet Explorer supports the `dynsrc` attribute in the `` element.

The purpose of this element is to embed multimedia elements in web page:

```

```

The code fragment above displays a WAVE file embedded in a web page.

Note: The `dynsrc` attribute is not a standard HTML or XHTML attribute. It is supported by Internet Explorer only.

Using The `<embed>` Element

Internet Explorer and Netscape both support an element called `<embed>`.

The purpose of this element is to embed multimedia elements in web page:

```
<embed src="beatles.mid" />
```

The code fragment above displays a MIDI file embedded in a web page.

A list of attributes for the <embed> element can be found in a later chapter of this tutorial.

Note: The <embed> element is supported by both Internet Explorer and Netscape, but it is not a standard HTML or XHTML element. The World Wide Web Consortium (W3C) recommend using the <object> element instead.

Using The <object> Element

Internet Explorer and Netscape both support an HTML element called <object>.

The purpose of this element is to embed multimedia elements in web page:

```
<object  
classid="clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95">  
<param name="FileName" value="liar.wav" />  
</object>
```

The code fragment above displays a WAVE file embedded in a web page.

A list of attributes for the <object> element can be found in a later chapter of this tutorial.

Using A Hyperlink

If a web page includes a hyperlink to a media file, most browsers will use a "helper application" to play the file:

```
<a href="beatles.mid">  
Click here to play the Beatles  
</a>
```

The code fragment above displays a link to a MIDI file. If the user clicks on the link, the browser will launch a helper application like Windows Media Player to play the MIDI file.

Playing Videos On The Web

Videos can be played "inline" or by a "helper", depending on the HTML element you use.

Inline Videos

When a video is included in a web page it is called inline video.

Inline video can be added to a web page by using the `` element.

If you plan to use inline videos in your web applications, be aware that many people find inline videos annoying. Also note that some users might have turned off the inline video option in their browser.

Our best advice is to include inline videos only in web pages where the user expects to see a video. An example of this is a page which opens after the user has clicked on a link to see the video.

Using A Helper (Plug-In)

A helper application is a program that can be launched by the browser to "help" playing a video. Helper applications are also called Plug-Ins.

Helper applications can be launched using the `<embed>` element, the `<applet>` element, or the `<object>` element.

One great advantage of using a helper application is that you can let some (or all) of the player settings be controlled by the user.

Most helper applications allow manual (or programmed) control over the volume settings and play functions like rewind, pause, stop and play.

Using The `` Element

Internet Explorer supports the `dynsrc` attribute in the `` element.

The purpose of this element is to embed multimedia elements in web page:

```

```

The code fraction above displays an AVI file embedded in a web page.

Note: The `dynsrc` attribute is not a standard HTML or XHTML attribute. It is supported by Internet Explorer only.

Using The <embed> Element

Internet Explorer and Netscape both support an element called <embed>.

The purpose of this element is to embed multimedia elements in web page:

```
<embed src="video.avi" />
```

The code fraction above displays an AVI file embedded in a web page.

A list of attributes for the <embed> element can be found in a later chapter of this tutorial.

Note: The <embed> element is supported by both Internet Explorer and Netscape, but it is not a standard HTML or XHTML element. The World Wide Web Consortium (W3C) recommend using the <object> element instead.

Using The <object> Element

Internet Explorer and Netscape both support an HTML element called <object>.

The purpose of this element is to embed multimedia elements in web page:

```
<object data="video.avi" type="video/avi" />
```

The code fraction above displays an AVI file embedded in a web page.

A list of attributes for the <object> element can be found in a later chapter of this tutorial.

Using A Hyperlink

If a web page includes a hyperlink to a media file, most browsers will use a "helper application" to play the file:

```
<a href="video.avi">  
Click here to play a video file  
</a>
```

The code fraction above displays a link to an AVI file. If the user clicks on the link, the browser will launch a helper application like Windows Media Player to play the AVI file.

Windows Multimedia Formats

Windows media files have the extensions: .asf, .asx, .wma, and .wmv.

The ASF Format

The ASF format (Advanced Streaming Format) is specially designed to run over the Internet.

ASF files can contain audio, video, slide shows, and synchronized events.

ASF files can be highly compressed and can be delivered as a continuous flow of data (on-line TV or radio). Files can be of any size, and can be compressed to match many different bandwidths (connection speeds).

The ASX Format

ASX (Advanced Stream Redirector) files are not media files, but metafiles.

Metafiles provides information about files. ASX files are plain text files used to describe multimedia content:

```
<ASX VERSION="3.0">
<Title>Holiday 2001</Title>
<Entry>
  <ref href="holiday-1.avi"/>
</Entry>
<Entry>
  <ref href="holiday-2.avi"/>
</Entry>
<Entry>
  <ref href="holiday-2.avi"/>
</Entry>
</ASX>
```

The file above describes three multimedia files. When the ASX file is read by a player, the player can play the files described.

The WMA Format

The WMA (Windows Media Audio) format is an audio format developed by Microsoft.

WMA is designed to handle all types of audio content. The files can be highly compressed and can be delivered as a continuous flow of data (on-line radio). WMA files can be of any size, and be compressed to match many different bandwidths (connection speeds).

The WMA format is similar to the ASF format (see above)

The WMV Format

The WMV (Windows Media Video) format is a video format developed by Microsoft.

WMV is designed to handle all types of video content. The files can be highly compressed and can be delivered as a continuous flow of data (on-line radio). WMV files can be of any size, and be compressed to match many different bandwidths (connection speeds).

The WMV format is similar to the ASF format (see above)

Other Windows Media Formats

WAX (Windows Media Audio Redirector) files are much the same as ASX files, but intended to describe audio files (.wma files)

WMP (Windows Media Player) files and WMX are reserved file types for future use by Windows.

Introduction to Flash

Flash is a tool for creating interactive and animated Web sites.

What you should already know

Before you continue you should have a basic understanding of the following:

- WWW, HTML and the basics of building Web pages

If you want to study these subjects first, go to our [Home Page](#)

What is Flash?

- Flash is a multimedia graphics program specially for use on the Web
 - Flash enables you to create interactive "movies" on the Web
 - Flash uses vector graphics, which means that the graphics can be scaled to any size without losing clarity/quality
 - Flash does not require programming skills and is easy to learn
-

Flash vs. Animated Images and Java Applets

Animated images and Java applets are often used to create dynamic effects on Web pages.

The advantages of Flash are:

- Flash loads much faster than animated images
 - Flash allows interactivity, animated images do not
 - Flash does not require programming skills, java applets do
-

Who can View Flash?

In September 2000, NPD Research, the parent company of MediaMetrix, conducted a study to determine what percentage of Web browsers have Flash preinstalled. The results show that 96.4% of Web users can experience Macromedia Flash content without having to download and install a player.

If you do not have the Shockwave Player installed you can [download it for free](#) from Adobe's site.

Who can Create Flash Movies?

To create your own Flash movies you need to buy a Flash program.

The latest version from Adobe is Adobe Flash Lite (or Flash CS3 Pro).

If you do not have a Flash program, you can [download a 30 days free](#) trial version of Flash from Adobe.

Where to Start?

After you have installed Flash, you should go through the lessons that are included in the program. Start Adobe Flash, click Help in the menu and choose Lessons. These lessons will teach you the basics of Flash.

Flash in HTML

Flash Embedded in HTML

After creating a Flash movie you choose File > Save As from the top menu to save your movie. Save the file as "Somefilename fla".

To embed the Flash movie you just made into an HTML page, you should go back to your Flash program and do the following steps:

Step 1

Choose File > Open. Open a Flash movie you have created.

Step 2

Choose File > Export Movie.

Step 3

Name the file "somefilename.swf". Choose the location where the file is to be stored (in your Web folder). Click OK.

Step 4

Open the HTML page where you want to insert your Flash movie. Insert this code:

```
<object width="550" height="400">  
<param name="movie" value="somefilename.swf">  
<embed src="somefilename.swf" width="550" height="400">  
</embed>  
</object>
```

Note: This is the minimum code you need to embed a Flash movie in a browser. A broken icon will appear on the Web page if the user does not have the Flash plug-in installed.

Note: In the code above there is both an <embed> tag and an <object> tag. This is because the <object> tag is recognized by Internet Explorer, and Netscape recognizes the <embed> tag and ignores the <object> tag.

Step 5

Type in the address of the HTML file in your browser and look at your first Flash movie.

Let the Flash Program do the Work

The code above is the absolute minimum code to embed Flash movies in HTML pages. It is not recommended to use the minimum code. There should be a few more attributes added:

- `classid` is an attribute to the `<object>` tag. It tells Internet Explorer to load the ActiveX plug-in if it is not installed
- `pluginspage` is an attribute to the `<embed>` tag. It displays a link to the Shockwave download page if Netscape does not have it

The Flash program can add these attributes for you:

Step 1

Choose File > Publish. Flash will now create the `<object>`, `<param>`, and `<embed>` tags for you. It will also create the `classid` and `pluginspage` attributes.

Step 2

Open the HTML document that Flash created, view the HTML source and copy the code into your HTML page where you want your Flash movie.

Step 3

Be sure that you have the "somefilename.swf" in your Web folder.

Step 4

Type in the address of the HTML file in your browser and look at your first Flash movie.

WEEK Fourteen

Introduction to FrontPage

What is FrontPage?

FrontPage is a Microsoft software program for designing Web sites.

It has the familiar look and feel of other **Microsoft Office** products, as well as being tightly integrated with them.

FrontPage also integrates with other **third-party** products.

What's new for 2003?

Microsoft FrontPage incorporates a major upgrade from the previous version, resulting in many **new** design tools, new web site maintenance and administration features.

The major changes in **FrontPage** concentrate in the following areas:

- Design
- Coding
- Extending FrontPage

Changes in Design

New layout tools include

Layout Tables and Cells Task Pane – enables more control of the page layout.

Dynamic Web Templates – enables additional capabilities when working within a design team environment.

Page Ruler and Layout Grid options – options to display page rulers and layout grids.

Image Tracing – enables site development by tracing graphic elements in Design View.

Changes in Coding

New coding tools include

HTML – cleaner code.

Split View – displays both the design and the underlying code on the screen at the same time.

Quick Tag Selector and Editor – enables quick changes to a tag when working in Design View. This can be very useful when using Find and Replace.

Editing Text Files – text files, such as JavaScript, XML, XSLT, etc, can be edited without corrupting the code.

Cleaning Up HTML – for producing cleaner HTML code.

Scripting Tools – enables Jscript and VBScript authoring.

Code View – now supports word wrap, line numbering, matching tags, and indentation, as well as the ability to store blocks of code for later site integration.

Extending FrontPage

With the new integration of **FrontPage** with **Windows SharePoint Services**, the web hosting service no longer has to support **FrontPage Extensions**.

The new **Review Site** view enables you to see the contents of the remote site to compare with the local copy of the site.

The FrontPage Interface

HTML Basics

An HTML overview

Hypertext Markup Language (HTML) is a universal formatting language used by web site developers to create web pages.

FrontPage generates an easier environment to develop web pages based on **HTML Tables** as well as adding more interactivity to your site.

Naming files

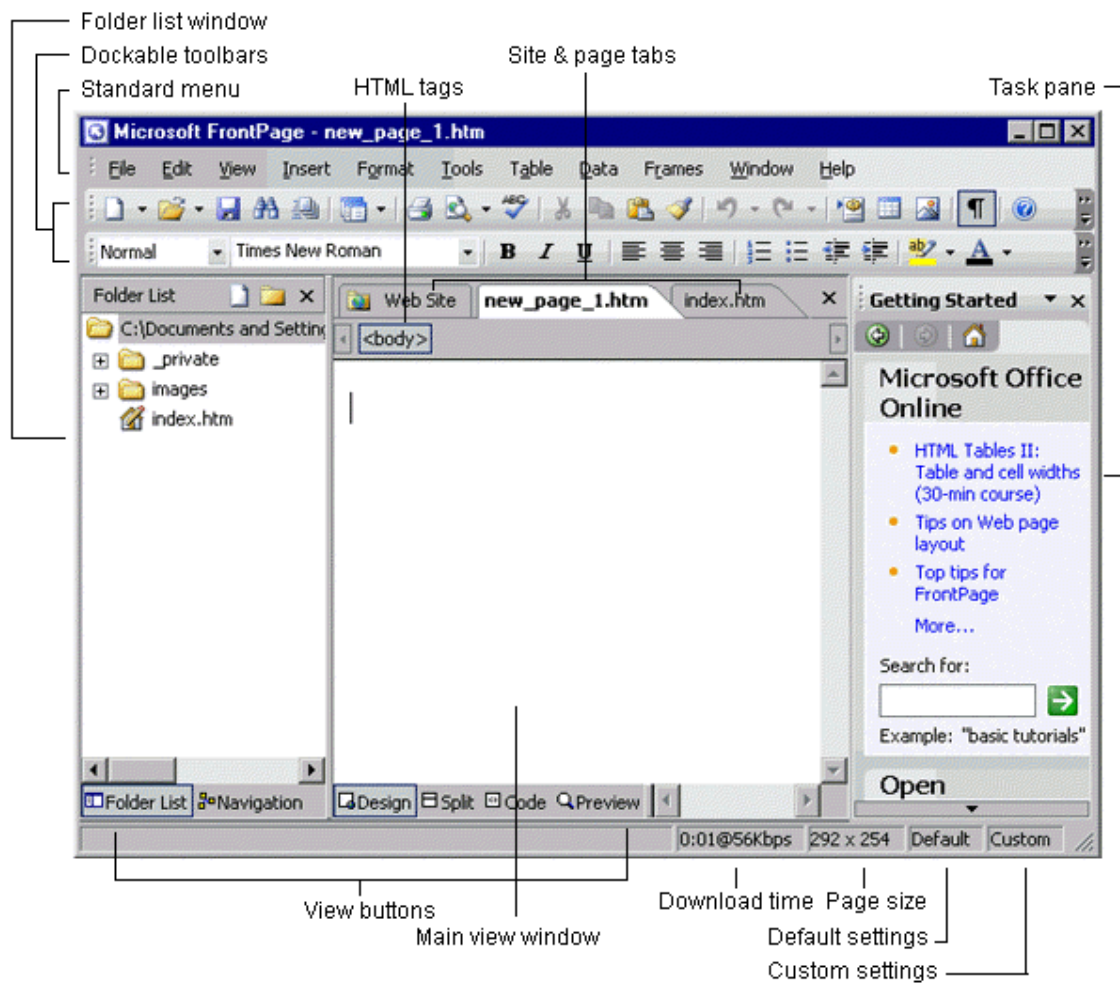
HTML files are text files which can be delivered using the extensions:

.htm (8.3 format)

OR

.html

The **index.htm** (or **.html** file) is the mandatory file in a web site and is automatically named in FrontPage. It is usually referred to as the “homepage” or “splash” page. This is the page which opens when the user enters a domain name or URL into their browser.



The basic **workspace** displays:

Standard or main menu – the familiar style menu used in Microsoft Office products.

Dock able toolbars - toolbars which can be “docked”, or “free-floating”, which moved around the screen.

Folder List pane - displays the Web site folders and files in a directory tree.

Main view window - page or site related views appear in this space.

Site and page tabs - to toggle between various pages in the Web site.

Task panes - contains tools which are automatically made available when certain tasks are performed.

View buttons - change the workspace view by clicking buttons located at the bottom of the interface.

Download Time - specifies the time it will take for the current page to download at a specific modem speed.

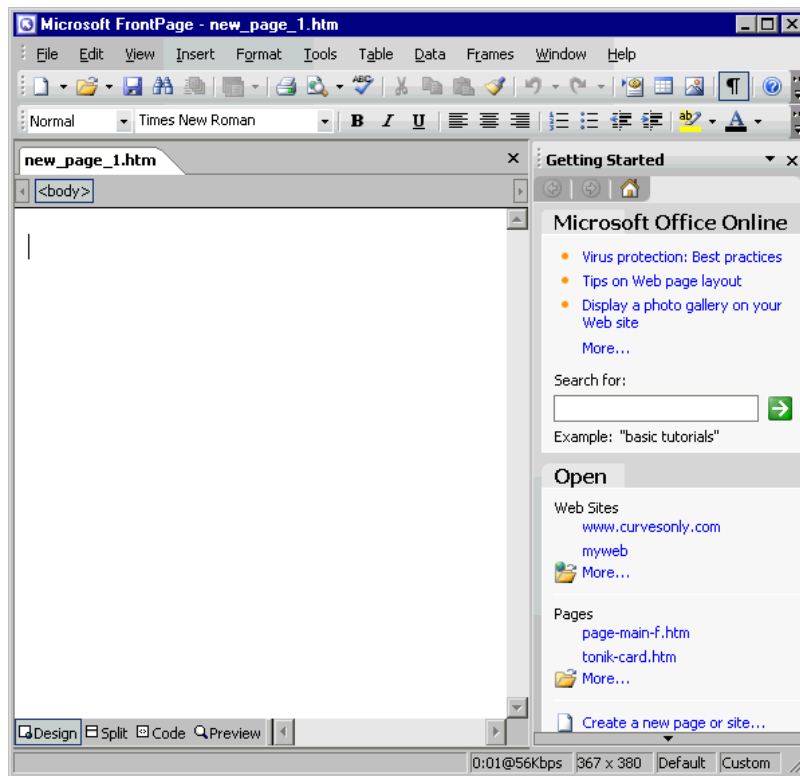
Page size - the dimensions of the page in pixels.

Default and Custom - to access the page Properties dialog box.

Creating New Pages

Opening a New Page

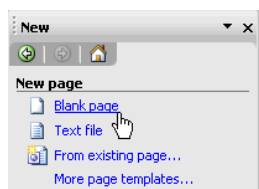
When you open a new blank page, it appears in the **Page** view:



Adding new pages

From the main menu, choose **File > New**. The **New Task** pane opens on the right side of the screen

From the **New task** pane choose **Blank page**:



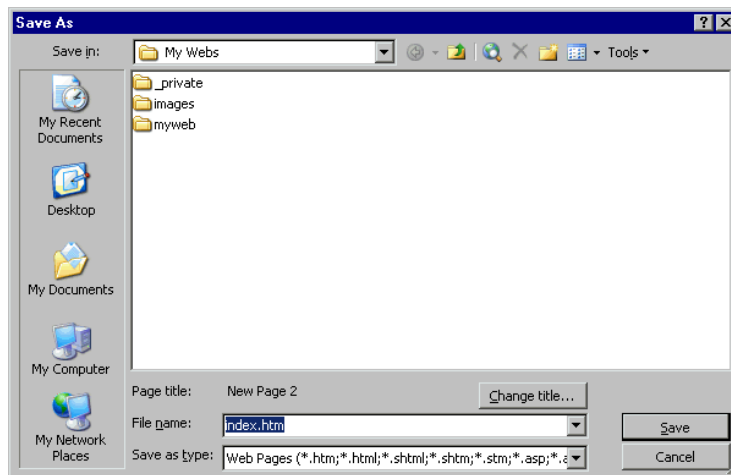
Saving pages

From the main menu, choose **File > Save**

OR press the **Ctrl + S** key combination.

If you previously saved the file, it is updated in its current location under its current name.

If this is your first time saving the page, the **Save As** dialog box opens:



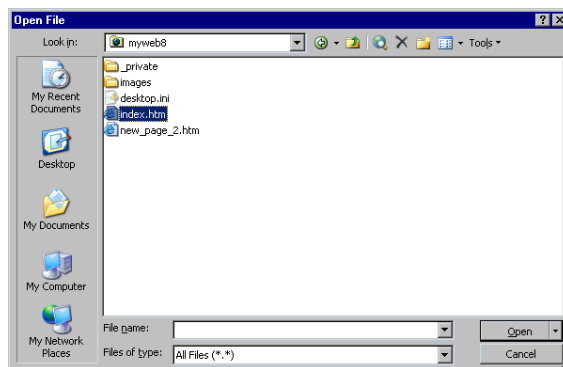
Type a **file name** for the page and ensure the **Save in** location is correct. Select the appropriate file type from the **Save as type** drop-down menu. While creating your web page, you should save it as **Web Pages**.

Opening pages

From the main menu, choose **File > Open**

OR press the **Ctrl + O** key combination.

The **Open File** dialog box opens:



Locate and select the desired file, and click **Open**.

Note: FrontPage has removed the **Navigation toolbar** from the application. There are now **31 new Task panes** which have been added to FrontPage.

Using Page templates

Page templates are pre-formatted web pages provided with **FrontPage**.

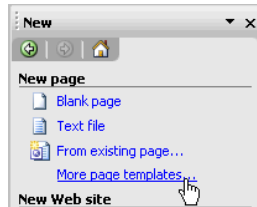
Depending on the type of information you want to present, you can select the **template** you need and use it as a basis for **your own page**.

To use a template

From the main menu, choose **File > New**.

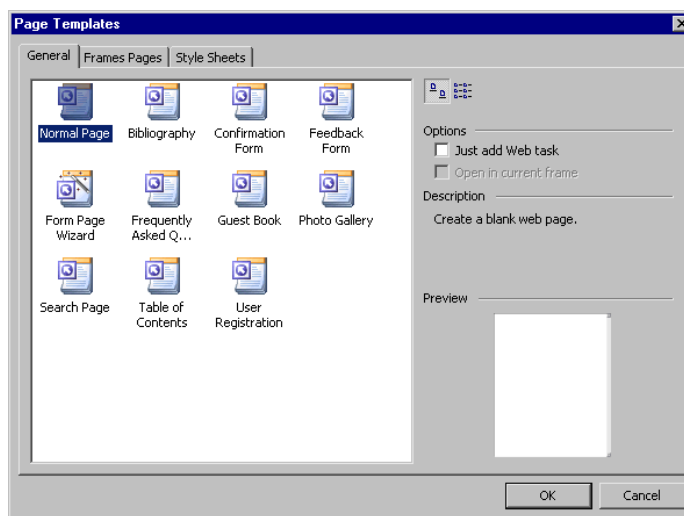
The **New Task** pane opens on the right side of the screen.

Choose **More page templates**, and the **Page Templates** dialog box appears:



In the **Page Templates** dialog box, select a template.

Click **OK**:

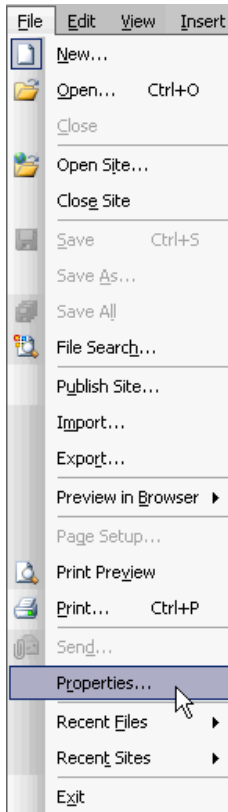


The **template** page opens. You can now enter the content of your web page into the format provided.

Page Properties

Using Page Properties dialog box

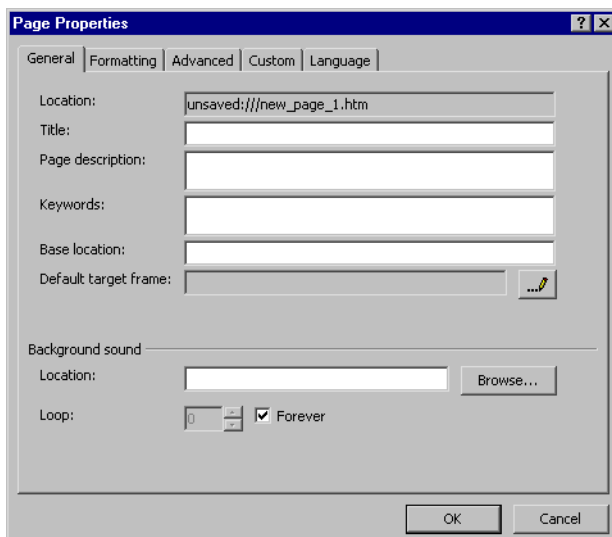
The **Page Properties** dialog box allows you to configure your web page. It consists of tabs where properties such as background and margin settings can be specified. From the main menu, choose **File > Properties**:



Using the General tab

The **General** tab allows you to specify the location (**URL**) and title of your page.

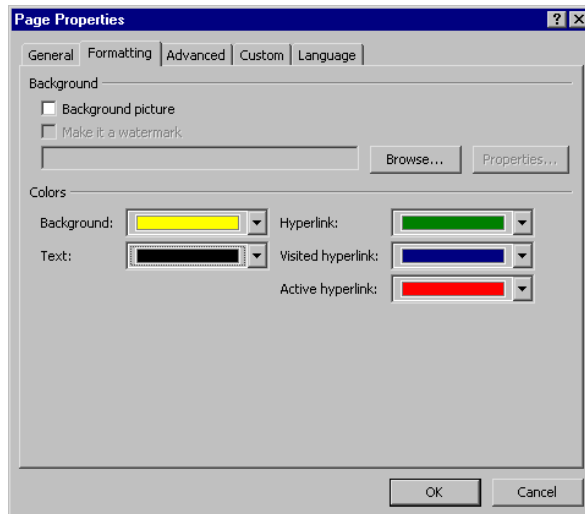
Sound files can also be included to be played when the page opens (if such files are supported by the browser):



Using the Formatting tab

The **Formatting** tab allows you to configure **Background** and **Colors** options.

You can specify a **picture** to be used as the **page's background**, create effects when users move the mouse pointer over hyperlinks on your page, as well as set the colours for the **background**, **text** and **hyperlinks**:

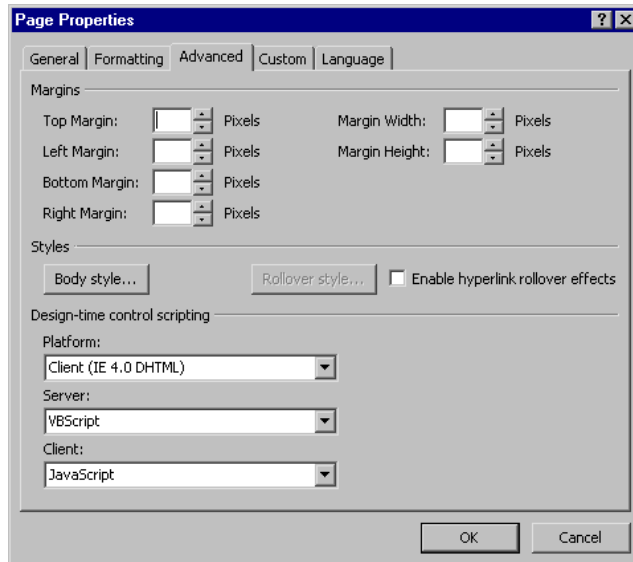


Using the Advanced tab

The **Advanced** tab allows you to specify **Top**, **Left**, **Bottom**, and **Right** margins for your page.

Additionally, **Body style** and **Rollover style** can be specified.

The **Advanced** tab can also enable you to do **Design-time control scripting**:



Creating a New Web File

Using Web site templates

Web site templates are pre-formatted web sites, complete with themed pages and a link structure.

The **templates**, provided by **FrontPage**, are designed to suit different contexts.

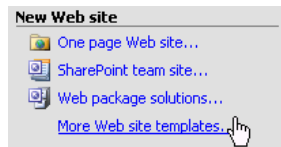
Depending on the type of web site you want to create, you can select the style you need and use it as a basis for your own site.

To open a web site template

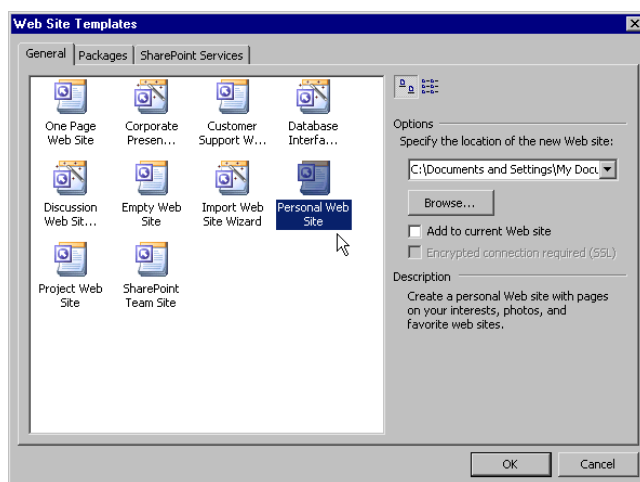
From the main menu, choose **File > New**.

The **New Task** pane opens on the right side of the screen.

Choose **More Web site templates**, and the **Web Site Templates** dialog box appears:



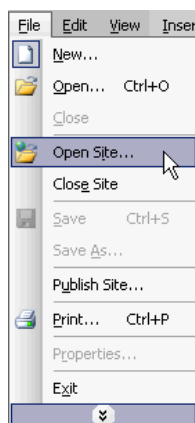
In the **Web Site Templates** dialog box, select a template and click **OK**:



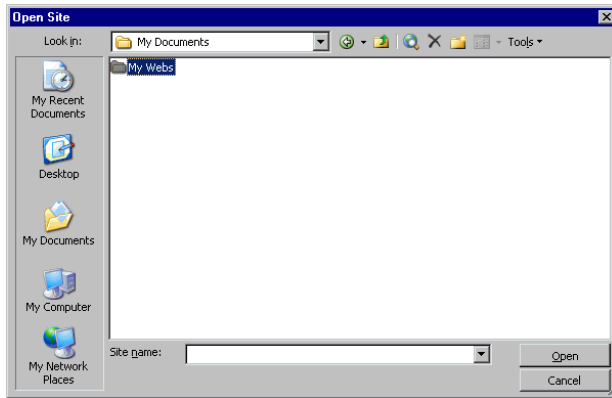
You can now enter the **content** for your website into the structure provided.

Opening a web site

From the main menu, choose **File > Open Site**:



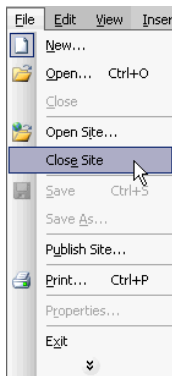
The **Open Site** dialog box opens:



Locate and select the **desired file**.
Click **Open**.

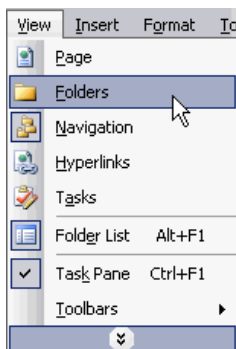
Closing a web site

From the main menu, choose **File > Close Site**:



Deleting a web site

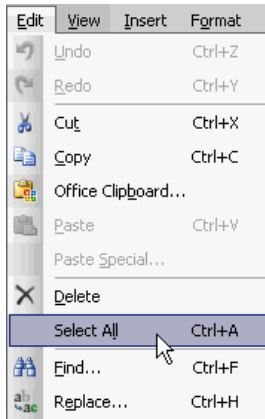
Open the web site you want to delete. From the main menu, choose **View > Folders**:



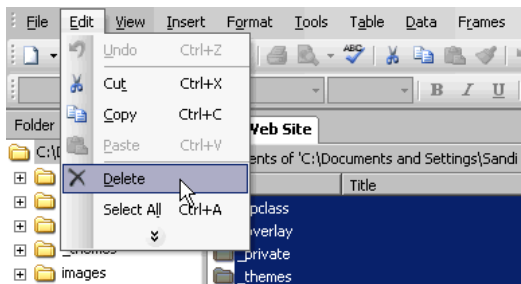
Note: Before you **delete** a web site, be sure that you have no more need for it. Once deleted, there is **no undo**.

From the main menu, choose **Edit > Select All**

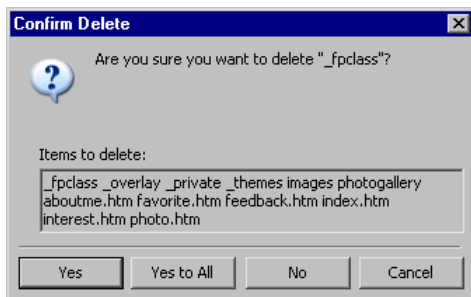
OR press the **Ctrl + A** key combination:



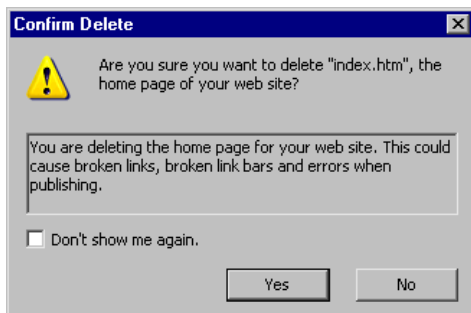
From the main menu, choose **Edit > Delete**:



A **Confirm Delete** dialog box will open, asking if you would like to delete one item or all, click the **Yes to All** button:



A second **Confirm Delete** dialog box will appear. Click **Yes** to confirm your selection:



Building a Web Site

Using a Wizard or Template

Creating a New Web Site

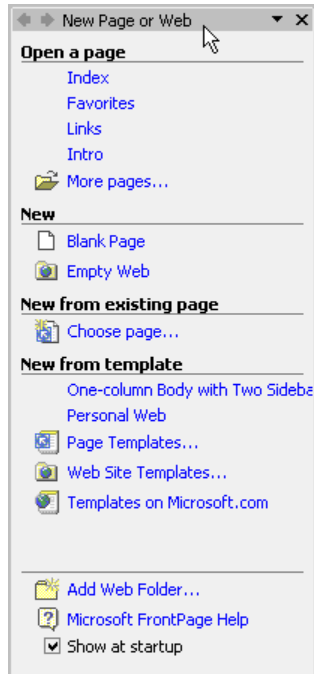
- **When a web page is created onto a Web server**, it is accessible to external viewers, via the Internet or an Intranet.
- The computer that the page is created on must have the **Hypertext Transfer Protocol (HTTP)** server software installed on it.
- This allows you to **create and edit** documents on the web from any location, and will remove the steps of having to **download, edit**, and then **upload** the page every time.
- On the other hand, when a web page is created directly onto a **local hard disk or network drive**, it can be published at a later point by using the **Publish Web** feature.

Note: This procedure will not allow you to test certain features.

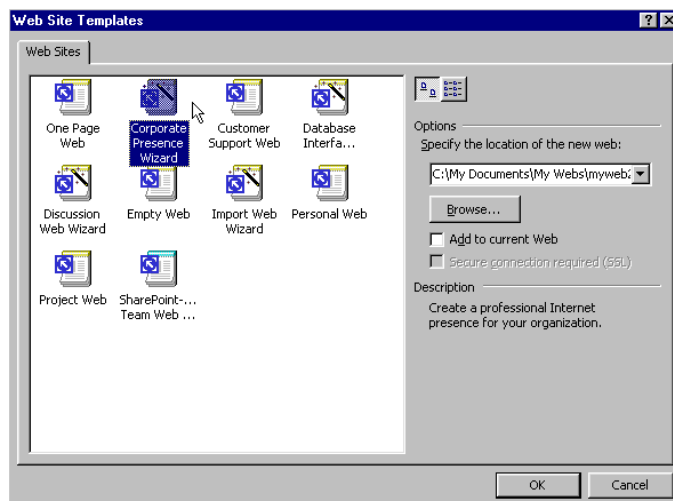
- This is a **very good** way to create a web page for a **first time user**.
- When a **web site** is created, the location of the files for the web that are saved must be specified, **either to a local hard disk, network drive, or a Web server**.
- If saving to a **Web server**, this will be a web site address that viewers can enter into their browsers to view the page.
- If saving to a local web for testing, and only if Web server software like **Microsoft PWS** is running on the computer, specify the server address as <http://localhost> or **http://127.0.0.1**. The web site will be functional immediately when web creation begins.
- If saving to a local web other than Microsoft PWS, the **FrontPage Server Extensions** will have to be installed on the computer being used for the web site creation.
- If saving to a **Web hosting service**, the service will provide the server address that the web will be designed on. Most hosting services will provide the **FrontPage server extensions** with the hosting package.
- If using a Web server not supported by **FrontPage server extensions**, or **saving to a hard disk**, the web pages can be exported to the Web server with the **Publish** feature.

Creating a Web Site with a Wizard

- From the main menu, choose **File > New > Page or Web** to open the **New Page or Web** panel:



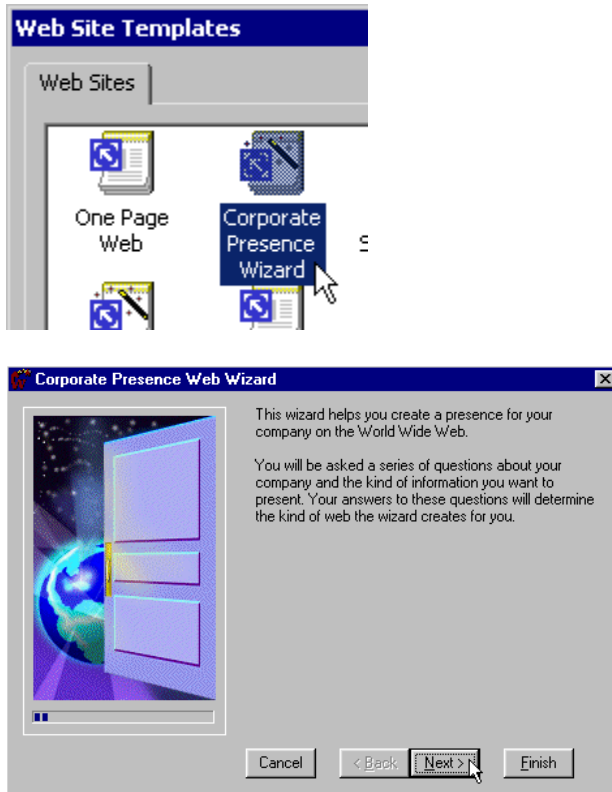
- Select **Web Site Templates** to open the **Web Site Templates** dialog box:



There are four web site Wizards available to use:

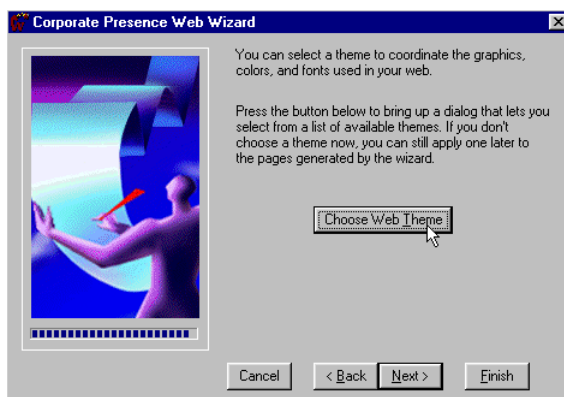
- The **Corporate Presence Wizard**.
- The **Discussion Web Wizard**.
- The **Database Interface Wizard**
- The **Import Web Wizard**.

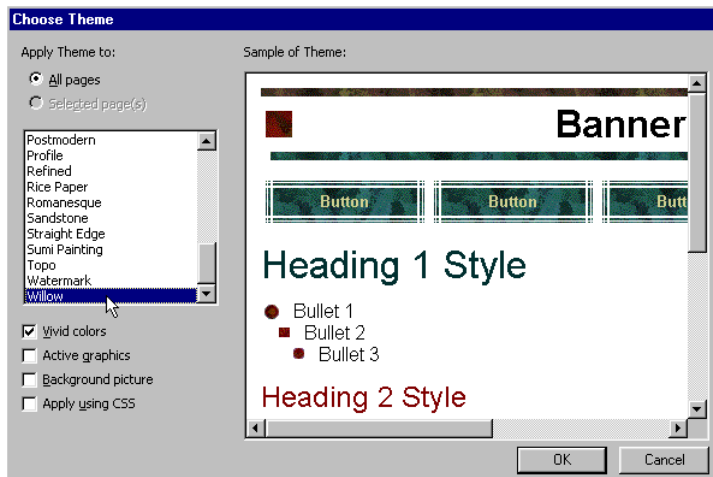
- The **Corporate Presence Wizard** allows you to create a full web site, generally for a smaller company:



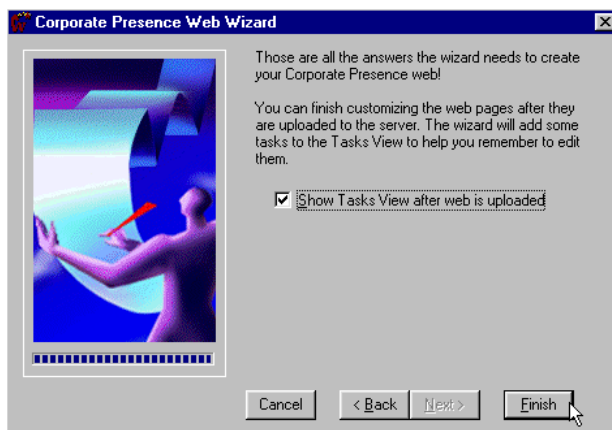
This wizard moves through a series of steps which include:

- Selecting what appears on the **opening web site page**.
- Choosing the **contact information**.
- Selecting a **theme** for the site from an existing list:





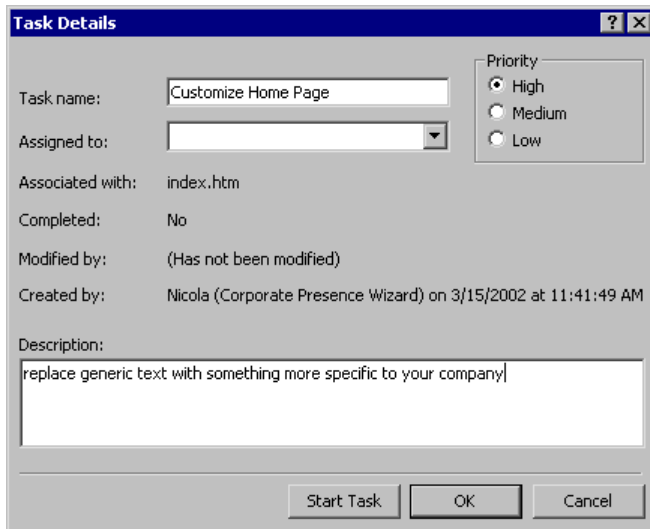
- Once all steps are completed, the final wizard will allow you to launch the **Tasks** view in a new window for the newly created web site, allowing you to view the total pages to be edited:



Status	Task	A..	Priority	Associated With	Modified Date	Description
Not Started	Customize Home Page	M..	High	Home	2/20/02 1:2...	replace generic text with something ...
Not Started	Customize News Page	M..	High	ABC Inc. News Page	2/20/02 1:2...	add your own public relations text
Not Started	Customize Products P...	M..	High	ABC Inc. Products Page	2/20/02 1:2...	create data sheets for your own pro...
Not Started	Customize Services P...	M..	High	ABC Inc. Services Page	2/20/02 1:2...	describe your service offerings
Not Started	Customize Feedback ...	M..	Medium	ABC Inc. Feedback Page	2/20/02 1:2...	adjust input areas in the form
Not Started	Customize TOC Page	M..	Medium	ABC Inc. Table of Cont...	2/20/02 1:2...	describe sections in more detail
Not Started	Customize Search Page	M..	Medium	ABC Inc. Search Page	2/20/02 1:2...	explain how to search for common t...

- Double click on a task to open the **Task Details** dialog box:

Status	Task
Not Started	Customize Home Page
Not Started	Customize News Page



Task Details

Task name:

Assigned to:

Associated with: index.htm

Completed: No

Modified by: (Has not been modified)

Created by: Nicola (Corporate Presence Wizard) on 3/15/2002 at 11:41:49 AM

Priority:

☒ High

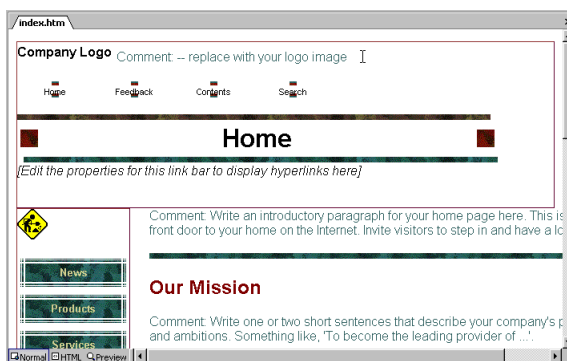
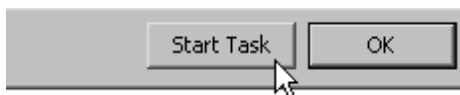
☐ Medium

☐ Low

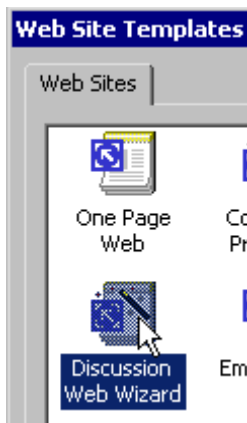
Description:

Start Task OK Cancel

- Click the **Start Task** button to open the page for editing:



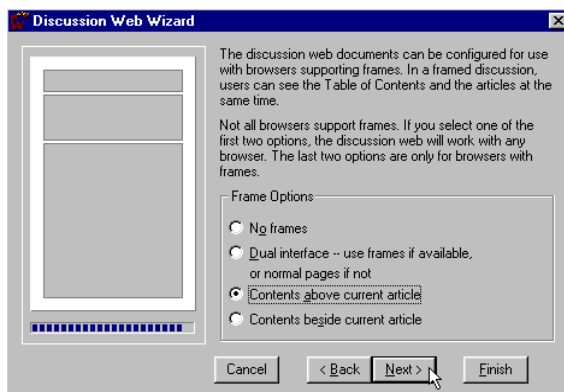
- The **Discussion Web Wizard** allows you to create an area for viewers to engage in a forum on various topics:



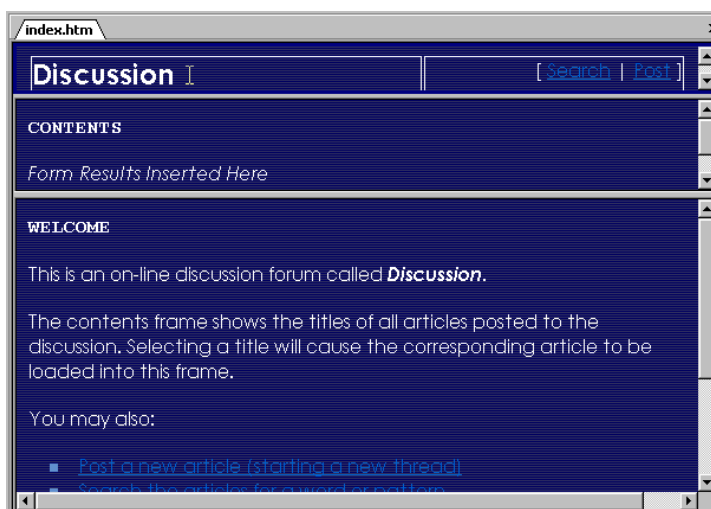


This wizard moves through a series steps which include:

- Selecting the **theme** for the web pages.
- Choosing how **information** is listed.
- Using **frames** within the documents:



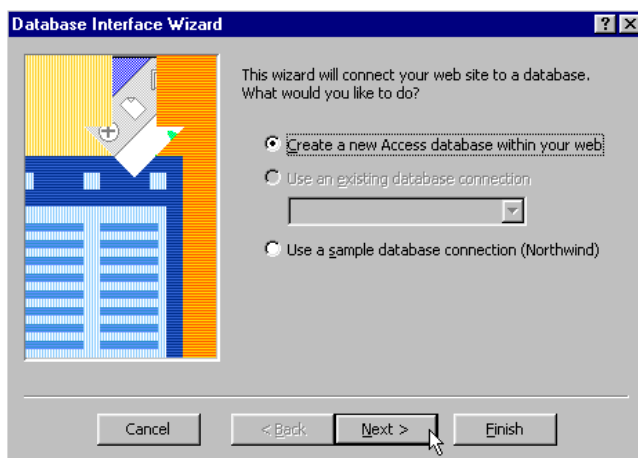
- Once done, click the **Finish** button, and the discussion page will open in a new window, allowing you to begin editing:



- The **Database Interface Wizard** allows you to connect to an outside database so viewers of the Web site can do searches that are answered from the database:

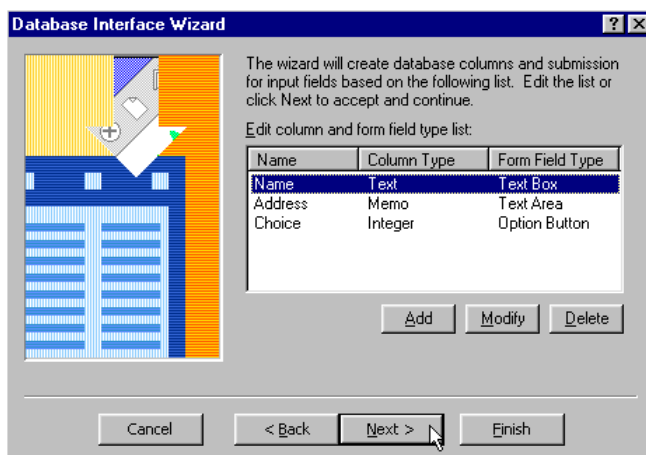


Note: This Wizard needs the **FrontPage server extensions** installed in order to work:

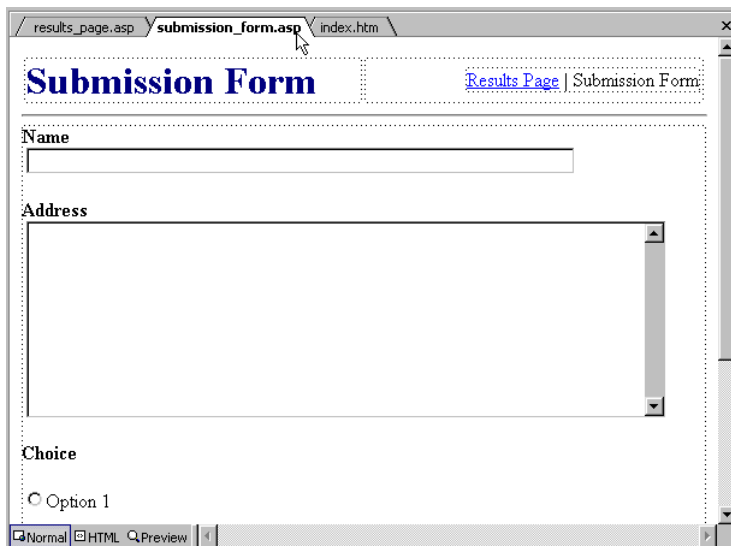
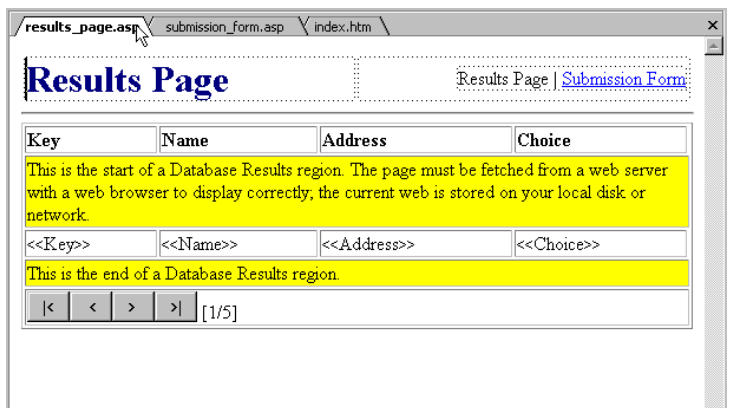
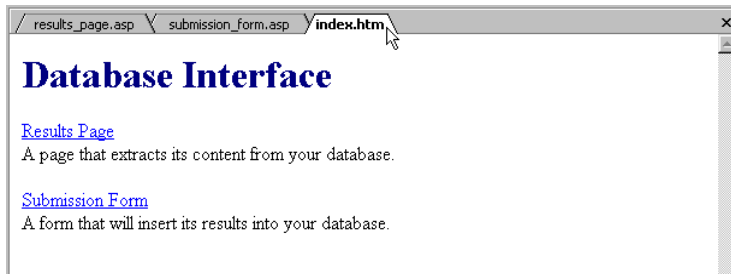


This wizard moves through a series steps which include:

- Setting the **database name**.
- Selecting the **main fields** to be created.
- Setting **which pages** to create:



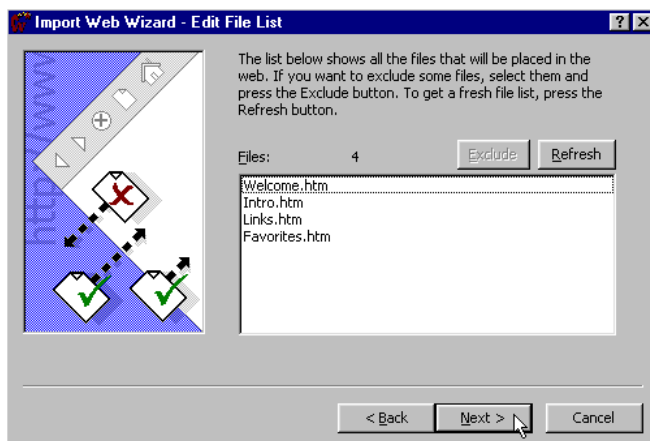
- Once the database wizard is done, click **Finish** and the database will open in a new window, showing the database index page, and any other pages selected to be created:



- The last wizard is the **Import Web Wizard**. It allows you to create a web site containing documents from a **local directory** or an **outside system**:



- Once a source has been selected, click the **Next** button to open the next wizard window:

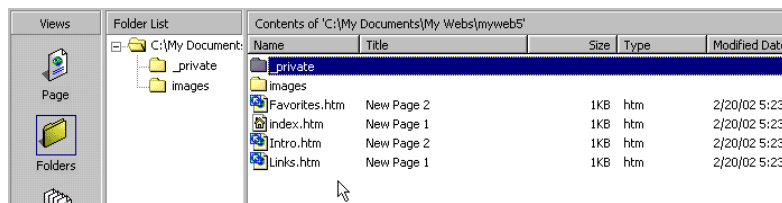


- Select which files to remove from the import, and click the **Next** button to open the last wizard window:



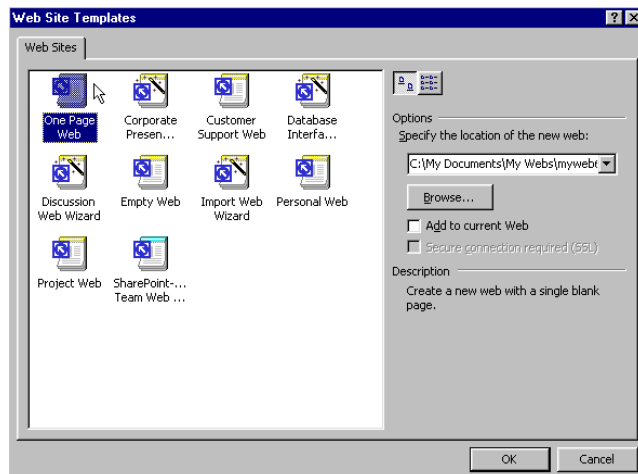
- Click the **Finish** button, and a **new window** will open.

- Click the **Folders** button in the **Views** panel. This will open to show the web files that were imported to the new web site being created:

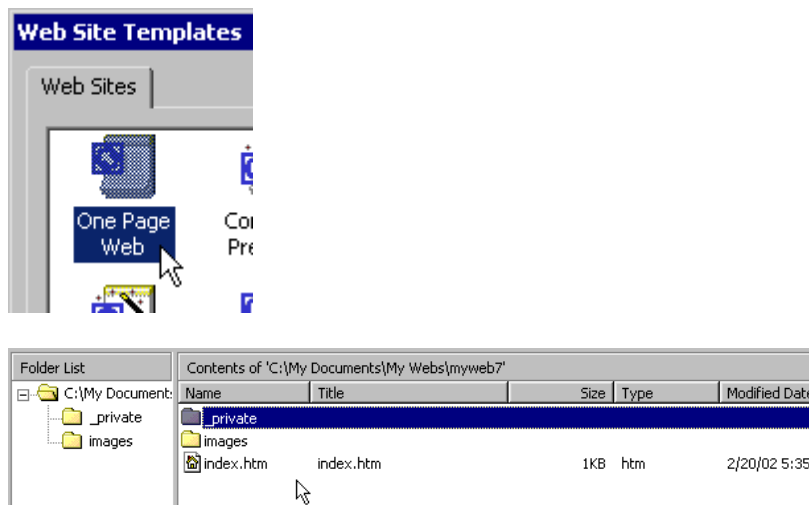


Creating a Web Site using a Template

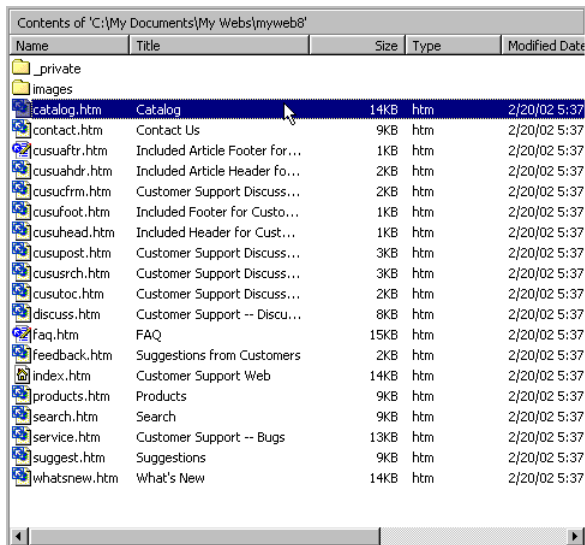
- From the main menu, choose **File > New > Page or Web** to open the **New Page or Web** task pane.
- Select **Web Site Templates** to open the **Web Site Templates** dialog box:



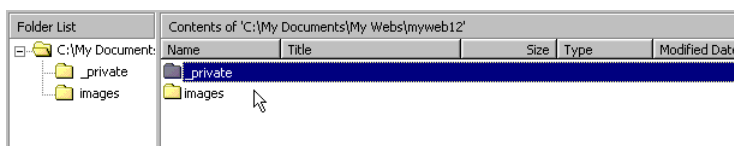
- The **One Page Web** template allows you to create an empty home page, that will open in a new window:



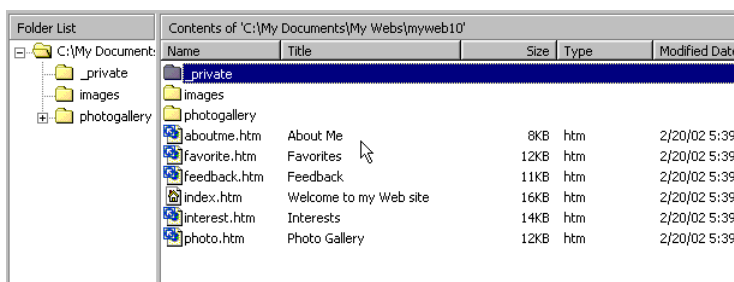
- The **Customer Support Web** template allows you to create a full customer support web site, without having to use a wizard:



- Simply double click on a page to **open** and **edit**.
- The **Empty Web** template allows you to create exactly that, a web site with no content at all. It will only create a directory that all pages, once created, are stored in:



- The **Personal Web** template allows you to create a full personal web site, without having to use a wizard:



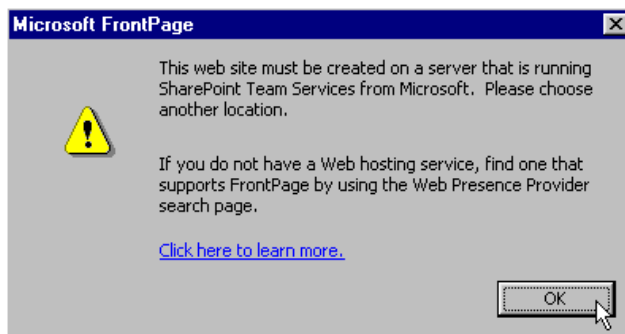
- Simply double click on a page to **open** and **edit**.
- The **Project Web** template allows you to create a way for multiple members of a team to touch base.

- All information that is being created by **each individual of the team** for a project, can be **stored** here as well:

Contents of 'C:\My Documents\My Webs\myweb11'

Name	Title	Size	Type	Modified Date
private				
images				
archive.htm	Archive	14KB	htm	2/20/02 5:40 PM
contact.htm	Contact Us	7KB	htm	2/20/02 5:40 PM
discuss.htm	Discussions	8KB	htm	2/20/02 5:40 PM
index.htm	Home	10KB	htm	2/20/02 5:40 PM
kbaftr.htm	Included Article Footer for...	3KB	htm	2/20/02 5:40 PM
kbahdr.htm	Included Article Header fo...	3KB	htm	2/20/02 5:40 PM
kbcfm.htm	Knowledge Base Confirma...	4KB	htm	2/20/02 5:40 PM
kbfoot.htm	Included Footer for Knowl...	3KB	htm	2/20/02 5:40 PM
kbhead.htm	Included Header for Knowl...	3KB	htm	2/20/02 5:40 PM
kbpost.htm	Knowledge Base Submissi...	5KB	htm	2/20/02 5:40 PM
kbscr.htm	Knowledge Base Search F...	4KB	htm	2/20/02 5:40 PM
kbtoc.htm	Knowledge Base TOC	3KB	htm	2/20/02 5:40 PM
members.htm	Members	15KB	htm	2/20/02 5:40 PM
reqdaftr.htm	Included Article Footer for...	3KB	htm	2/20/02 5:40 PM
reqdahdr.htm	Included Article Header fo...	3KB	htm	2/20/02 5:40 PM
reqdcfm.htm	Requirements Discussion ...	4KB	htm	2/20/02 5:40 PM
reqdfoot.htm	Included Footer for Requi...	3KB	htm	2/20/02 5:40 PM
reqdahdr.htm	Included Header for Requi...	3KB	htm	2/20/02 5:40 PM
reqdpost.htm	Requirements Discussion ...	4KB	htm	2/20/02 5:40 PM
reqdsch.htm	Requirements Discussion ...	4KB	htm	2/20/02 5:40 PM
reqdtoc.htm	Requirements Discussion ...	4KB	htm	2/20/02 5:40 PM
schedule.htm	Schedule	11KB	htm	2/20/02 5:40 PM
search.htm	Search	7KB	htm	2/20/02 5:40 PM

- The main use for this template is for an Intranet.
- The **SharePoint-based Team Web Site** template allows you to create a more intense collaborative site that has a shared document area, contact and task lists, calendar, and more.
- This template can only be created on a server that has **SharePoint (Office Web)** server extensions installed. Otherwise, a warning dialog box will open:

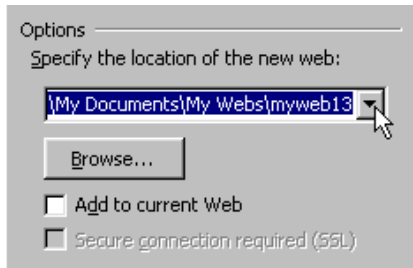


Web Location Options

Using the Web Location Options area

- The **Web Site Templates** dialog box allows you to select the appropriate location that the new site will be hosted from.
- From the main menu, choose **File > New > Page or Web** to open the **New Page or Web** task pane.
- Select **Web Site Templates** to open the **Web Site Templates** dialog box.

- In the **Options** area, select the directory to where the new web site will be stored:



- Typically, FrontPage will automatically offer a storage directory, a name for the new web site, as well as a home page called **index.htm** within the root index of the newly created web site.
- If the **Add to current Web** checkbox is selected, the newly created web site, along with its new home page **index.htm**, will be added to the currently open site.
- If the **new site** also has a home page named **index.htm**, one of the pages will be deleted, so it is best to leave the **Add to current Web** checkbox **unselected** as to not lose any information.

Choosing Web Hosting Service Providers

- There are a variety of **web hosting providers**, and with that, they need to be compared to know which will provide the needed requirements on a per-user basis.
- **Pricing** - Look at this closely, as some plans can include site design, high-speed connection and more.
- **Disk Space** - The **Disk space** tells you how much space you will need with all pages and any multimedia files contained in the web site. To get a better idea of how much space will be required, create a disk-based web site closely matched to what the final product will be. The space taken on the disk will tell you the space you will need from the provider.
- **Connection Speed** – This can be quite different between providers, but the best to use are **T-1** and **T-3**.

Note: Keep in mind that the speed offered is from the hosting provider to the Internet, not from the computer that the web site was created on to the hosting provider.

- **Traffic** - Watch the traffic that the new web site receives. If more traffic occurs than originally specified to the web hosting provider, extra charges will apply.
- **FrontPage Server Extensions** - Make sure that the web hosting provider will include the **FrontPage server extensions**, and not charge separately for them, as well as have the most up-to-date version of the extensions.
- **Domain Name Hosting** - Some service providers can provide better pricing on domain names, rather than going through the older web name registries.
- **Email Accounts/Mailing Lists** – Look into how many email accounts are provided with the hosting, most will include at least 1 account, depending on the disk space being taken by the site.

- **FTP** - The **Publish** feature in FrontPage is what is used to upload the web page file onto the Internet, so confirm that at least one FTP account is included from the hosting provider.
- **Server Security** - This is an important feature to have if the new site requires portions to be placed within a secure server for transmitting any data. Confirm that the provider can offer this.

- **E-commerce** - If creating a web site offering products for online purchasing, hosting providers will typically have this capability, but for a higher package price. The hosting provider can also aid in creating a merchant account, for accepting credit cards online.
- **CGI - CGI (Common Gateway Interface)** is the main scripting technology behind web programming. If the hosting provider is on Unix servers, it can provide a CGI script directory, but if it is on NT servers, they might not be available.
- **Access and SQL** - The hosting provider must be able to support database serving if the web site is database-driven. Most providers will provide for both Access and SQL databases, but it is important to check.
- **Statistics** - Most hosting providers can provide information on the number of times the site is viewed, who is viewing the site, and the click rate on any banners ads or downloads on the site. If budgeting and marketing campaigns will be based from the web site, make sure that the correct information and reports can be tracked and reported on.
- **Backups** - It is very important to make sure that the hosting provider being used is backing up web sites on a regular basis. This will cut down on any frustration for the viewer if information is not updated on time. It is also good for keeping accurate information captures for reports.

Choosing an ISP

- An **ISP (Internet Service Provider)** is mainly concerned with the user that is connected to the Internet, not the web that is being hosted on the Internet. However, using an ISP provider allows you to use only one provider for all your needs, and can be a better solution for smaller companies with smaller sites.

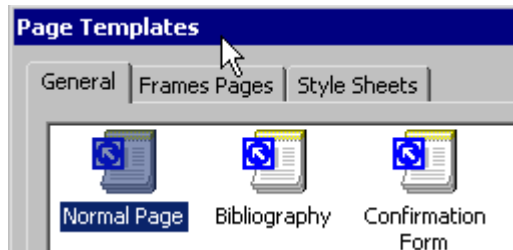
Using a Personal Web Server

- When a web site is created as a disk-based Web on a hard-drive, testing it in a minimal fashion without having to have an outside hosting provider can be done with a personal web server.
- If using a **NT4** or **Windows 2000** environment, access to the **Internet Information Server (IIS)** is probably already provided.
- If using a **NT4** environment without access provided, the **NT4 Option Pack** can install a version of **IIS**, located on the Microsoft web site.
- If using a **Windows 95** or **Windows 98** environment, the **NT4 Option Pack** can install a version of **IIS**, located on the Microsoft web site.
- If using a **Windows 98** or **Windows 98 Second Edition** environment, the **Windows Installation CD** can install a version of **IIS**.

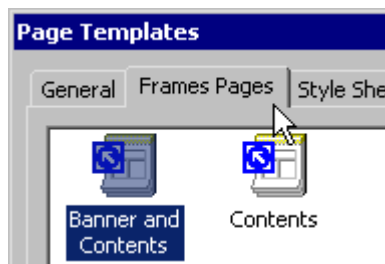
Designing a Web Site

Creating a new Page using a Template

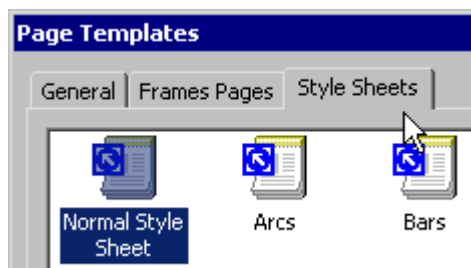
- Individual Web pages can be created through the **Page Templates** dialog box, and formatted with the **Page Properties** dialog box.
- From the main menu, choose **File > New > Page or Web** to open the **New Page or Web** task pane.
- Select **Page Templates** from the **New from Template** area to open the **Page Templates** dialog box:



- Select the **Frames Pages** tab to view various templates that already contain frames:

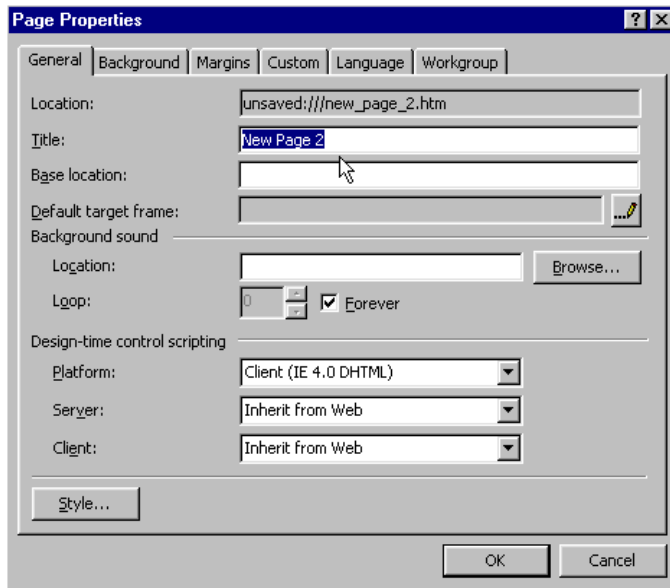


- Select the **Style Sheets** tab to view various templates that already contain a variety of set styles:



- Once a template is selected, it will open in a new window.
- From the main menu, choose **File > Properties** to open the **Page Properties** dialog box

OR right-click anywhere on the page, and from the pop-up menu, select **Page Properties** to open the **Page Properties** dialog box:

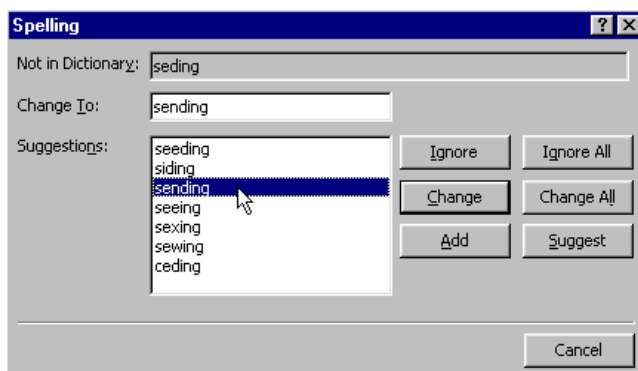


- Use this **dialog box** to alter the **name of the page**, the **location**, the **color**, and any other required changes.
- Save **the page** once any change is done.

Note: To view the **Background** tab within the Page Properties the web will have to be closed. From the main menu choose **File > Close Web** to close the web settings and to disable any themes or style sheets that may be in place.

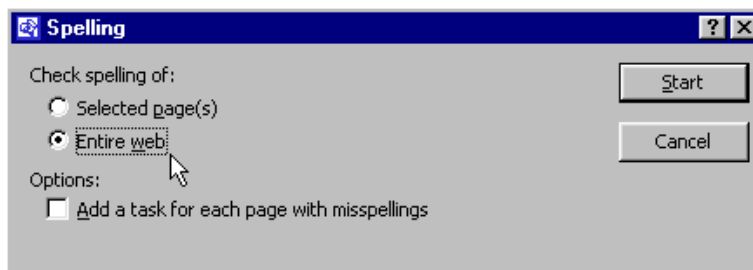
Checking Spelling

- A simple spell check on individual pages can be done. From the main menu, choose **Tools > Spelling** to open the **Spelling** dialog box:

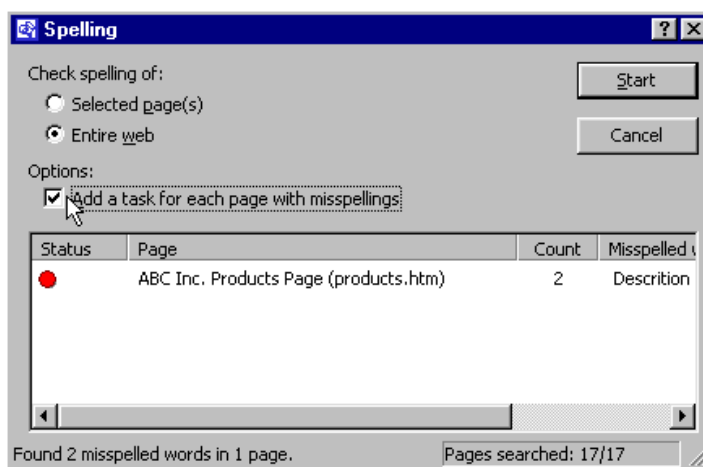


- Spelling should be checked across **multiple pages** of an entire web site, to make sure that all spelling and choices are the same.
- Open the web site in any view except for **Page**.

- From the main menu, choose **Tools > Spelling** to open the **Spelling** dialog box:



- Select **Entire web** under the **Check spelling of** area.
- Click **Start**:

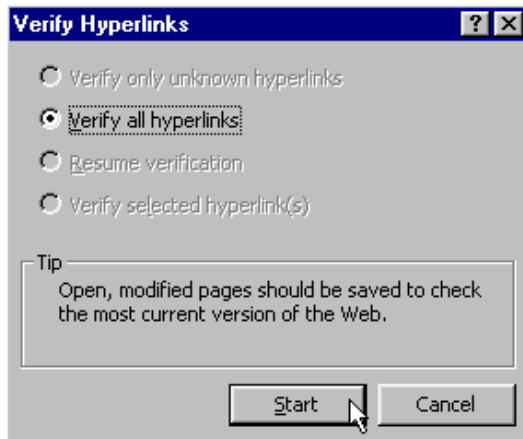


- This allows you to do a Global spell check of all pages within the web site created. Select the **Add a task for each page with misspellings** checkbox to remind you or to assign the task to another person to correct the misspellings.
- Double click on the file listed in the dialog box to open the page and correct any misspellings.

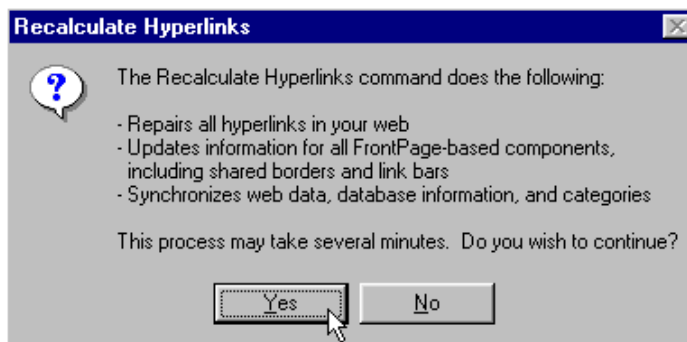
Checking Hyperlinks

- From the main menu, choose **View > Toolbars > Reporting** to open the **Reports** toolbar.
- From the **Reports** toolbar, select the **Verify Hyperlinks** icon:





OR from the main menu, choose **Tools > Recalculate Hyperlinks** to update all views of the web site as well as the multiple items within the web site:



Note: You must be online to use either method.

- The **Verify Hyperlinks** icon allows you to have FrontPage check any external hyperlinks connected to the web site. You can verify hyperlinks on all pages, or for one selected page.
- The result will show in the **Broken Hyperlinks** report, as **Unknown**, **Broken** or **OK**.
- The **Recalculate Hyperlinks** command allows you to update all views of the web site, as well as update any text indexes created if there is a search component in the web site. This command will regenerate all of the **Include** components in the web site, and will update all Web content that is connected to those components.

Setting Tasks

- Open the page in the **Page** view.
- From the main menu, choose **Edit > Tasks > Add Task**

OR from the main menu, choose **File > New > Task** to open the **New Task** dialog box:

- The **Associated with** line shows which page the task will be connected to. When selecting the **Start Task** command, it will automatically open the connected page.
- Enter the required information, and click **OK**.
- If you are in the **Tasks, Folders, Navigation, Hyperlinks** or **Reports** view, select the page that the task needs to be connected to.
- Right click on the selected page, and from the pop-up menu that appears, select **Add Task**.
- This will open the **Add Task** dialog box.
- Enter the required information and click **OK**.

Setting Permissions

- **Permissions** can be set to allow various users access to administer, author or simply browse a web site.
- From the main menu, choose **Tools > Server > Permissions** to open the **Permissions** dialog box.
- **Browse** - Allows users to access the web pages through an internet browser, and view the pages. Alterations of any kind cannot be made by a browser.
- **Author and Browse** - Allows certain users to create and edit any content on the web site, but they cannot add, delete or manage the web site and pages in any way.
- **Administer, Author and Browse** - Allows users to have full access to the web site and all the administration that is connected to the site. This user can add and delete web pages, and set any web permissions and configurations themselves.

Note: Only the **original creator** of the web site can restrict other administrators to specific web sites.

Publishing

Publishing a Site

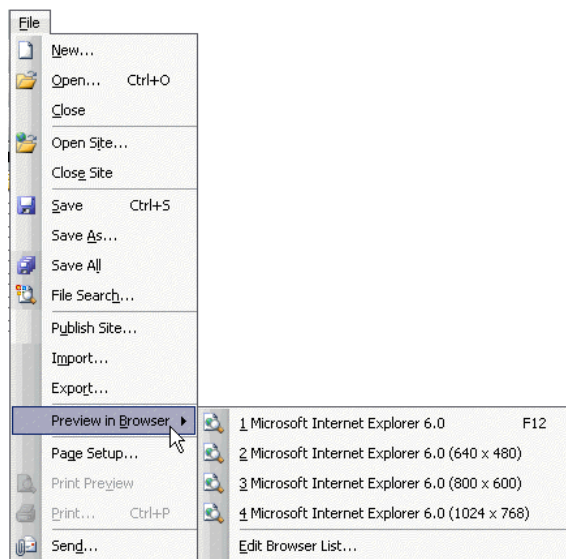
Previewing the web site

When you have completed your web site, you should **preview** it in a **web** browser before publishing it on a server.

To preview a web site

Open the first page of your web.

From the **File** menu, choose **Preview in Browser**:



The **Preview in Browser** expands to a sub-menu which lists a selection of browsers and screen resolutions.

Select the browser for testing the Web site and proceed

OR click the **Preview in Browser** button from the **Standard** toolbar:



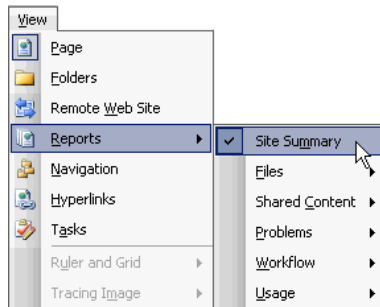
It may take a few moments for the browser to open the web page.

Trouble-Shooting

The **Reports** site summary provides a quick way to check for problems in your web site, so you can **troubleshoot** them before publishing your web to the server.

To view the site summary

From the **View** menu, choose **Reports > Site Summary**:



The **Site Summary** window opens, listing the types of reports available:

Name	Count	Size	Description
All files	1	7KB	All files in the current Web site
Pictures	0	0KB	Picture files in the current Web site (GIF, JPG, BMP, etc.)
Unlinked files	0	0KB	Files in the current Web site that cannot be reached by s..
Linked files	1	7KB	Files in the current Web site that can be reached by start.
Slow pages	0	0KB	Pages in the current Web site exceeding an estimated do.
Older files	0	0KB	Files in the current Web site that have not been modified .
Recently added f...	1	7KB	Files in the current Web site that have been created in th.
Hyperlinks	7		All hyperlinks in the current Web site
Unlinked hyperli...	4		Hyperlinks pointing to unconfirmed target files
Broken links			Click on this item to see the "Hyperlinks" report. Links pointing to unavailable target files
External hyperlinks	4		Hyperlinks pointing to files outside of the current Web site
Internal hyperlinks	3		Hyperlinks pointing to other files within the current Web sit
Component errors	0		Files in the current Web site with components reporting a.
Uncompleted tasks	0		Tasks in the current Web site that are not yet marked co..
Unused themes	0		Themes in the current Web site that are not applied to an.
Style Sheet Links	0		All Style Sheet Links in the current web site.
Dynamic Web Te...	0		All files that are associated with a Dynamic Web Template.

Click on the name of the report you want to view. A **zero** ("0") in the **Count** column next to a report indicates that there is no report available.

Note: From the **View** menu, choosing **Reports > Problems** allows you to choose from a list of problem reports. These same reports are also listed on the **Site Summary**.

Using a server or disk-based web sites

You can generate both server-based and disk-based web sites.

Server-based web sites are delivered through a web server.

Disk-based web sites are HTML files stored locally on your computer and opened in a browser.

Using web servers

Server-based web sites are a collection of interconnected web pages which you can copy onto a **web server** in order to make them accessible to the public via the internet. This is referred to as publishing.

A **web server** is a computer which is configured to host web sites. Most people don't own their own servers. In this case, you can use an Internet Service Provider's (ISP) web hosting service to make your web site available on the Internet.

Before you can publish a web site, you must know if the server supports HTTP publishing. If it does not, you can publish your web site to an FTP server.

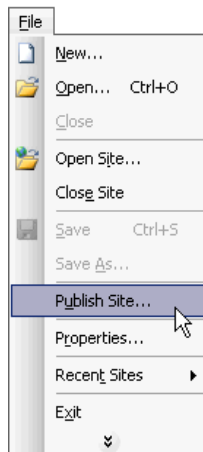
HTTP publishing

The **Hypertext Transfer Protocol (HTTP)** is used to transfer information between computers via the World Wide Web.

In order to publish a web site using HTTP, you must be certain that the server (yours or your Internet Service Provider's) is equipped with **Microsoft FrontPage Server Extensions** or **SharePoint Services**.

To publish to an http server

From the main menu, choose **File > Publish Site**:



The **Remote Web Site** Properties dialog box opens.

Select the **Remote Web Site** tab:

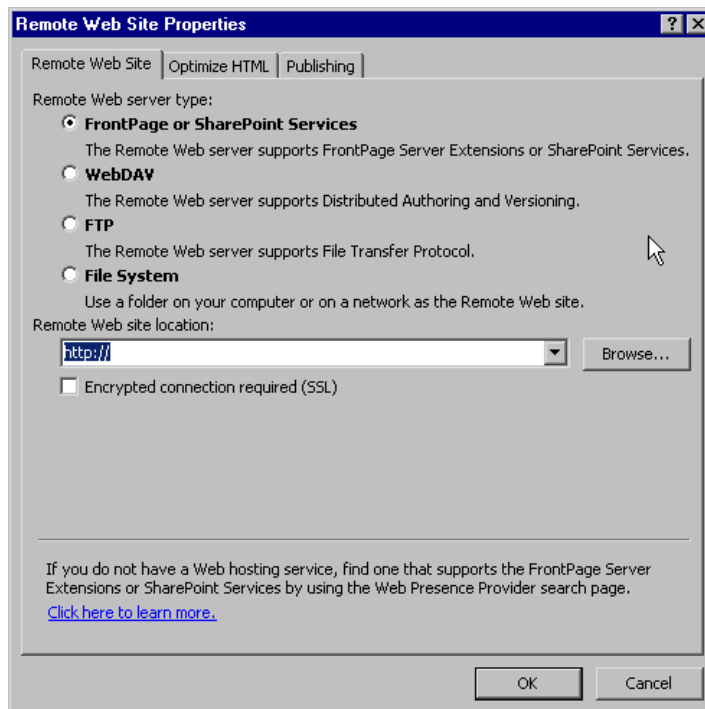
FrontPage or SharePoint Services – the web server must support either FrontPage Server Extensions or SharePoint Services.

WebDAV – the server supports Distributed Authoring and Versioning.

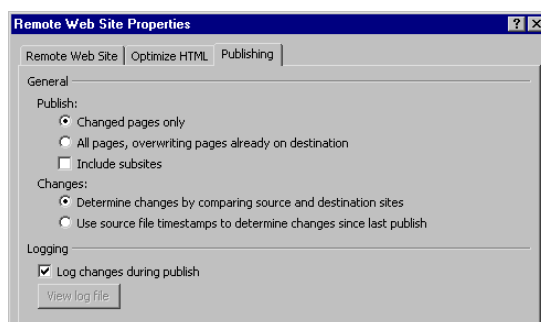
FTP – the server supports File Transfer Protocol.

File System – to use a folder on your computer or on a network as the Remote Web site.

Specify the **Remote Web site location**:



Select the **Publishing** tab, and choose the type of items to be published to the remote server. Click **OK**:



FTP publishing

The **File Transfer Protocol (FTP)** is an older, but faster, internet protocol.

Use FTP to publish your web to a server if it does not have the **FrontPage Server** or **SharePoint Team Services** installed.

To publish to an FTP server

Select **FTP** from the **Remote Web server type** selection in the **Remote Web Site Properties** dialog box.

Specify the **Remote Web site location** to which the Web site is to be published.

Specify the pages to be published and then click **OK**

WEEK Fifteen

Introduction to XML

XML was designed to transport and store data.

HTML was designed to display data.

What is XML?

- XML stands for EXtensible Markup Language
 - XML is a markup language much like HTML
 - XML was designed to carry data, not to display data
 - XML tags are not predefined. You must define your own tags
 - XML is designed to be self-descriptive
 - XML is a W3C Recommendation
-

The Difference Between XML and HTML

XML is not a replacement for HTML.

XML and HTML were designed with different goals:

XML was designed to transport and store data, with focus on what data is.

HTML was designed to display data, with focus on how data looks.

HTML is about displaying information, while XML is about carrying information.

XML is Just Plain Text

XML is nothing special. It is just plain text. Software that can handle plain text can also handle XML.

However, XML-aware applications can handle the XML tags specially. The functional meaning of the tags depends on the nature of the application.

With XML You Invent Your Own Tags

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

That is because the XML language has no predefined tags.

The tags used in HTML (and the structure of HTML) are predefined. HTML documents can only use tags defined in the HTML standard (like <p>, <h1>, etc.).

XML allows the author to define his own tags and his own document structure.

XML is Not a Replacement for HTML

XML is a complement to HTML.

It is important to understand that XML is not a replacement for HTML. In most web applications, XML is used to transport data, while HTML is used to format and display the data.

My best description of XML is this:

XML is a software and hardware independent tool for carrying information.

XML is a W3C Recommendation

XML became a W3C Recommendation 10. February 1998.

XML is Everywhere

We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has developed, and how quickly a large number of software vendors have adopted the standard.

XML is now as important for the Web as HTML was to the foundation of the Web.

XML is everywhere. It is the most common tool for data transmissions between all sorts of applications, and is becoming more and more popular in the area of storing and describing information.

How Can XML be Used?

XML is used in many aspects of web development, often to simplify data storage and sharing.

XML Separates Data from HTML

If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes.

With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for layout and display, and be sure that changes in the underlying data will not require any changes to the HTML.

With a few lines of JavaScript, you can read an external XML file and update the data content of your HTML.

You will learn more about this in a later chapter of this tutorial.

XML Simplifies Data Sharing

In the real world, computer systems and databases contain data in incompatible formats.

XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data.

This makes it much easier to create data that different applications can share.

XML Simplifies Data Transport

With XML, data can easily be exchanged between incompatible systems.

One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.

Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

XML Simplifies Platform Changes

Upgrading to new systems (hardware or software platforms), is always very time consuming. Large amounts of data must be converted and incompatible data is often lost.

XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

XML Makes Your Data More Available

Since XML is independent of hardware, software and application, XML can make your data more available and useful.

Different applications can access your data, not only in HTML pages, but also from XML data sources.

With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

XML is Used to Create New Internet Languages

A lot of new Internet languages are created with XML.

Here are some examples:

- XHTML the latest version of HTML
- WSDL for describing available web services
- WAP and WML as markup languages for handheld devices
- RSS languages for news feeds
- RDF and OWL for describing resources and ontology
- SMIL for describing multimedia for the web

If Developers Have Sense

If they DO have sense, future applications will exchange their data in XML.

The future might give us word processors, spreadsheet applications and databases that can read each other's data in a pure text format, without any conversion utilities in between.

We can only pray that all the software vendors will agree.

XML Tree

XML documents form a tree structure that starts at "the root" and branches to "the leaves".

An Example XML Document

XML documents use a self-describing and simple syntax:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set).

The next line describes the **root element** of the document (like saying: "this document is a note"):

```
<note>
```

The next 4 lines describe 4 **child elements** of the root (to, from, heading, and body):

```
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

And finally the last line defines the end of the root element:

```
</note>
```

You can assume, from this example, that the XML document contains a note to Tove from Jani.

Don't you agree that XML is pretty self-descriptive?

XML Documents Form a Tree Structure

XML documents must contain a **root element**. This element is "the parent" of all other elements.

The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.

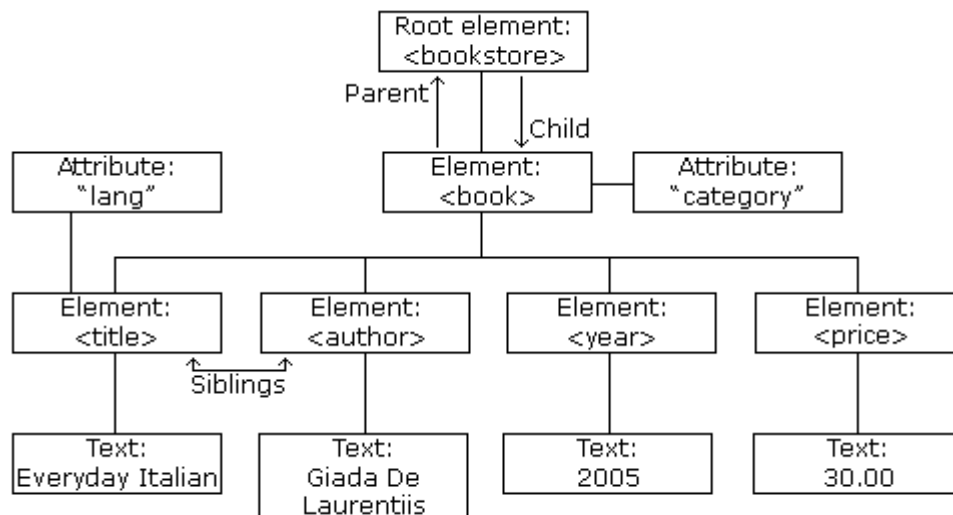
All elements can have sub elements (child elements):

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

The terms parent, child, and sibling are used to describe the relationships between elements. Parent elements have children. Children on the same level are called siblings (brothers or sisters).

All elements can have text content and attributes (just like in HTML).

Example:



The image above represents one book in the XML below:

```
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
```

```

</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>

```

The root element in the example is `<bookstore>`. All `<book>` elements in the document are contained within `<bookstore>`.

The `<book>` element has 4 children: `<title>`, `<author>`, `<year>`, `<price>`.

XML Syntax Rules

The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.

All XML Elements Must Have a Closing Tag

In HTML, you will often see elements that don't have a closing tag:

```

<p>This is a paragraph
<p>This is another paragraph

```

In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:

```

<p>This is a paragraph</p>
<p>This is another paragraph</p>

```

Note: You might have noticed from the previous example that the XML declaration did not have a closing tag. This is not an error. The declaration is not a part of the XML document itself, and it has no closing tag.

XML Tags are Case Sensitive

XML elements are defined using XML tags.

XML tags are case sensitive. With XML, the tag `<Letter>` is different from the tag `<letter>`.

Opening and closing tags must be written with the same case:

```
<Message>This is incorrect</message>
```

```
<message>This is correct</message>
```

Note: "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.

XML Elements Must be Properly Nested

In HTML, you will often see improperly nested elements:

```
<b><i>This text is bold and italic</b></i>
```

In XML, all elements **must** be properly nested within each other:

```
<b><i>This text is bold and italic</i></b>
```

In the example above, "Properly nested" simply means that since the `<i>` element is opened inside the `` element, it must be closed inside the `` element.

XML Documents Must Have a Root Element

XML documents must contain one element that is the **parent** of all other elements. This element is called the **root** element.

```
<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>
```

XML Attribute Values Must be Quoted

XML elements can have attributes in name/value pairs just like in HTML.

In XML the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

```
<note date=12/11/2007>
<to>Tove</to>
<from>Jani</from>
</note>
```

```
<note date="12/11/2007">
<to>Tove</to>
<from>Jani</from>
</note>
```

The error in the first document is that the date attribute in the note element is not quoted.

Entity References

Some characters have a special meaning in XML.

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

```
<message>if salary < 1000 then</message>
```

To avoid this error, replace the "<" character with an **entity reference**:

```
<message>if salary &lt; 1000 then</message>
```

There are 5 predefined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Note: Only the characters "<" and "&" are strictly illegal in XML. The greater than character is legal, but it is a good habit to replace it.

Comments in XML

The syntax for writing comments in XML is similar to that of HTML.

```
<!-- This is a comment -->
```

With XML, White Space is Preserved

HTML reduces multiple white space characters to a single white space:

HTML:	Hello my name is Tove
Output:	Hello my name is Tove.

With XML, the white space in your document is not truncated.

XML Stores New Line as LF

In Windows applications, a new line is normally stored as a pair of characters: carriage return (CR) and line feed (LF). The character pair bears some resemblance to the typewriter actions of setting a new line. In Unix applications, a new line is normally stored as a LF character. Macintosh applications use only a CR character to store a new line.

XML Elements

An XML document contains XML Elements.

What is an XML Element?

An **XML element** is everything from (including) the element's start tag to (including) the element's end tag.

An element can contain other elements, simple text or a mixture of both. Elements can also have attributes.

```
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
```

```

<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>

```

In the example above, <bookstore> and <book> have **element contents**, because they contain other elements. <author> has **text content** because it contains text.

In the example above only <book> has an **attribute** (category="CHILDREN").

XML Naming Rules

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation character
- Names must not start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces

Any name can be used, no words are reserved.

Best Naming Practices

Make names descriptive. Names with an underscore separator are nice: <first_name>, <last_name>.

Names should be short and simple, like this: <book_title> not like this: <the_title_of_the_book>.

Avoid "-" characters. If you name something "first-name," some software may think you want to subtract name from first.

Avoid "." characters. If you name something "first.name," some software may think that "name" is a property of the object "first."

Avoid ":" characters. Colons are reserved to be used for something called namespaces (more later).

XML documents often have a corresponding database. A good practice is to use the naming rules of your database for the elements in the XML documents.

Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software vendor doesn't support them.

XML Elements are Extensible

XML elements can be extended to carry more information.

Look at the following XML example:

```
<note>
<to>Tove</to>
<from>Jani</from>
<body>Don't forget me this weekend!</body>
</note>
```

Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output:

MESSAGE

To: Tove
From: Jani

Don't forget me this weekend!

Imagine that the author of the XML document added some extra information to it:

```
<note>
<date>2008-01-10</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Should the application break or crash?

No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.

One of the beauties of XML, is that it can often be extended without breaking applications.

XML Attributes

XML elements can have attributes in the start tag, just like HTML.

Attributes provide additional information about elements.

XML Attributes

From HTML you will remember this: ``. The "src" attribute provides additional information about the `` element.

In HTML (and in XML) attributes provide additional information about elements:

```

<a href="demo.asp">
```

Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant to the data, but important to the software that wants to manipulate the element:

```
<file type="gif">computer.gif</file>
```

XML Attributes Must be Quoted

Attribute values must always be enclosed in quotes, but either single or double quotes can be used. For a person's sex, the person tag can be written like this:

```
<person sex="female">
```

or like this:

```
<person sex='female'>
```

If the attribute value itself contains double quotes you can use single quotes, like in this example:

```
<gangster name='George "Shotgun" Ziegler'>
```

or you can use character entities:

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

XML Elements vs. Attributes

Take a look at these examples:

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

In the first example sex is an attribute. In the last, sex is an element. Both examples provide the same information.

There are no rules about when to use attributes and when to use elements. Attributes are handy in HTML. In XML my advice is to avoid them. Use elements instead.

My Favorite Way

The following three XML documents contain exactly the same information:

A date attribute is used in the first example:

```
<note date="10/01/2008">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

A date element is used in the second example:

```
<note>
  <date>10/01/2008</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

An expanded date element is used in the third: (THIS IS MY FAVORITE):

```
<note>
<date>
  <day>10</day>
  <month>01</month>
  <year>2008</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Avoid XML Attributes?

Some of the problems with using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

Attributes are difficult to read and maintain. Use elements for data. Use attributes for information that is not relevant to the data.

Don't end up like this:

```
<note day="10" month="01" year="2008"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

XML Attributes for Metadata

Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the ID attribute in HTML. This example demonstrates this:

```
<messages>
  <note id="501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
```



```

</note>
<note id="502">
  <to>Jani</to>
  <from>Tove</from>
  <heading>Re: Reminder</heading>
  <body>I will not</body>
</note>
</messages>

```

The ID above is just an identifier, to identify the different notes. It is not a part of the note itself.

What I'm trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements.

XML Validation

XML with correct syntax is "Well Formed" XML.

XML validated against a DTD is "Valid" XML.

Well Formed XML Documents

A "Well Formed" XML document has correct XML syntax.

The syntax rules were described in the previous chapters:

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must be quoted

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>

```

Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The DOCTYPE declaration in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.

XML DTD

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

```
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to    (#PCDATA)>
  <!ELEMENT from  (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body  (#PCDATA)>
]>
```

If you want to study DTD, you will find our DTD tutorial on our [homepage](#).

XML Schema

W3C supports an XML based alternative to DTD called XML Schema:

```
<xs:element name="note">
<xs:complexType>
  <xs:sequence>
    <xs:element name="to"    type="xs:string"/>
    <xs:element name="from"  type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body"  type="xs:string"/>
  </xs:sequence>
```

```
</xs:complexType>
```

```
</xs:element>
```

If you want to study XML Schema, you will find our Schema tutorial on our [homepage](#).

A General XML Validator

To help you check the syntax of your XML files, we have created an XML validator to syntax-check your XML.

Please see the next chapter.

XML Validator

◀ Previous	Next ▶
------------	--------

Use our XML validator to syntax-check your XML.

XML Errors Will Stop You

Errors in XML documents will stop your XML applications.

The W3C XML specification states that a program should stop processing an XML document if it finds an error. The reason is that XML software should be small, fast, and compatible.

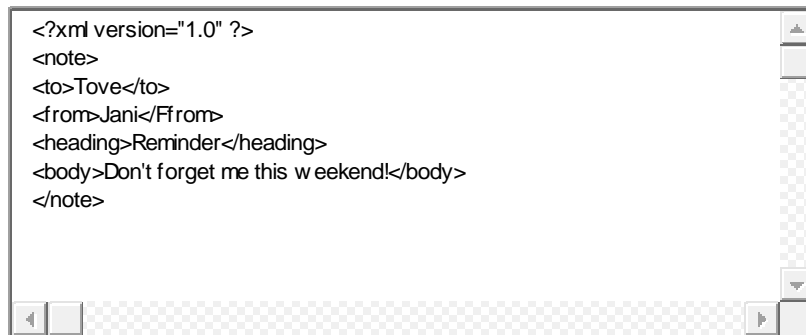
HTML browsers will display documents with errors (like missing end tags). HTML browsers are big and incompatible because they have a lot of unnecessary code to deal with (and display) HTML errors.

With XML, errors are not allowed.

Syntax-Check Your XML

To help you syntax-check your XML, we have created an XML validator.

Paste your XML into the text area below, and syntax-check it by clicking the "Validate" button.



Note: This only checks if your XML is "Well formed". If you want to validate your XML against a DTD, see the last paragraph on this page.

Syntax-Check an XML File

You can syntax-check an XML file by typing the URL of the file into the input field below, and then click the "Validate" button:

Filename:

Note: If you get an "Access denied" error, it's because your browser security does not allow file access across domains.

The file "note_error.xml" demonstrates your browsers error handling. If you want see an error free message, substitute the "note_error.xml" with "cd_catalog.xml".

Validate Your XML Against a DTD

If you know DTD, you can validate your XML in the text area below.

Just add the DOCTYPE declaration to your XML and click the "Validate" button:

```

<?xml version="1.0" ?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to   (#PCDATA)>
  <!ELEMENT from  (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body   (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<message>Don't forget me this weekend!</message>
</note>

```

Note: Only Internet Explorer will actually check your XML against the DTD. Firefox, Mozilla, Netscape, and Opera will not.

Viewing XML Files



Raw XML files can be viewed in all major browsers.

Don't expect XML files to be displayed as HTML pages.

Viewing XML Files

```

<?xml version="1.0" encoding="ISO-8859-1"?>
- <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>

```

Look at this XML file: [note.xml](#)

The XML document will be displayed with color-coded root and child elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure. To view the raw XML source (without the + and - signs), select "View Page Source" or "View Source" from the browser menu.

Note: In Netscape, Opera, and Safari, only the element text will be displayed. To view the raw XML, you must right click the page and select "View Source"

Viewing an Invalid XML File

If an erroneous XML file is opened, the browser will report the error.

Look at this XML file: [note_error.xml](#)

Other XML Examples

Viewing some XML documents will help you get the XML feeling.

[An XML CD catalog](#)

This is a CD collection, stored as XML data.

[An XML plant catalog](#)

This is a plant catalog from a plant shop, stored as XML data.

[A Simple Food Menu](#)

This is a breakfast food menu from a restaurant, stored as XML data.

Why Does XML Display Like This?

XML documents do not carry information about how to display the data.

Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.

Without any information about how to display the data, most browsers will just display the XML document as it is.

In the next chapters, we will take a look at different solutions to the display problem, using CSS, XSLT and JavaScript.

Displaying XML with CSS

◀ Previous	Next ▶
------------	--------

With CSS (Cascading Style Sheets) you can add display information to an XML document.

Displaying your XML Files with CSS?

It is possible to use CSS to format an XML document.

Below is an example of how to use a CSS style sheet to format an XML document:

Take a look at this XML file: [The CD catalog](#)

Then look at this style sheet: [The CSS file](#)

Finally, view: [The CD catalog formatted with the CSS file](#)

Below is a fraction of the XML file. The second line links the XML file to the CSS file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  .
  .
  .
  .
</CATALOG>
```

Formatting XML with CSS is not the most common method.

W3C recommend using XSLT instead. See the next chapter.

Displaying XML with XSLT

◀ Previous	Next ▶
------------	--------

With XSLT you can transform an XML document into HTML.

Displaying XML with XSLT

XSLT is the recommended style sheet language of XML.

XSLT (eXtensible Stylesheet Language Transformations) is far more sophisticated than CSS.

One way to use XSLT is to transform XML into HTML before it is displayed by the browser as demonstrated in these examples:

[View the XML file](#), [the XSLT style sheet](#), and [View the result](#).

Below is a fraction of the XML file. The second line links the XML file to the XSLT file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="simple.xsl"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
      two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
</breakfast_menu>
```

If you want to learn more about XSLT, find our XSLT tutorial on our [homepage](#).

Transforming XML with XSLT on the Server

In the example above, the XSLT transformation is done by the browser, when the browser reads the XML file.

Different browsers may produce different result when transforming XML with XSLT. To reduce this problem the XSLT transformation can be done on the server.

Note that the result of the output is exactly the same, either the transformation is done by the web server or by the web browser.

