

Internet Concepts with Basic Web Design

Course Designer and Acquisition Editor

Centre for Information Technology and Engineering

Manonmaniam Sundaranar University

Tirunelveli

Lecture 1

Introduction to Internet

Objective

This lecture provides an insight into the following:

- ❖ The evolution from small network to Internet
- ❖ The working of the world wide web
- ❖ How do you get connected to the net
- ❖ Domain Name registering
- ❖ How does intranet differ from Internet

Lecture - 1

- 1.1 *Snap Shot*
- 1.2 *What is Internet*
- 1.3 *History of Internet*
- 1.4 *How the Web works?*
- 1.5 *Web Server and Clients*
- 1.6 *Connections as ISP*
- 1.7 *ISDN*
- 1.8 *Dial-up or Leased*
- 1.9 *Domain Naming System*
- 1.10 *Registering our own Domain Name*
- 1.11 *Intranet*
- 1.12 *Short Summary*
- 1.13 *Brain Storm*

1.1 Snap Shot

If there is one technology that caught up literally overnight and has affected more users than any others, it is the World Wide Web (WWW) or Web in short. The Internet is similar to the international telephone system – no one owns or controls the whole thing, but it is connected in a way that makes it work like one big network. The connections to the Internet can be obtained as either for individual users or clients, and connections for servers. And we can register our own Domain name to host our web page on the Internet.

1.2 What is Internet?

Small network to Internet

Two or more computers connected together with the capability of exchanging information is called as Network. The networks that usually take the form of a small office network are called as *Local Area Network (LAN)*. The networks that cover a city or some other large physical distance can be called *Metropolitan Area Network (MAN)* or, more generally, *Wide Area Network (WAN)*.

The most obvious advantage of building networks include

- ◆ Communication (for example, e-mail)
- ◆ Information sharing (for example, database access)
- ◆ Resource sharing (for example, networked printers)

Typically networks are created in the hope of saving money and increasing efficiency. Buying a laser printer for every computer in a large office, for example, would be cost prohibitive. A single laser printer shared across a network can save money and improve productivity.

Moving beyond a small office network to a large network, things get more complicated. Suppose a company has two LANs, one on the first floor of the building and one on the second that need to be joined together. This can be done by drilling a hole in the floor, setting up some network equipment, and wiring the two networks. The resulting combination of networks has a special name – an *internetwork*, or *Internet* in short. A diagram of an internetwork is shown in Figure 1.1

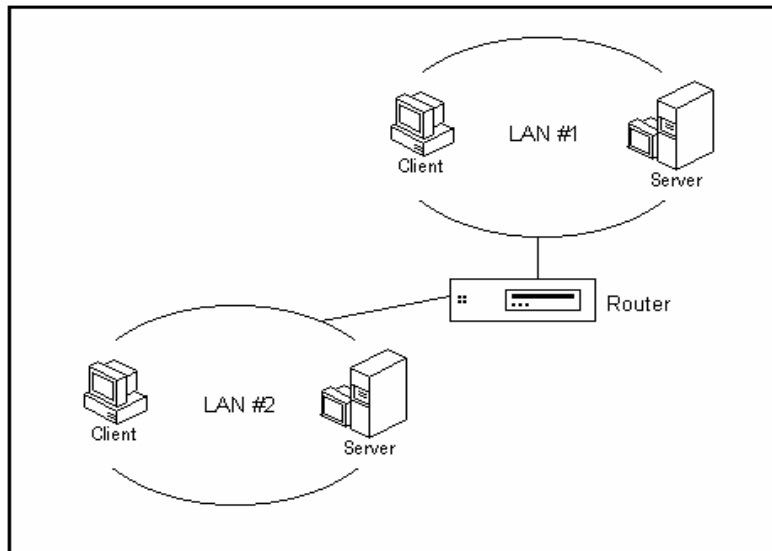


Figure 1.1 An internetwork.

An *internetwork*, or *Internet* is a collection of two or more distinct networks joined together, typically using a router, to form a larger “network of networks.”

A router is a system that runs software to manage the exchange of information between two different networks.

Though the network that contains two or more distinct networks can be called as Internet, this name (Internet) is given to the worldwide collection of networks. The Internet is similar to the international telephone system – no one owns or controls the whole thing, but it is connected in a way that makes it work like one big network.

Network of Networks can also form an Intranet. Unlike Internet, which is a global network, Intranet is a private network. But it uses the Internet communication standards and tools to provide information to the restricted users. For example, a company may setup a Web site that is accessible only to its employees who are geographically separated.

Basic Uses of the Internet

Networks tend to be used for communication, information sharing, and resource sharing. In this sense, the Internet is no different to any other network. Typically Internet is not used for resource sharing as a printer is shared in a LAN. But on the Internet Super Computer etc. are shared among scientists. Typical Internet users usually encounter resource sharing when they use telnet to access a remote database or use a timesharing system to do some work remotely. While this is not necessarily the main use of the Internet, as with all networks, resource sharing is a very important factor.

Telnet is a means of logging onto a remote computer system.

The most basic use- some say the most important use- of the Internet is communication in the form of electronic mail, or e-mail. With millions of users online, e-mail is a prime motivation for many firms to connect to the Internet.

There are millions of people using the Internet - and almost as many connected computers. Each one of these computers may have disk drives filled with information ready to be accessed by other Internet users. This motivates the third and most important aspect of the Internet at this point: information sharing. Many tools can be used to access information over the Internet, but most people tend to characterize information on the Internet by the World Wide Web. The Web is relatively new but very useful for information sharing.

1.3 History of Internet

The Internet grew out of an earlier U.S. Department of Defense project, the ARPANET (Advanced Research Projects Agency Network) that was put into place in 1969 as a pioneering project to test packet-switching networks. Packet switching is a technique for transmitting packets of information through multiple linked networks. ARPANET provided links between researchers and remote computer centers. In 1983, the military communication portion of ARPANET was split off into MILNET (Military Network), although cross-communication was still possible. ARPANET was officially dismantled in 1990. Its successor, Internet, continues to grow.

1.4 How the Web Works

Hypertext Transfer Protocol (HTTP) is a fast and efficient communication protocol that controls many different operations that take place between the Web browser client and the server. HTTP uses the Transmission Control Protocol (TCP) to transport all of its control and data messages from one computer to another.

Web pages are typically grouped at a Web site, where the main page is referred to as the home page. The user navigates by mouse clicking on hyperlinks displayed as text, buttons, or images. These hyperlinks reference other information. When you click a hyperlink, you jump to another part of the same page, a new page at the same Web site, or to another Website. You might also execute a program, display a picture, or download a file. All of this hyperlinking is done with Hyper Text Markup Language, which works in concert with HTTP.

To connect with a Web site, you type the Uniform Resource Locator (URL) for the site into the Address field of a Web browser. Here is an example of the URL that retrieves the Microsoft Home Page.

<http://www.microsoft.com>

When you type this request, the Web browser first gets the IP address of `www.microsoft.com` from a Domain Name System (DNS) server, and then connects with the target server. The server responds to the client and transfers this HTML-coded document to your Web browser. Your Web browser then translates and displays the HTML information.

1.5 Web Server and Clients

The World Wide Web has Client/Server architecture. This means that a client program running on our computer (our Web browser) requests information from a server program running on another computer somewhere on the Internet. HTTP is the command and control protocol that sets up communication between a client and server and passes commands between the two systems.

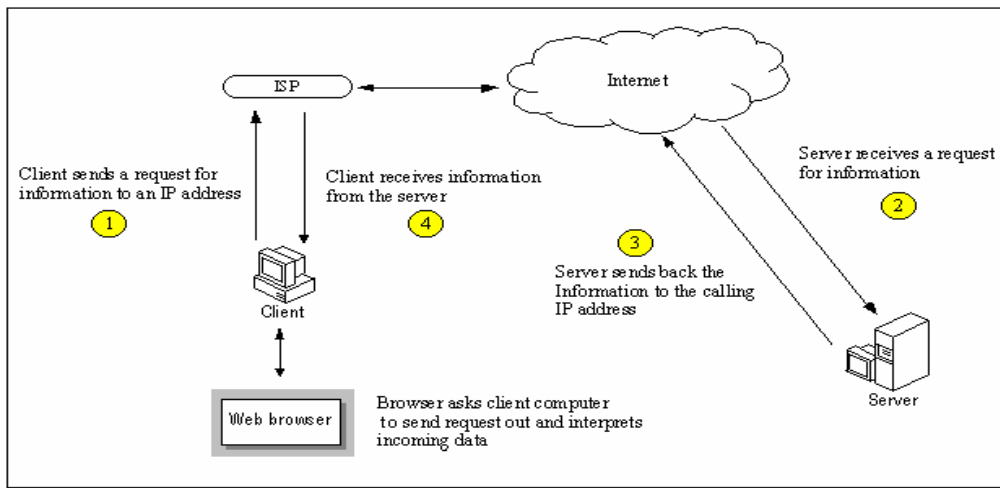


Figure 1.2 Web Server/Client

The way, by which the communication between *Server* and *Client* takes place, is via protocols. The Web browser handles much of the processing in connecting to a Web Site. It formats and displays the HTML information, which is transferred as a simple, relatively small file from the Web server.

The browser can also include add-ons that allow it to display video, sound, and 3-D images. These add-ons do most of the processing based on a relatively small set of commands or data transferred from the Web server, thus reducing excessive dialog between the Web client and server.

The Web client/server relationship is stateless, meaning that the server does not retain any information about the client, and the connection between the Web browser and Web server is terminated as soon as the requested information is sent. While this is efficient for activities such as downloading a single Web document, it produces a lot of overhead if the client keeps requesting additional information from the server.

Each new connection requires a negotiation phase, which takes time and requires the packets be exchanged between client and server. It should be noted that a Web server needs a more robust Internet connection than a Web client does.

1.6 Looking at Connection as ISP

Internet connections can be classified as connections for individual users or clients, and connections for servers. A typical client connection goes through an ISP (Internet Service Provider). You get an account, attach a modem to your computer, then dial into the ISP and log on. Once logged on, you can use a Web browser to access the Web. ISPs are everywhere. They basically lease high-bandwidth lines that connect into the Internet, then sublease the bandwidth to Internet users and Web sites. Home users connect via modems over dial-up lines to an access server at the ISP. This data link usually runs the PPP (Point-to-Point Protocol), which encapsulates and delivers IP (Internet Protocol) packets to the ISP's site. IP is an internetwork protocol. It provides a communication system that works across linked networks. The ISP then forwards these through a router to the Internet. Businesses that want to set up Web sites may also go through ISPs.

1.7 ISDN

Integrated Services Digital Network (ISDN) is an all-digital, circuit-switched telephone system that was originally designed by the world's telephone companies and service providers as a replacement for the aging analog telephone system. An all-digital system has many advantages, including reliability, scalability and a good fit for data transmission. ISDN services can provide end-to-end digital connectivity for consumers. ISDN requires special ISDN phones that convert voice into digital data. The ISDN is built on two main types of communication channels: B channels, which carry voice, data, or images at a rate of 64 Kbps (kilobits per second), and a D channel, which carries control information, signaling, and link management data at 16 Kbps. Standard ISDN Basic Rate desktop services is called 2B+D.

1.8 Dial-up or Leased Connection

A dial-up line is a connection or circuit between two sites through a switched telephone network (i.e., the telephone network). In the data communication world, a dial-up line forms a link between two distant computers. There are several variations to the dial-up account, which provide different capabilities, depending on the protocols used. All these connections require the Internet Protocol, and are therefore called IP accounts. Dial-up connections are typically used for individual client browsing and are not recommended for servers. A connection to the Internet using a telephone line and modem can serve up to 10 simultaneous users. The transmission rate is typically 28,800 bits/sec with the V.34 standard but higher rates (up to 56 Kbits/sec) are available for download speeds.

A Leased line is a communication circuit that is set up on a permanent basis for an organization. For all practical purposes we refer to a leased line as a private dedicated circuit. These are fixed point-to-point connections that provide a fixed rate. An organization uses leased lines to build private networks that interconnect its remote sites or the sites of business partners. The downside of this is that companies can't easily change their lines and the data rate is inflexible.

1.9 Domain naming system

Domain naming system (DNS) is a hierarchical client/server distributed database management system. In the DNS, the clients are called resolvers and the servers are called name servers. DNS servers are strategically located on the Internet of convert domain names to IP addresses. Your own Internet service provider may do this conversion or connect to a specific DNS server that does. When you type a domain name in a Web browser, a query is sent to the primary DNS server defined in your Web browser's configuration dialog box. The DNS server converts the name you specified to an IP address and returns this address to your system. From then on, the IP address is used in all subsequent communications. The top-level domains are listed here

COM commercial
EDU education
GOV government
ORG organizational
NET networks
INT International treaty organizations
MIL U.S. military organizations.

DNS address information is stored at many locations on the Internet, not just at one central depository. These sites "reflect one another" so that similar users can obtain information from the DNS server closest to them.

1.10 Registering our own Domain Name

When an organization applies for a domain name, it chooses a name that includes one of the domain names listed above. The only way you can register domain names is with the NSI (Network Solutions, Inc.), the Network integrator that manages Registration Services. You can go through an ISP, but they basically charge you a fee for registering your domain name with the InterNIC (Internet Network Information Center), which assigns all Internet domain names and ensures that no names or addresses are duplicated. Domain names are hot commodities and people or organizations want to register names that exemplify their products or services. For

example, Proctor and Gamble have registered a whole batch of domain names, including badbreath.com, dandruff.com etc. Registrations are made on a first-come, first-server basis, but a name that infringes on a registered trademark will be rescinded by NSI if the trademark holder contests the name.

1.11 Intranet

An Intranet is an internal network that implements Internet and Web technologies. An intranet runs the TCP/IP protocol and other Internet-related protocols, includes Web servers to publish information and provide access to back-end systems, supports Web browsers as a universal client interface, and supports Internet mail as the pervasive mail system. Some examples of applications that are easily deployed on Intranet platforms include

- ◆ Employee directory or personal Web pages for employees
- ◆ Collaborative applications such as calendaring / scheduling tools, group editing and workflow software
- ◆ Electronic meeting tools such as chat rooms, voice and videoconferencing and whiteboard applications
- ◆ Libraries for marketing, technical and other types of information

1.12 Short Summary

- ◆ The networks that usually take the form of a small office network are called as *Local Area Network (LAN)*.
- ◆ The networks that cover a city or some other large physical distance can be called *Metropolitan Area Network (MAN)* or, more generally, *Wide Area Network (WAN)*.
- ◆ An *internetwork*, or *Internet* is a collection of two or more distinct networks joined together, typically using a router, to form a larger “network of networks.”
- ◆ Telnet is a means of logging onto a remote computer system
- ◆ The world wide web has client/server architecture
- ◆ Internet connections can be classified as connections for individual users or clients, and connections for servers
- ◆ Domain naming system (DNS) is a hierarchical client/server distributed database management system.
- ◆ An Intranet is an internal network that implements Internet and Web technologies.

1.13 Brain Storm

1. What is Internet?
2. Evolution of Building Blocks of WEB
3. Applications of Internet
4. Client/Server Architecture
5. Types of Internet Connection
6. What is a Router
7. Discuss DNS
8. Intranet and its applications

Lab Unit (2 Real Time Hours)

1. Open Internet explorer and have a look at the Menu Bar, Tool Bar and the Address bar.
2. Enter the address of any of the portal and try navigating using the links provided.

For ex: www.yahoo.com

www.indya.com

3. Change the Font size of the browser window using the 'view -> Text Size' Menu.
4. Open a new instance of the browser and enter the address of any of the search engine.

Eg : www.altavista.com

www.yahoo.com

www.askjeeves.com

www.Google.com

5. Enter some word and notice the results provided by that search engine.
6. Try navigation through a few links provided by the search engine.

Lecture 2

Introducing HTML

Objective

This lecture provides an insight into the following:

- ❖ The web browsers in use
- ❖ The use of hypertext
- ❖ How to design a web page using web documents
- ❖ What Hyper Text Markup Language is and is not
- ❖ The HTML Elements
- ❖ How do you format your HTML text
- ❖ Rules for the HTML Language

Lecture - 2

- 2.1 Snap Shot
- 2.2 Overview of Web Browsers
- 2.3 Hypertext
- 2.4 Hyper Text Markup Language
- 2.5 Basic components of HTML
- 2.6 Formatting the Text HTML
- 2.7 Short Summary
- 2.8 Brain Storm

2.1 Snap Shot

The World Wide Web is the set of all Web sites and the documents they can provide to clients (users). This lecture introduces the Internet, in which the WWW is a subset and HTML (Hypertext Markup Language), which lies foundation and builds the WWW. A Web page is basically a text file that contains the text to be displayed and references to elements such as images, sounds, and, of course, hyperlinks to other documents. These concepts including basic components of HTML are discussed in this lecture.

2.2 Overview of Web Browsers

Web browsers provide a unique tool for accessing information on any network, whether it is an internal Intranet or the Internet. They remove the mystery of the Internet and eliminate the need for users to understand arcane commands. Most people begin accessing resources the first time they use a browser. Little training is necessary, and most browser software is free. Browsers do most of the work in accessing and displaying the documents, making the process almost transparent to the user. The traditional role of a Web browser has been to contact a Web site and obtain information from the site in the form of an HTML page. Let us take a look at the various Web browsers available to our users.

NCSA Mosaic:

NCSA (National Center of Supercomputing Applications), University of Illinois, in late 1993 introduced the Web browser NCSA Mosaic. Mosaic first introduced millions of Internet users to linking graphics and text in a browser window to documents located throughout the Internet. Mosaic's popularity was aided by its cross-platform presence; versions were available for the Windows and MacOS environments in addition to several flavors of Unix. Mosaic is still maintained by the NCSA and is free to users. Many browsers are based on the original Mosaic code.

Netscape:

The original developers of Mosaic left the NCSA in 1994 to form what eventually became Wall Street and Media darling Netscape Communications. The company's flagship product, the Netscape Navigator Web browser, is estimated to be used by up to 70 percent of all users of the Web. Netscape Navigator was the first browser to allow inline display of graphics written in JPEG format. Netscape Navigator also supports encrypted HTTP connections thereby facilitating growth of a booming online transaction market. With this browser, users are able to send encrypted information for use in online purchase of goods and services

Internet Explorer:

With Netscape dominating the Web browser and server market, Microsoft, in the early 1996 introduced a new Web browser and server to the Internet community. Microsoft Internet Explorer is destined to be one of the most popular Web browser for several reasons. Not only is the browser easy to use and is supported by the world's largest software company, it also supports several other specific HTML extensions. These include the following

- ◆ Background sounds played automatically when the Web page is loaded
- ◆ Inline animations of AVI files instead of graphic images
- ◆ Marquees that scroll across the browser window

2.3 Hypertext

Traditional text in the form of a book is typically defined as sequential or linear because there is an order in which the text must be read-page two follows page one, and so on. There are many advantages to this method of presenting information. It provides a logical sense of order. It can, however, be an inefficient way to access large bodies of information. (Imagine reading an entire 20-volume encyclopedia page by page to find a single relevant bit of information.)

A variety of mechanisms can speed a user's search for information within documents. For example, a book such as this one uses an index, table of contents, and section headings to speed access to various bits of information. The index provides a mapping from an idea to a particular page in the document containing these related pieces of information. Non-sequential ways to access information such as footnotes, references, and indexes are useful way to deal with navigating and organizing large bodies of related information. With the amount of information available for consumption, exploring an alternative to sequential access seems appropriate. This is where the idea of hypertext comes in.

A hypertext document is an electronic document that contains links to related pieces of information. It could be characterized as providing generalized footnotes. For example, a hypertext document about cows may feature a link from the word milk, as shown in Figure 2.1 Hypertext is a nonlinear way to access information. Many people find it similar to the way they think about problems.

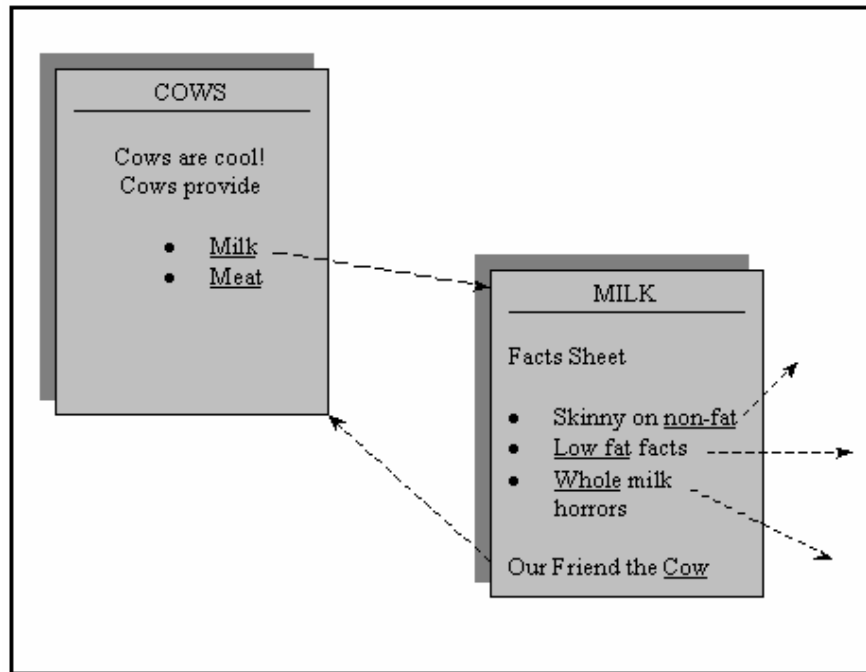


Figure 2.1 Hypertext documents with underlined links

2.4 Hypertext Markup Language (HTML)

HTML (Hypertext Markup Language) is the language that puts the face on the Web by helping to prepare documents for online publications. HTML documents are also called *Web documents*, and each HTML document is known as *Web page*. A page is what is seen in the browser at any time. Each *Web site*, whether on the Internet or Intranet, is composed of multiple pages. And it is possible to switch among them by following hyperlinks. The collection of HTML pages makes up the *World Wide Web*.

HTML pages can be created using simple text editor such as Notepad or WYSIWYG (What-You-See-Is-What-You-Get) Web page editors such as Microsoft FrontPage. In either case the result is a plain text file that computers can easily exchange. The *browser* interprets this text file and renders on the client computer.

Without knowing the HTML even a bit, one can create Web pages by using the Web Page editors. But this method has a drawback. These editors often introduce their own little nuances or quirks to the way code is produced. Moreover even the best editors do not support all the tags that are part of HTML at any given time.

This brings the first reason to learn the HTML. Sometimes it is necessary to directly modify the source of the page to add or change tags and attributes. Learning HTML will clarify how the tags relate to each other.

Second, HTML is as much an organizational tool, as it is design tool. Even with Web page editors, the rationale behind the tags is to give a structure and purpose to a

page. Learning how HTML organizes a page, leads to better planning and designing of Web pages for the readers.

Finally, HTML can be downright fun. It gives certain satisfaction from building a Web page from the ground up. It's like building an own house, in which every brick, every board, every nail etc. are known. This facilitates the easy modification to acquire the desired result. It also makes it much easier to take a look at someone else's page and know how they achieved their effect.

What HTML Is Not

HTML is a powerful technology, but there are many misconceptions about it. Many people have a basic notion of what HTML is, but few seriously consider what it isn't.

HTML is not a Programming Language

Many people think that making pages is like programming. However, HTML is unlike programming in that it does not specify logic. It specifies only the layout and structure of a document. HTML is a structured language; but unlike a programming language, in which the rules are enforced by the compiler, HTML rules are not strictly enforced by browsers. The language does not provide any sort of execution order like programming languages do.

HTML is Not a WYSIWYG Design Language

HTML is not a specific, screen or printer-precise formatting language. Many people struggle against the technology on a daily basis trying to create perfect layouts by using HTML elements inappropriately, or using images to make up for HTML's lack of screen and font-handling features.

HTML Is Not Complete

HTML is not finished. The language does not provide all the facilities it should. If HTML is meant to be a physical design language, it still does not provide font or pixel-level control, which is very important.

HTML Is Not All, Needed to Create Good Web Pages

While HTML is the basis for Web pages, it is required to know a lot more in order to build useful Web pages unless the page is very simple. Document design, graphic design, and even programming, are often necessary to create sophisticated Web pages. HTML serves as the foundation environment for all these tasks, and complete understanding of HTML technology can only aid document authors.

2.5 Basic components of HTML

HTML documents are created by combining special markup code called *tags*. The tags define the structure of the document and provide the framework for holding the actual content, which can be text, images, or other special content. When a browser reads a document that has HTML markup in it, it determines how to render it onscreen by considering the HTML elements (tags) embedded within the document (Figure 2.2)

An HTML document is simply a text file that contains the information, which is to be published. It also contains embedded instructions, called elements (tags) that indicate how a Web browser should structure or present the document. HTML elements generally consist of a pair of angle-bracketed tags surrounding some text. The *end-tag* (`</TAG>`) is just like the *start tag* (`<TAG>`), except that it has a slash (/) in it as shown:

```
<TAG>      ← Start Tag
           -----
           Text that the tags affect
           -----
</TAG>     ←End Tag
```

HTML elements indicate the “markup” on the surrounded text. They may indicate the meaning of the enclosed information (for example, a citation) or how it should be rendered (for example, in bold). HTML elements normally consist of a pair of tags called *container tags*, because content goes between them. However, some elements, such as the horizontal rule tag `<HR>`, do not have a corresponding end tag. These are termed empty elements. For some other elements, like the paragraph element `<P>`, the end tag may be optional. It is always good idea to use the end tag if one is available. Given the following HTML code,

```
<B> Indian </B>
```

a Web browser should render the phrase “Indian” in a bold typeface.

An HTML file usually begins with the `<HTML>` element, which indicates that the contents of the file include markup. The file should end with that element’s end tag, `</HTML>`. The rest of a typical HTML file is composed of the head and the body.

The head, which is enclosed with the `<HEAD>` element (consisting of the `<HEAD>` and `</HEAD>` tags), includes supplementary information about the document, such as the title of the document, which most browsers display in a title bar at the top of the browser window. The title is indicated between the `<TITLE>` and `</TITLE>` tags. The document title is required under the current HTML specification. While some browser may not require the inclusion of the `<TITLE>` element, it should always be included for correctness, book marking, and the sake of good HTML style.

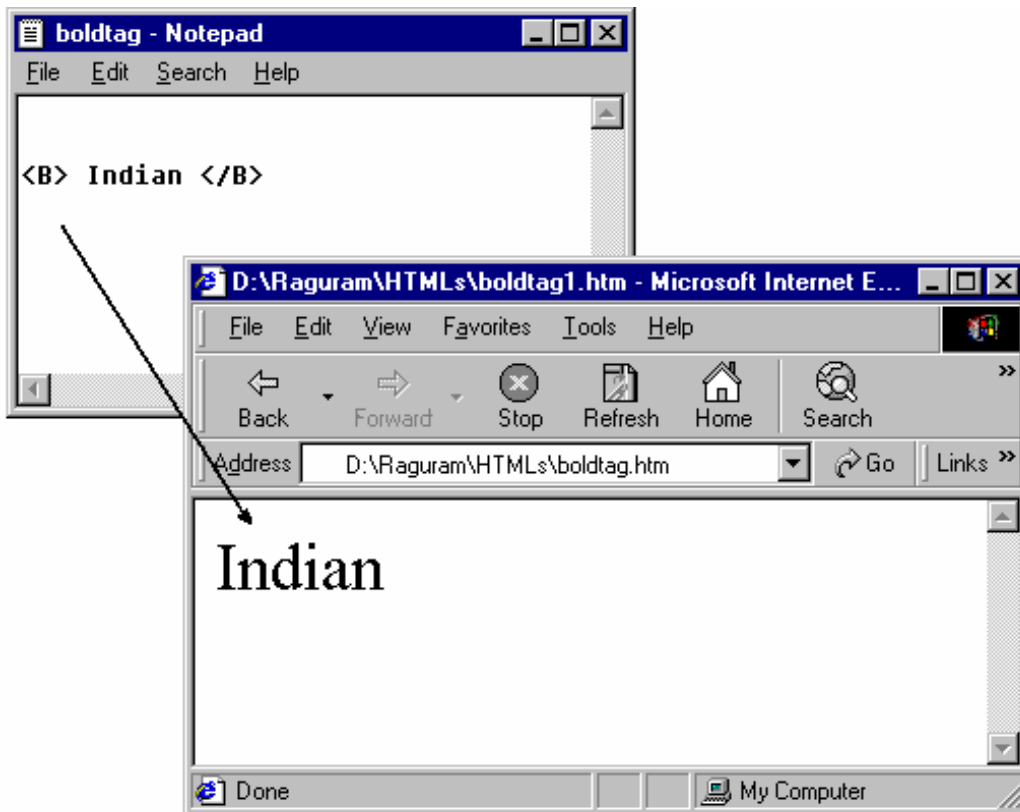


Figure 2.2 Interpretation of a Web page with HTML markup.

The body, which is enclosed between `<BODY>` and `</BODY>` tags, contains the actual content and appropriate markup tags needed to render the page.

A basic HTML template is as follows:

```

<HTML>
<HEAD>
    <TITLE>
        Document Title
    </TITLE>
</HEAD>
<BODY>
    The Document's body: text, images, sounds, and HTML commands
</BODY>
</HTML>
    
```

The HTML Elements

The following sections describe the basic elements used in formatting an HTML document and also some other most commonly used elements.

Setting the Boundaries with `<HTML>`

The first tag to learn about is `<HTML>`. It's paired with `</HTML>` to encase all the other tags in an HTML page; the two mark the absolute beginning and end of the file.

Syntax

```
<HTML>  
---document and tags---  
</HTML>
```

The `<HTML>` tag is an optional element for building Web pages. If it's not explicitly included most browsers or other user agents will assume its existence, along with the appropriate defaults for text direction. However, some browsers might be confused when handed a document without any clear indication of type with HTML, resulting in a page that's displayed unpredictably, displayed as plain text or that refuses to display at all.

There are a couple of good reasons to use the `<HTML>` tag. First, HTML isn't the only markup language on the Web. There are cousins to HTML, such as extensible markup language (XML) that are interpreted in slightly different ways and are gaining in acceptance and use. Second, using the `<HTML>` tag is good style and shows that whoever builds the document had some idea what to do.

The HEAD Element

Theoretically, every document has header and a body. The header of the document is where global settings are defined; it's contained between the `<HEAD>` and `</HEAD>` tags.

Syntax

```
<HEAD>  
    header content  
</HEAD>
```

It is a good place to place `<TITLE>`. Besides it is a favorite place to include scripting language function definitions.

Giving to a Page a `<TITLE>`

Probably the most commonly used HEAD feature in HTML, and the only required element is the `<TITLE>` tag.

```
<TITLE>Text</TITLE>
```

In this tag, *text*, is a short, one-line name for the document that's displayed in the browser's title bar. Without title, most browsers default to the HTML filename. Because file-names aren't always terrible descriptive and are sometimes long and clunky, it's good practice to supply a title for all HTML documents.

Only one title is allowed per document and its size is limited by the size of the user's browser window. Of course, a title can be of any length, but it is truncated in the browser's title bar if it stretches beyond the limits.

The BODY Element

Like the <HEAD> tag, the <BODY> tag's primary purpose is to delineate the main portion of the document - the part seen by the user.

Syntax

```
<BODY  
  [ ALINK = color name | #RRGGBB ]  
  [ BACKGROUND = path of background image ]  
  [ BGCOLOR = color name | #RRGGBB ]  
  [ BOTTOMMARGIN = pixels ]  
  [ LEFTMARGIN = pixels ]  
  [ LINK = color name | #RRGGBB ]  
  [ RIGHTMARGIN = pixels ]  
  [ SCROLL = YES | NO ]  
  [ TEXT = color name | #RRGGBB ]  
  [ TITLE = advisory text ]  
  [ TOPMARGIN = pixels ]  
  [ VLINK = color name | #RRGGBB ] >
```

--The Document's body: text, images, sounds, and HTML commands
</BODY>

The attributes of the <BODY> element is described below:

ALINK

This attribute sets the color for active links within the document. Active links represent the state of a link as it is being pressed. The value of the attribute can either be a browser-dependent named color or a color specified in the hexadecimal #RRGGBB format.

BACKGROUND

This attribute contains a path for an image file, which will be titled to provide the document background.

BGCOLOR

This attribute sets the background color for the document. Its value can be either a browser-dependent named color or color specified using the hexadecimal #RRGGBB format.

BOTTOM MARGIN

This attribute specifies the bottom margin for the entire body of the page and overrides the default margin. When set to 0 or "", the bottom margin is the bottom edge of the window or frame the content is displayed in.

LEFTMARGIN

This Internet Explorer-specific attribute sets the left margin for page in pixels, overriding the default margin. When set to 0 or "", the left margin is the left edge of the window or the frame.

LINK

This attribute sets the color for hyperlinks within the document that have not yet been visited. Its value can be either a browser-dependent named color or color specified using the hexadecimal #RRGGBB format.

RIGHTMARGIN

This Internet Explorer-specific attribute sets the right margin for page in pixels, overriding the default margin. When set to 0 or "", the right margin is the right edge of the window or the frame.

SCROLL

This attribute turns the scroll bars on or off. The default value is YES.

TEXT

This attribute sets the text color for the document. Its value can either be a browser-dependent named color or a color specified in the hexadecimal #RRGGBB format.

TOPMARGIN

This Internet Explorer-specific attribute sets the top margin for page in pixels, overriding the default margin. When set to 0 or "", the top margin will be exactly on the top edge of the window or frame.

VLINK

This attribute sets the color for links within the document that have already been visited. Its value can be either a browser-dependent named color or color specified using the hexadecimal #RRGGBB format.

Headings <H1> Through <H6>

No page begins with a lead paragraph but with a headline. The six HTML heading styles are a way of showing the level of importance among different parts a page.

Syntax

`<Hn> Heading Text </Hn>`

Here, n is an integer from 1 to 6 indicating the level of heading used, with 1 being the most important and 6 being the least. Although browsers vary in the size and typestyle given to the six headings levels, every browsers follows the basic rule of giving the biggest and boldest style to <H1> and the smallest and most unobtrusive style to <H6>. The following lines produce the document as in Figure 2.3.

```
<HTML>
<HEAD>
<TITLE>Headers</TITLE>
</HEAD>
<BODY BGCOLOR="Black" Text="White">

<H1> I am Heading 1 </H1>
<H3> I am Heading 3 </H3>
<H5> I am Heading 5 </H5>
<H6> I am Heading 6 </H6>

</BODY>
</HTML>
```

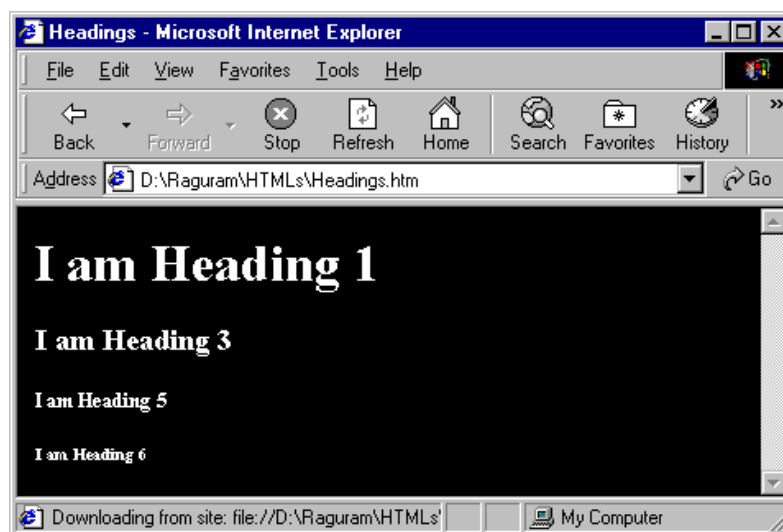


Figure 2.3 Different Styles of Headings.

A New Paragraph: <P>

Now that the headings are in place, it's time to turn the attention to the rest of the document. The next step is to start breaking the text into paragraphs. The following listing appears to be a Web page broken into logical paragraphs, but look at Figure 2.4 to see how a browser interprets it.

```
<HTML>
<HEAD>
<TITLE>Proverbs</TITLE>
</HEAD>
<BODY>
```

Don't forget - enemies are made, not born.

Live by the advice you give to others.

Live for others and they will not forget you.

```
</BODY>
```

```
</HTML>
```

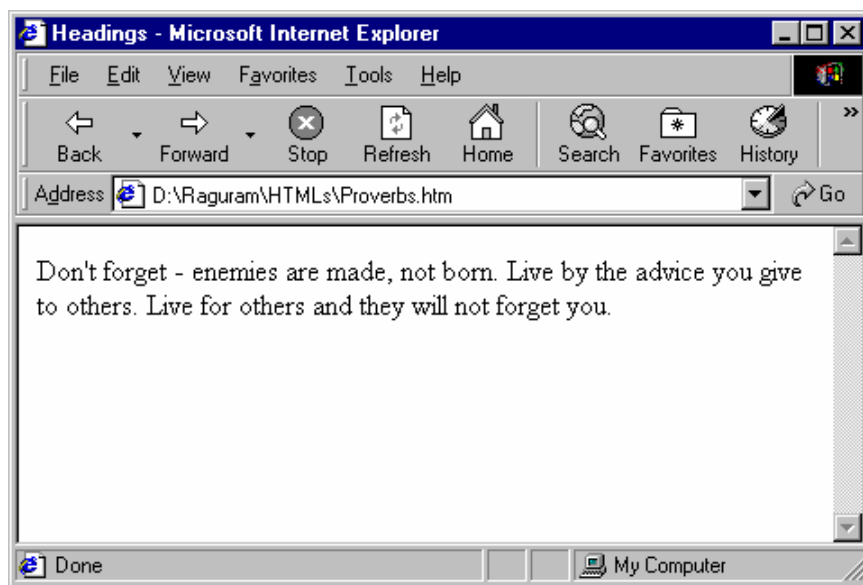


Figure 2.4 All lines appearing together without any line break.

This result is obviously isn't what the designer had in mind. The browser has ignored the extra line breaks and runs the lines together into one long paragraph.

The first way to break text into paragraphs is to use the paragraph tag: <P>.

Syntax

```
<P [ STYLE = Style Information ] > ....Text....</P>
```


Text is a line or paragraph text that should begin on a new line and remain together. The closing tag is optional and can be omitted. The STYLE attribute has Property:Value pair to specify appearance. Each Property:Value pair is delimited by semi-colon (;) and, Property and Value itself by colon (:). For example, the following line constructs a paragraph with green text color, yellow background, font-weight normal and font-size 20.

```
<P STYLE= "Color:Green; Background-color:Yellow; Font-weight:Normal;Font-size:20">
```

```
Don't forget - enemies are made, not born. </P>
```

The preceding listing is rewritten with this paragraph tag as given below:

```
<HTML>
<HEAD>
<TITLE>Proverbs</TITLE>
</HEAD>
<BODY>

<P>Don't forget - enemies are made, not born. </P>

<P>Live by the advice you give to others.</P>

<P>Live for others and they will not forget you.</P>

</BODY>
</HTML>
```

This listing produces the document as in Figure 2.5

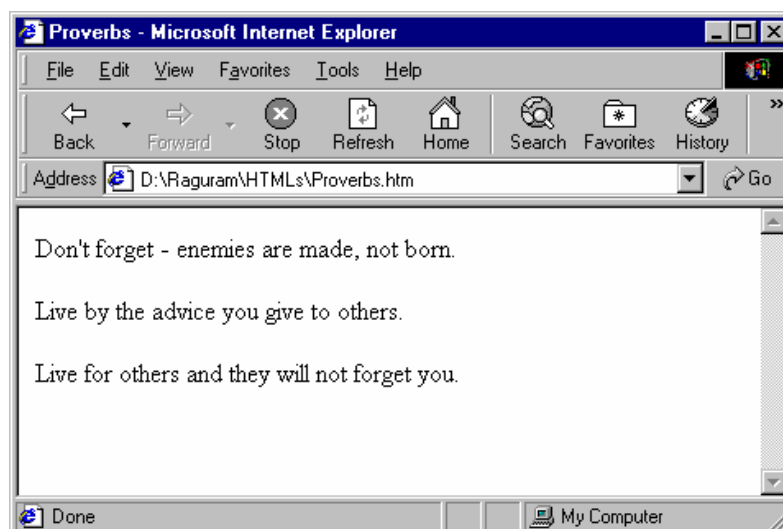


Figure2.5 Sentences as new paragraphs due to <P> tag.

**A New Line:
**

A blank return tag,
, which is an empty tag, is similar to a paragraph tag. But it behaves in a slightly different way. It starts a new line within the current paragraph, but it doesn't start a new paragraph.

The following lines produce a document in Figure 2.6 demonstrating the difference between the <P> tag and
 tag.

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>
```

```
Paragraph and Blank Return Tags
```

```
</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
This document demonstrates the difference between  
the Paragraph(P) and Blank Return(BR) tags.
```

```
<P> The Paragraph tag forces the browser to insert a carriage return to begin a  
new paragraph. It also causes the browser to insert additional vertical space before  
and after the paragraph.
```

```
</P>
```

```
<P>
```

```
But the Blank Return tag just inserts a carriage return <BR>  
without inserting additional vertical space.
```

```
</P>
```

```
</BODY>
```

```
</HTML>
```

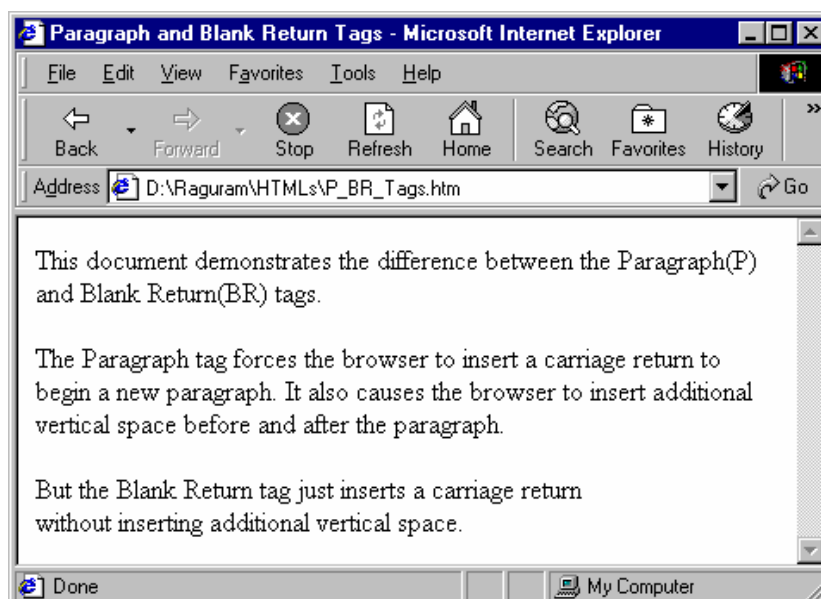


Figure 2.6 An HTML document demonstrating the difference between the <P> and
 tags.

2.6 Formatting the HTML Text

Text plays predominant role in Web pages. Reader's eyes should not turn to red when he looks at the text in the Web pages. For that the text must be well aligned and formatted to ensure readability. HTML provides many elements to attain the same. Let us now discuss them in detail.

The Meaning Elements

The "meaning" elements are used to identify words or phrases that have meaning to the text. These elements show characteristics above the text. For example, a person's name would be identified by the <PERSON> element, and the title of a book might be identified by the <CITE> element. These elements enhance the quality of your documents by adding structure to the document. Indexing tools and programs that you write can use these elements to identify pieces of the text that you as the author found important. When publishing documents on the Web, where data is valuable, representing a document in a structured manner makes all the difference in the world

The (EMPHASIS) Element:

provides emphasis and is typically displayed as italics. The element should be used where the word or phrase should be emphasized rather than italicized.

The <CITE> Element:

is to denote a citation. The citation is usually displayed in italics.

The Element:

denotes strong emphasis. is usually displayed as bold

The <CODE> Element:

denotes that the text is some type of computer output or example of programming code. <CODE> is usually displayed in a fixed-pitch font. The same font is usually used with <PRE>; however <PRE> is for preformatted text while the browser will format the text associated with <CODE>

The <SAMP> Element:

denotes characters that are to used as sample text

The <KBD> Element:

denotes text that might be typed at the keyboard by a user. Of course, this is useful when writing books like this one or for writing instructions to be followed

The <VAR> Element:

denotes the text is a variable name as one might find in a C or Perl program

The <DFN> Element:

denotes the definition of a term

The <Q> Element:

is used to denote a short quotation

The <AU> Element:

is used to identify an author

The <PERSON> Element:

is used to identify the names of people

The <ACRONYM> Element:

denotes the word or phrase to be an acronym

The <ABBREV> Element:

is used to denote abbreviations

The <INS> Element:

denotes text that has been inserted into the surrounding text. This is handy when trying to show differences to a document, much like change bars are used in some word processors

The Element:

denotes text that has been deleted from the surrounding text. This is handy when trying to show differences to a document, much like change bars are used in some word processors

Example:

```
<html>
<head>
  <title>Logical Formatting Examples</title>
</head>
<body>
```

The spirit of the law is sometimes more important <p>

To be or not to be", <CITE>Shakespeare</CITE> <p>

When coming to a red light at an intersection, **you must stop**

One of the first lines that every C programmer learns is: puts ("Hello World!");

<H1>This is the first headline</H1>

At the prompt enter your login name.

Login: farrar

At the next prompt enter your password. Your password will not be displayed on the screen

Password: !!urdead

In the above perl example, the \$height is a constant, while \$width increments by 5

Aqua is typically known as water

To be or not to be? That is the question

</body>

</html>

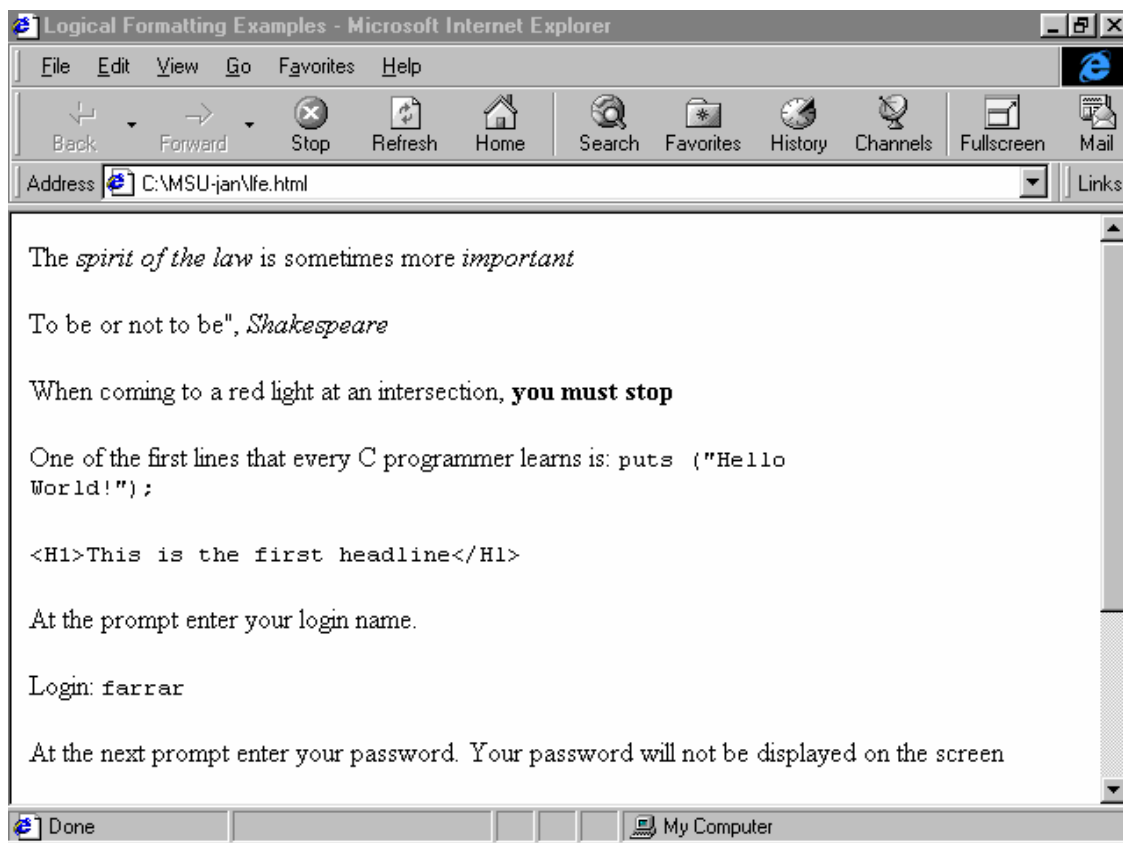


Figure 2.7 Example for The Meaning Elements

The Look Elements

The “look” elements deal with how the text is to be displayed not its meaning. These are useful when you want to italicize or bold a word or phrase

The **** (BOLD) Element:

is used to denote boldface. Bolding characters make them stand out in the text.

The **<I>** (ITALIC) Element:

is used to denote italics

The **<TT>** (TELETYPE) Element:

is for teletype or typewriter style fixed-pitch font

The **<U>** (UNDERLINE) Element:

is used for underline. Underline is widely supported by most browsers, but some ignore the **<U>** element or display the text as italics

The **<STRIKE>** Element:

is used for showing strikethrough. This is handy when you want to show a phrase is no longer needed, while still allowing the text to be seen

The **<BIG>** Element:

is used to display the character, word, or phrase in a larger font compared to the rest of the text

The **<SMALL>** Element:

is used to display the character, word or phrase in a smaller font compared to the rest of the text

The **<SUB>** Element:

is for showing subscripts (those below the line)

The **<SUP>** Element:

is for showing superscripts (those above the line)

Example

```
<html>
<head>
  <title>Look Elements</title>
</head>
<body>
  This is in <B> Bold </B> text<p>
  This is in <I> Italics </I> text<p>
  Example for <TT> teletype </TT> text<p>
  Example for <U> Underlined </U> text<P>
```

This text has been ~~striked~~
 Once upon a time, in a land not so far away, in a time not so long ago..
Once upon a time, in a land not so far away, in a time not so long ago...
 H₂O is the chemical designation of water
 Pi * r² is the mathematical formula for the area of a circle
 </body>
 </html>

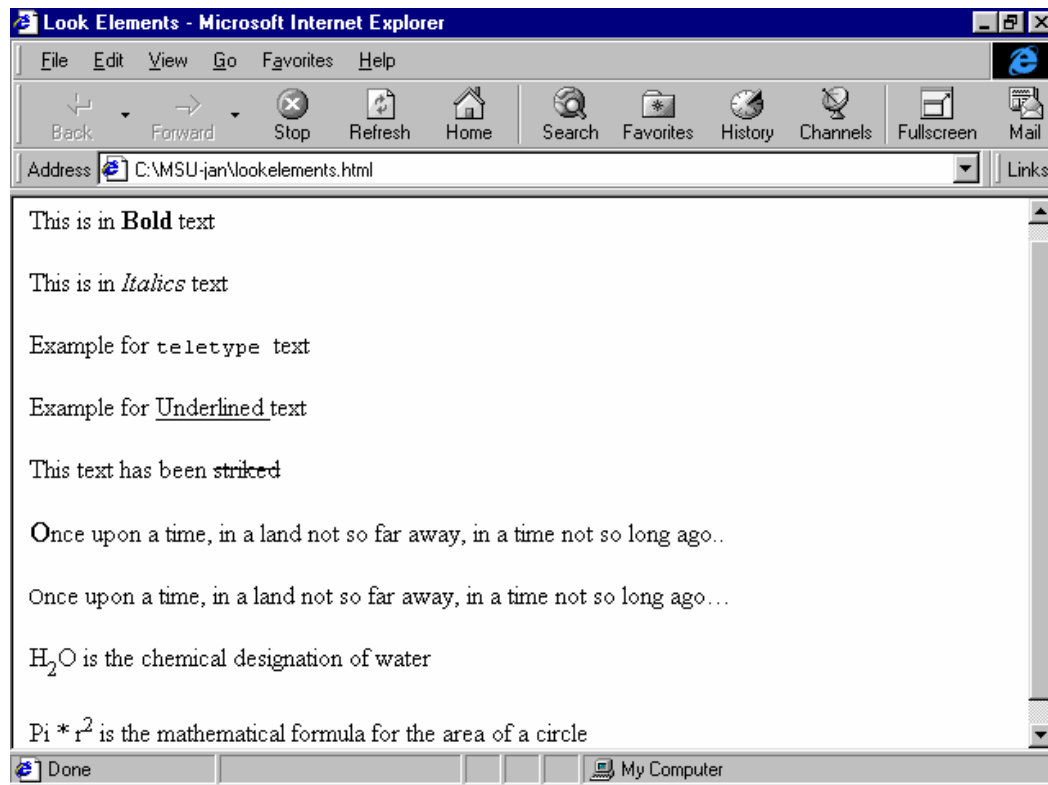


Figure 2.8 Example for the Look Elements

Preformatted Text, Spaces and All: <PRE>

The method for making several lines of text appear in the way they are typed is to mark the whole section as preformatted text using the <PRE> element.

Syntax

<PRE> ...Text...</PRE>

Text is any text, including returns, spaces, and other hard formatting. See the following listing and Figure 2.9. All the text is contained by the <PRE> tags, which causes the browser to display everything in between exactly as it finds it. It also uses a mono-spaced font for display, which helps format the text into rows and columns for data presentation.

```

<HTML>
<HEAD>
<TITLE>Hot Water Safety</TITLE>
</HEAD>
<BODY BACKGROUND="D:\Raguram\Images\backgrd.gif">
<B>
<PRE>
Hot water is dangerous.

    Turn down the hot water heater thermostat.

        Always supervise children in the bathtub.

            Don't encourage children to play around the tub.
</PRE>
</B>
</BODY>
</HTML>

```

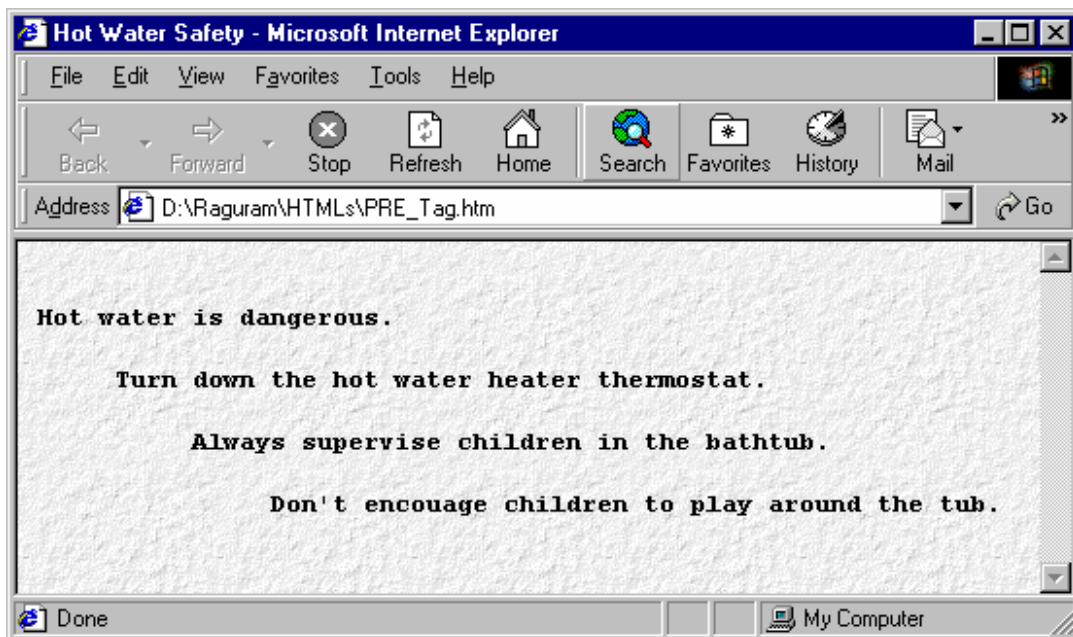


Figure 2.9 Appearance of the preformatted text.

Notice in the above Figure 2.9 that the mono-space font preserves character spacing for accurate display of indents and other formatting.

The mono-spaced font helps when preserving the line breaks and spaces inserted by the user. If he start a new line, insert five spaces, and then start typing, that's exactly how the text appears when using the `<PRE>` tag. Browsers disable automatic word wrap to further ensure that the text is displayed exactly as it is typed.

For browsers that aren't compatible with tables (explained in the next Chapter), preformatted text is an effective way to display a table made out of textual elements representing the border lines, such as the following:

```
<PRE>
=====
Country      |      Gold Medals
=====
India        |      199
-----+-----
Russia       |      101
-----+-----
USA          |      99
-----+-----
Japan        |      30
-----+-----
China        |      01
=====
</PRE>
```

Address Information: <ADDRESS>

The <ADDRESS> tag is used to mark contact information for the current document, whether it's an email address or a complete mailing address and phone number. It behaves much like a paragraph tag, forcing the text within its confines to be separated from surrounding material *by additional line spacing*.

Syntax

```
<ADDRESS>...ContactInformation...</ADDRESS>
```

ContactInformation is the address information, along with any paragraph-level formatting such as line breaks. It's typically displayed as italic body text as in Figure 2.10.

```
<ADDRESS>
Godzilla <BR>
-9182/~ Rocket Highway<BR>
Moon<BR>
(605) 374-5716
```

Drawing a Line on the Page: <HR>

Horizontal lines are an easy-to-use element for dividing a page into logical sections. They signal the reader to be alert for a change in subject or style or they can separate figures and captions from body text. This tag also includes several attributes to fine-tune its appearance.

</ADDRESS>



Figure 2.10 Text formatted with <ADDRESS>.

<HR [WIDTH=lineWidth] [SIZE=lineThickness] [NOSHAE]>

Here, *lineWidth* and *lineThickness* set the basic appearance of the line. The height of a horizontal rule is set in pixels, such as SIZE=6. The WIDTH is set either in pixels or as a percentage of the browser window width. If percentage value is used, it must be suffixed with a percentage sign (%) after the value. The typical defaults for height and width are 3 pixels high and 100 percent of page width (Figure 2.11.)

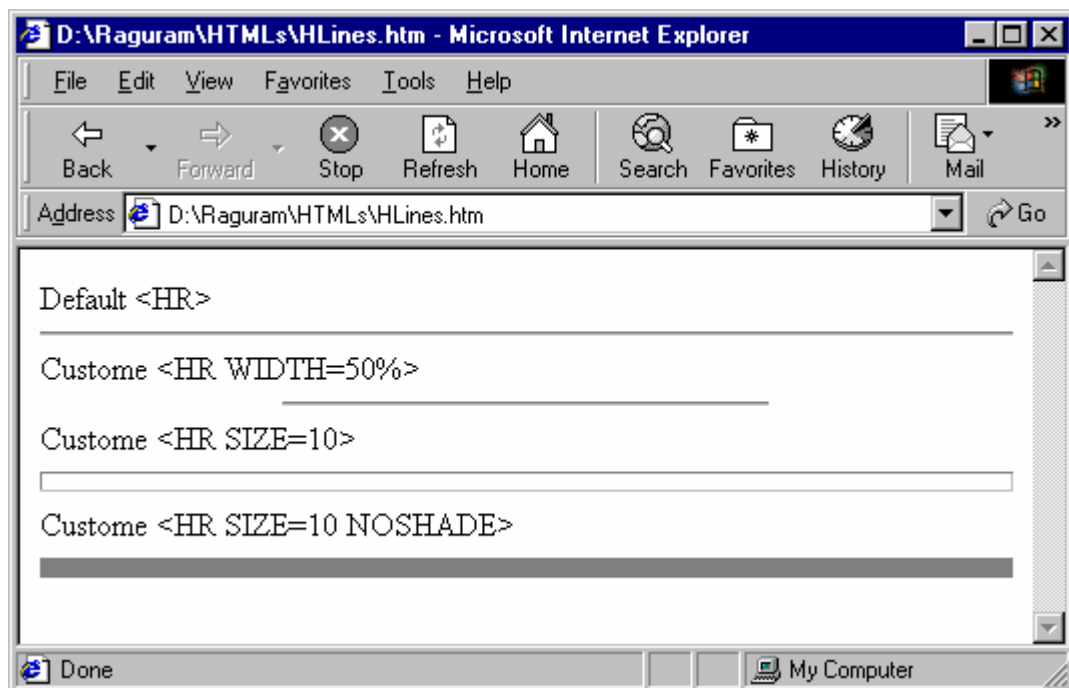


Figure 2.11 Different styles of Horizontal Lines.

The last attribute is NOSHADE. Normally, a browser displays a horizontal rule in some form of three-dimensional shading. This shading varies from browser to browser, some show it as depressed, some raised, some as an outline. Using NOSHADE forces the horizontal rule appear as a thick line with no additional highlighting. Again, this appearance can vary among browsers. Some use black for the rule and others use shades of gray.

Centering a Block: <CENTER>

Text or embedded objects like images that are enclosed within <CENTER> and </CENTER> are centered in the browser according to its current coordinates. The <CENTER> element is simply an alias for <DIV ALIGN= "CENTER"> and is treated exactly the same way. It is unlikely that the <CENTER> element will go away, considering its simplicity and widespread use.

Syntax

```
<CENTER>...Text or images </CENTER>
```

Dividing document into Sections: <DIV>

The <DIV> element is used to structure HTML documents into unique sections or divisions.

Syntax

```
<DIV [ALIGN = "LEFT" | "RIGHT" | "CENTER"]  
    [ STYLE = style information ]>
```

```
    ... Text or images ...
```

```
</DIV>
```

The ALIGN attribute makes it possible to align a portion of the document to the left, right, or center. By default, content within the <DIV> element is left aligned. Divisions are much useful when they are used with the STYLE attribute. Because this attribute enables to specify segments position, appearance as in below:

```
STYLE = "Position: Absolute; Left:20; Top:30; Font-weight: Bold; Height:100;  
Width:100; background-color:Yellow;color:Black"
```

The first property-value pair, Position:Absolute, tells the browser to position the DIV (segment) with respect to the top edge and left edge of the browser window.

Another possible value of for Position is Relative. In contrast with Absolute, Relative situates the segment in relation to other segments on the page.

The properties left and top specifies the x and y coordinates of the segment. The Height and Width properties species the size of the segment.

The following listing, which uses three <DIV> elements, produces the document as in Figure 2.12

```
<HTML>
<HEAD>
<TITLE>Divisions</TITLE>
</HEAD>

<BODY BACKGROUND="D:\Raguram\Images\backgrd.gif">

<CENTER>
Little Things          <BR>
<I> that make a </I>   <BR>
<BIG><BIG>BIG</BIG></BIG>    <BR>
<I> Difference </I>
</CENTER>

<DIV ALIGN="Left" STYLE= "Position:Absolute;Left:20;Top:120;Height:80; Width:120;
background-color:Yellow;color:Black" >
You can win more friends by your ears than by your tongue.
</DIV>

<DIV ALIGN="Center" STYLE= "Position:Absolute;Left:190;Top:120;Height:80; Width:120;
background-color:Yellow;color:Black" >
Your children are mirrors - the reflection of what you live.
</DIV>

<DIV ALIGN="Right" STYLE= "Position:Absolute;Left:360;Top:120;Height:80; Width:120;
background-color:Yellow;color:Black" >
The best way to destroy your enemies is to make them friends.
</DIV>

</BODY>
</HTML>
```

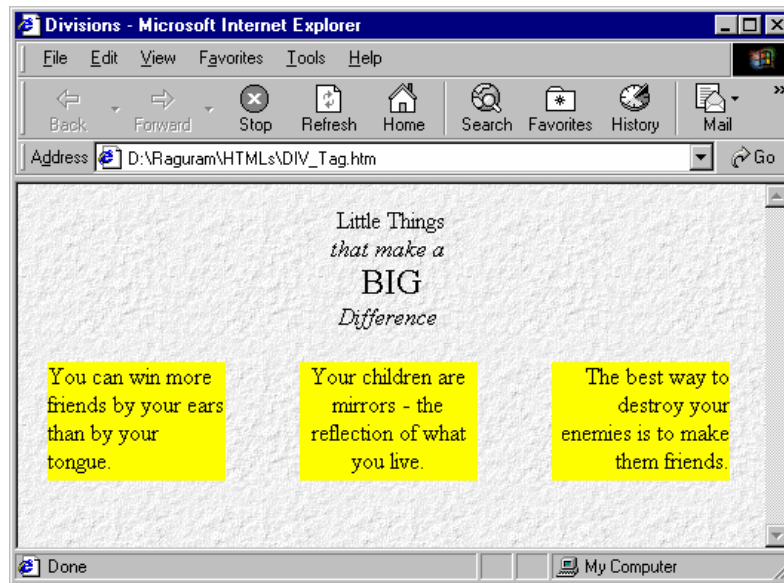


Figure 2.12 Appearance of the document that uses Divisions.

Character Entities

After covering the basic text formatting elements, there would appear to be nothing left to talk about – but there is still one more level to HTML documents: the characters themselves.

Sometimes it is necessary to put special characters within a document. These include accented letters, copyright symbols, or even the angle brackets used to enclose HTML elements. To use such characters in an HTML document, they must be “escaped” using a special code. All character codes take the form *&code*; where code is a word or numeric code indicating the actual character that is to be put on the screen. Some of the more commonly used characters are shown in the following table.

Numeric Value	Named Value	Symbol	Description
"	"	“	Quotation mark
&	&	&	Ampersand
<	<	<	Less than
>	>	>	Greater than
™	N/A	™	Trademark
 	 		Non-breaking space
©	©	©	Copyright symbol
®	®	®	Registered trademark

A Few common character entities.

The following listings which uses character entities produces the document as in Figure 2.13

```
<HTML>
<HEAD>
```


Commenting the Lines: <!--...-->

While designing large web pages, the designers may get confused even with his own sentences. This can be avoided by adding meaningful comments. HTML provides tags to make comments whose syntax is given below:

```
<!-- ... -->
```

Example

```
<!-- This is an informational comment that can occur
```

```
anywhere in an HTML document -->
```

HTML Rules (For Self Reading):

There are some rules to remember when writing HTML. These are briefly reviewed here.

- ◆ **Element names are not case sensitive.** An element like <hTml> is equivalent to <html> or <HTML>. Element case does not matter to a browser. However, writing elements consistently in upper – or lowercase makes HTML documents easier to understand and maintain. Convention suggests that uppercase is the preferred practice.
- ◆ **Attribute names are not case sensitive.** Just as <body> is equivalent to <BODY>, <BODY BGCOLOR= “Yellow”> is equivalent to <BODY bgcolor= “Yellow”>. As with elements, consistent use of case improves legibility, and uppercase is preferred.
- ◆ **Attribute values may be case sensitive.** The value of an attribute may be case sensitive, especially if it refers to a file. The file name in may not be the same as the filename in ; it depends on whether or not case matters to the operating system of the server containing the file. For best results, always specify a filename exactly as it has been saved.
- ◆ **Element names cannot contain spaces.** Browsers treat the first space encountered inside an element as the end of an element’s name and the beginning of its attributes. For example, <I M G> does not mean , the image element. It means <I>, the italic element, with two undefined attributes M and G.
- ◆ **Browsers collapse and ignore space characters in HTML content.** Browsers collapse any sequence of spaces, tabs, and returns in an HTML document into a single space character. These characters convey no formatting information, unless they occur inside a special formatting element like <PRE> which preserves their

meaning. Extra spacing can be used liberally within an HTML document to make it more legible to HTML authors.

- ◆ **An element that encloses the start tag of another element must also enclose its end tag, if one exists.** Elements often contain other elements inside the document section they enclose. Any element that starts within a section enclosed by another must also end there. In other words, an element's tag pairs should be nested within one another and their end tags should not cross. To make some text bold and italic, use `<I>Correct</I>` and not `<I>Not Correct</I>`.
- ◆ This is primarily a stylistic issue, since no major browsers at this time have problem with this. Authors are still advised to nest tags rather than cross them. Incorrect nesting may result in incompatibilities with emerging technologies, and may cause rendering problems in some esoteric browsers.
- ◆ **Browsers ignore unknown elements.** Browsers ignore elements they do not understand. They do attempt to interpret any content enclosed by an unknown element. If a browser does not understand the `<STORY>` element in `<STORY> Titanic - A Tale of Love </STORY>`, it ignores it. It does, however, render the words "Titanic - A Tale of Love" as normal text.
- ◆ **Browsers ignore unknown attributes.** As with elements, browsers ignore any attributes they do not understand.

2.7 Short Summary

- ◆ The web browsers in use - NCSA Mosaic, Netscape and Internet Explorer
- ◆ A hypertext document is an electronic document that contains links to related pieces of information.
- ◆ Hypertext is a nonlinear way to access information
- ◆ HTML (Hypertext Markup Language) is the language that puts the face on the Web by helping to prepare documents for online publications.
- ◆ The collection of HTML pages makes up the *World Wide Web*
- ◆ HTML is Not a WYSIWYG Design Language
- ◆ An HTML file is enclosed within `<HTML>` and `</HTML>` elements, which indicates that the contents of the file include markup.
- ◆ The "*meaning*" elements are used to identify words or phrases that have meaning to the text.
- ◆ To put special characters within a document, we use the Character Entity

2.8 Brain Storm

1. What is a Web browser? Discuss the browsers available.
2. Hypertext is a nonlinear way to access information – Discuss.
3. Is HTML a programming language?
4. List down the benefits of HTML
5. What is an HTML Tag (Element)?
6. Discuss the parts of an HTML document
7. What is the significance of using a Head Tag in an HTML Document?
8. What is the Tag used for setting the background color for an HTML Document?
 - a. BG color
 - b. Background color
 - c. Bgrnd color
 - d. Backcolor
9. List down the various attributes of body tag.
10. What are the ways of giving values for color?
 - a. colorname
 - b. RRGGB format
 - c. RGB combination
 - d. None of the above
11. What is the range for the Headings Tag?
12. What is the difference between the output obtained using the <P> and
 Tags?
13. Differentiate between the Meaning Elements and Look Elements.
14. What is the use of <ADDRESS> Tag? How it differs from <I> Tag?
15. What is the significance of using <DIV> tag?
16. HTML Elements are case sensitive – True or False?

Lab Unit (2 Real Time Hours)

At the end of this exercise you would have completed building your WEB page.

-Open the Notepad and give the necessary TAGS required for a WEB Page.

-The WEB Page is to display Information about RADIANT in the following way:

-The page should have a title 'Radiant Home Page'.

-The heading should be 'RADIANT SOFTWARE LIMITED'.

-The contents of the page:

RADIANT the pioneer in higher end technology training and specializes in Client Server Computing, GUI and OOPS methodology and Dedicates itself to evolve a New Era in Software Education.

Radiant Provides Job Oriented training with Ultimate Technical Excellence.

CITE Offers the Following Courses:

Oracle

Visual Basic

Oracle DBA

Unix Administration

CADRe

Unix C,C++

The courses are offered in the following timings:

Part Time

Full Time

Crash

Note:

- Try using various Heading styles.
- Give a Back color to the page and try changing the back color.
- Try using various font sizes, colors and font faces.
- Use your own data and text formatting to make this page more attractive, informative and impressive.

URL, Protocols and Ports

Objective

This lecture provides an insight into the following:

- ❖ Uniform Resource Locators
- ❖ Communication between Client and Server using Protocols
- ❖ The Protocols widely in use - TCP/IP, HTTP and FTP
- ❖ Primary, Secondary and Master Name Servers
- ❖ The role of Ports
- ❖ Relative URLs and Absolute URLs
- ❖ Linking between and within documents

Lecture - 3

- 3.1 Snap Shot
- 3.2 URL
- 3.3 Protocol
- 3.4 Server Name
- 3.5 Port
- 3.6 Relative URLs and Absolute URLs
- 3.7 Linking to other HTML Documents
- 3.8 Linking inside the same document
- 3.9 Short Summary
- 3.10 Brain Storm

3.1 Snap Shot

A resource on the Internet can be located using the Uniform Resource Locator. The communication between the Client and Server is done using Protocols. The protocol used most often on the Internet is HTTP (Hypertext Transfer Protocol). It provides a way for a Web browser to access a Web server and request a hypermedia documents created using HTML. The above-mentioned concepts are discussed in detail in this lecture. It is often necessary to link many documents together for a detailed description of the subject. Also it is sometimes necessary to create links to certain parts within documents when the size of the document is relatively large. The creation of links is also discussed here.

3.2 Uniform Resource Locator

A resource is an object on the Internet or an Intranet that resides on a host system. Objects include directories and an assortment of file types, including text files, graphics, video, and audio. A URL is the address of an object that is normally typed in the Address field of a Web browser. The URL is basically a pointer to the location of an object. According to IETF RFC 1738, the syntax for a URL is

<scheme>:<scheme-dependent-information>

It is best to discuss this in terms that are more understandable. Scheme is any one of the Internet protocols, including HTTP, FTP, Gopher, News (USENET news), NNTP (Network News Transport Protocol), Telnet and WAIS (Wide Area Information Servers), among others. The following address uses an http scheme:

<http://www.ntresearch.com/papers/security.html>

First, http indicates that the HTTP protocol is to be used to access the site. The double slash indicates that the host name follows and the first single slash indicates that either a directory or a filename follows. In this case, "paper" is a directory, and "security.html" is a file in that directory that is displayed when this URL is entered in a Web browser as an address.

3.3 Protocol

The way, by which the communication between *Server* and *Client* takes place, is via protocols. A protocol is a set of rules by which the client knows how to pass request for data to the server. Likewise, the server knows how to send responses back to the calling client. Protocols are useful because they don't rely on platform specifics. A client doesn't need to know the operating system of the *Server* and vice versa.

Transmission Control Protocol/Internet Protocol (TCP/IP) is an industry standard suite of protocols providing communication in a heterogeneous environment. In addition, TCP/IP provides a routable, enterprise networking protocol and access to the worldwide Internet and its resources.

It has become the standard protocol used for interoperability among many different types of computers. This interoperability is one of the primary advantages to TCP/IP. Because of its popularity, TCP/IP has become the de facto standard for internetworking.

The protocol used most often on the Internet is HTTP (Hypertext Transfer Protocol). It provides a way for a Web browser to access a Web server and request a hypermedia documents created using HTML.

According to RFC (Request for Comment) 1945, “the Hypertext Transfer Protocol is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol, which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands).

HTTP is the command and control protocol that sets up communication between a client and a server and passes commands between the two systems.

Another widely used protocol is FTP (File Transfer Protocol). FTP is an Internet file transfer service that operates on the Internet and over TCP/IP (Transfer Control Protocol / Internet Protocol). FTP is basically a client/server protocol in which a system running the FTP server accepts commands from a system running an FTP client.

The service allows users to send commands to the server for uploading and downloading files. FTP operates among heterogeneous systems and allows users on one system to interact with another type of system without regard for the operating system in place.

3.4 Server Name

DNS servers store information about the domain name space. These servers are called name servers. Name servers generally are responsible for one or more zones and are said to have authority for those zones. (A zone is a physical file in the DNS composed of database records that defines a group of domains). When you configure a DNS name server, you must specify the names of all the DNS name servers in the same domain.

Primary, Secondary and Master Name Servers

The primary name server obtains zone data from local files. Changes to a zone, such as adding domains or hosts, are done at the primary name server level. A secondary name server obtains the data for its zones from another name server across the network that has authority for that zone. This is called a zone transfer. It should be

noted that a particular name server may be a primary name server for certain zones and a secondary name server for other zones. The source of zone information for a secondary name server in a DNS hierarchy is referred to as a master name server. A master name server can be either a primary or a secondary name server for the requested zone. When a secondary name server starts up, it contacts its master name server and initiates a zone transfer with that server.

3.5 Port

A computer on a network that offers services to users typically runs a number of different processes. In the TCP/IP network environment, these services are available at ports. When a computer connects with another computer to access a particular service, an end-to-end connection is established and a socket is set up at each end of the connection. The socket is created at a particular port number, depending on the application. You can think of a socket as being like the telephones at either end of a phone call and the port as being like a phone number.

The most common services are available at predefined port numbers that serve as unique identifiers for those services. The assignment of port numbers to particular services is not strictly controlled but is commonly followed throughout the industry. For example, the HTTP (Web) server uses port 80.

3.6 Relative URLs and Absolute URLs

A *Uniform Resource Locator*, commonly abbreviated URL, specifies the exact location of a resource (usually a file) on the Internet. There are two different types of URLs: absolute and relative.

Absolute URLs

Absolute URLs use a complete Internet address to give the location of a resource. To do this, they specify the transfer protocol first, then the particular server on which the file is located, and finally the actual path to the file on the server. The following URL, for example, would be interpreted by a Web browser as an instruction to use the Hypertext Transfer Protocol (http) to go to the Macmillan Computer Publishing Web server (www.mcp.com) and retrieve the home page from the Sams Publishing Company Directory (/mcp_publishers/sams/ Index.html):

`http:// www.mcp.com /mcp_publishers/sams/index.html`

Relative URLs

Relative URLs (sometime called “partial” URLs) are used for accessing files when the full Internet address is unnecessary. This is the case with most Web sites, where most hyperlinks point to documents and other files located on the same site and,

often, in the same directory on that site. Relative URLs are so called because the address is taken to be relative to the URL of the document in which it is embedded (called the “base document”). When relative URLs are accessed, the base part of the URL is automatically concatenated to the relative URL by the browser (or other client agent), so it doesn’t have to be explicitly identified.

Say a Web site <http://something.com/index.html> has been browsed. Although that page could indeed be linked to another page on the same site with the absolute URL of <http://something.com/nextpage.html>, a more economical method is simply linking with the relative URL of `nextpage.html`.

The browser strips everything to the right of the last / in the current document’s address and adds the relative URL to the end of the resulting base URL. In this case, it would remove the `index.html` portion of the absolute URL <http://something.com/index.html> to get a base URL of <http://something.com/> and replace the removed portion with the relative URL `nextpage.html`. The result is the full URL of

<http://something.com/nextpage.html>. The browser can then find the correct page just as though the absolute URL of the second page had been coded into the calling Web page.

3.7 Linking to other HTML Documents

In theory, every hyperlink has two components: the source anchor and the destination anchor. For instance, in the link denoted by ``, the destination of the link is the southindia Web site. The source end of the link is whatever HTML document contains this code. All hyperlinks are two-way streets; for a link between two HTML pages, the source and destination ends are equally available by clicking the Back and Forward buttons of a Web browser.

The ANCHOR element: <A>

The ANCHOR element, which appears only in the body of an HTML page, is generally rendered in a special way based on its usage. If an anchor is used to turn text into a hyperlink, the text usually colored blue and underlined.

Syntax

```
<A HREF = URL>Content</A>
```

The HREF (hypertext reference) attribute gets a *URL* to point to an another network resource. The URL can be fully qualified absolute address or a relative URL and must be enclosed in quotation marks. Any of the following are valid hyperlinks:

Click `here` to go to south India.

`here`

The following listing uses three hyperlinks, which appears as in Figure 3.1

```
<HTML>
<HEAD>
<TITLE>
India
</TITLE>
</HEAD>
<BODY background="C:\Raguram\Images\backgrd.gif">
<H1 ALIGN="Center">INDIA</H1> <HR>
```

Click any one of the following States to jump to their corresponding page.

```
<BR><BR>
<A HREF="Kerala.htm">Kerala</A><BR>
<A HREF="Karnataka.htm">Karnataka</A><BR>
<A HREF="TamilNadu.htm">Tamil Nadu</A>
</BODY>
</HTML>
```

Note that URL specified the all the three anchors are just file name. This means that these HTML files will be found in the same directory, in which the source HTML file is located.



Figure 3.1 Appearance of Hypertexts

When the mouse button is over any one of the hypertext, the mouse pointer will turn as hand pointer and clicking the links will cause a jump to its corresponding Web page.

3.8 Linking Inside the Same Document

When we click a hypertext link, the linked document loads into our browser, starting at the top of the document. However, we can target our links to specific points within a document, by setting up named anchors with the NAME attribute of the <A> tag. For example, if the linked document is long and we want to save users from having to do a lot of scrolling, we set up named anchors at the start of each major section of the document. Then we provide a link to each major section of the long document, instead of a single link that always send users to the top of the document. if the long document consists of four major sections, we can set up a named anchor as:

```
<A NAME = "section_three"> <H2> Section 3</H2></A>
```

This makes the level 2 heading viz., Section 3, into a named anchor that can be targeted by a hypertext link. To set up a link that targets this anchor, we place a pound sign (#) and the anchor's name after the long document's URL.

```
<A HREF = "longdoc.html#section_three"> Section 3 </A>
```

Here clicking the hypertext, Section 3, instructs the browser to load the file longdoc.html and begin presenting material in the file, starting at the anchor with the name, section_three.

3.8 Short Summary

- ◆ A resource is an object on the Internet or an Intranet that resides on a host system.
- ◆ The protocol used most often on the Internet is HTTP (Hypertext Transfer Protocol).
- ◆ A protocol is a set of rules by which the client knows how to pass request for data to the server.
- ◆ HTTP is the command and control protocol that sets up communication between a client and a server and passes commands between the two systems.
- ◆ FTP is basically a client/server protocol in which a system running the FTP server accepts commands from a system running an FTP client.
- ◆ The most common services are available at predefined port numbers that serve as unique identifiers for those services.
- ◆ A *Uniform Resource Locator*, commonly abbreviated URL, specifies the exact location of a resource (usually a file) on the Internet.

- ◆ Every hyperlink has two components: the source anchor and the destination anchor.

3.10 Brain Storm

1. State the types of URL and explain them
2. What do protocols do?
3. What is zone and zone transfer?
4. Explain the role of Name servers
5. What are ports and how do we assign port numbers
6. What are the unique features of the anchor <A> Tag?
7. How is linking within the same document different from linking to other HTML documents?
8. In what best terms can you define HREF?

Lab Unit (2 Real Time Hours)

This Exercise will make you understand completely about Links to other documents and within the same document. The web page you are about to create is a brief article on "Practicing Knowledge Management"

Heading – Practicing Knowledge Management

The contents -

When people talk about knowledge management (KM), it devolves into highly abstract and a philosophical statements. But there is a real world of KM – a world of budgets, deadlines, office politics and organizational leadership. KM projects are attempts to make practical use of knowledge to accomplish some organizational objective through the structuring of people

APPLYING KNOWLEDGE

The above item should contain a link to the latter part of the document, which discusses this in detail titled same as the above.

In order to understand how knowledge is really being managed in companies today, 31 different KM project in 20 different firms were studied. In most companies, only one project was addressed.

TYPES OF KM PROJECTS

KNOWLEDGE ACCESS AND TRANSFER

SUCCESS IN KM PROJECTS

SUCCESSFUL KNOWLEDGE PROJECT

The listed items are HYPERLINKS to other pages.

Create corresponding web page, which gives more detailed info about the selected link items.

For example if you click SUCCESSFUL KNOWLEDGE PROJECT link it takes you to.../ skproj.html. which contains the given information

After the projects had been classified, the most telling variables were identified and nine factors were found common throughout the successful projects. They are knowledge-oriented culture, technical and organizational infrastructure, senior management support, link to economic or industrial value, modicum of process orientation, clarity of vision and language, nontrivial motivational aids, some level of knowledge structure and multiple channels of knowledge transfer.

The APPLYING KNOWLEDGE item should be linked to the part of the HTML document given below.

APPLYING KNOWLEDGE:

Knowledge forms found in organizations include Knowledge recorded in memos, reports, presentations and articles, External knowledge, Structured internal knowledge, Informal internal knowledge and Tacit knowledge.

KM is making practical use of knowledge to accomplish any organizational objective through the structuring of people, technology and knowledge content.

KM projects are attempts to make practical use of knowledge to accomplish some organizational objective through the structuring of people, technology and knowledge content.

Design the Web Pages according to your requirement. Use all the possible TAG options.

Lecture 4

Internet Services

Objective

This lecture provides an insight into the following:

- ❖ Linking to other Internet Services
- ❖ Linking using File Transfer Protocol
- ❖ Linking using Gopher

Lecture - 4

- 4.1 Snap Shot
- 4.2 Linking to other Internet Services
- 4.3 Short Summary
- 4.4 Brain Storm

4.1 Snap Shot

Just as we have learnt to link our Web page with other web documents, it is often necessary to link to other Internet Services. This lecture discusses the methods by which this is achieved. As we have seen earlier, File Transfer Protocol (FTP) is an Internet file transfer service that operates on the Internet and over TCP/IP (Transfer Control Protocol / Internet Protocol). Gopher, an updated file transfer system is also introduced for purpose we will discuss here.

4.2 Linking to other Internet Services

We can link to any Internet service virtually, that is addressed with an URL, since HREF takes on the value of an URL. Among the best-known Internet Services available on the Internet, we shall now deal with FTP and Gopher

4.2.1 File Transfer Protocol (FTP)

FTP support is one method of supporting remote networks. It is a protocol, which allows simple file transfers of documents. There are FTP servers, which provide vast amounts of information stored as files. The data in these files cannot be accessed directly; rather the entire file must be transferred from the FTP server to the local servers. It is a file transfer program for the Transmission Control Protocol/Internet Protocol (TCP/IP) environments and is implemented at the Application layer of the OSI (Open Systems Interconnection) model.

The most common protocol used for sending files between computers is the File Transfer Protocol (FTP). FTP allows for transferring both text and binary files. Most Internet browsers such as Mosaic, Netscape, or the Microsoft Internet Explorer support FTP and use it behind the scenes when transferring files.

To set up an FTP download link on one of our pages, we set it up much like a link to another Web page, except that, we use an FTP URL rather than a Web URL

Example:

```
<A HREF = ftp://ftp.xyz.com/pub/program.exe>Self_extracting (3.2 Mb)</A>
```

Note that when the user clicks the hypertext "Self_extracting (3.2 Mb)", their browser downloads the file Program.exe to a local disk drive.

4.2.2 Gopher

While FTP works well for transferring files, it does not provide a good means of dealing with file systems spread over multiple computers. An updated file transfer system called Gopher was developed in response to this issue.

Gopher is a widely used tool on the Internet. It is a menu-based program that enables you to browse for information without having to know where the material is specifically located. It allows you to search lists of resources and then helps send the material to you. Gopher is one of the most comprehensive browser systems and is integrated to allow you access to other programs such as FTP and Telnet.

Gopher computers are linked together with distributed indexes into a searchable system called a “Gopherspace”. Gopherspaces typically offer a menu-driven system for access and are searchable with several search engines. We can set up links to Gopher sites on our Web pages by using the <A> container tag and the appropriate Gopher URL.

Example

Visit the Mother Gopher

4.3 Short Summary

- ◆ File Transfer Protocol (FTP) is an Internet file transfer service that operates on the Internet and over TCP/IP (Transfer Control Protocol / Internet Protocol).
- ◆ FTP allows for transferring both text and binary files.
- ◆ Gopher is a menu-based program that enables you to browse for information without having to know where the material is specifically located.
- ◆ Gopher computers are linked together with distributed indexes into a searchable system called a “Gopherspace”.

4.4 Brain Storm

1. Explain the File Transfer Systems.
2. Write a note on “Gopherspace”

Lab Unit (2 Real Time Hours)

This session deals with connecting with an FTP server and getting friendlier with the FTP commands

To start an FTP session and connect to an FTP server, start the command prompt and then type:

```
ftp xxx.xxx.xxx.xxx
```

(where xxx.xxx.xxx.xxx is a valid IP address), and then press ENTER. You can also use a fully qualified domain name (FDQN).

Use the IP address 127.0.0.1 to test TCP/IP on your computer. This address is known as the loopback address for your computer. The loopback address uses the loopback drivers to reroute outgoing packets back to the source computer. The loopback drivers enable the packets to bypass the network adapter card completely and be returned directly to the computer that is performing the test.

You will be prompted to log on with a user name. Log on as anonymous and when prompted for a password and then press ENTER. An ftp> prompt appears.

Logon and password information during a FTP session is transmitted unencrypted over the Internet to the FTP server.

From this prompt, you may enter various FTP commands.

You can also use a Web browser to view FTP sites. However, with most Web browsers you will not be able to use the put command to copy files to the server.

FTP Commands

You must use FTP commands to perform FTP actions. To copy a file from the server computer, at the FTP prompt you would type get filename.

The command copies the specified file from the server to the client.

Some of the common FTP commands include:

- ◆ ! - Returns to the Microsoft MS-DOS shell. FTP is still active, so type exit to return to the FTP prompt.
- ◆ !command - Executes an MS-DOS command on the local computer from the FTP session
- ◆ bye - Ends the FTP session with the remote computer and exits FTP
- ◆ delete - Deletes files on the remote computer (it requires appropriate permissions)
- ◆ dir - Lists the remote directory's file to your computer.
- ◆ help - Displays descriptions for FTP commands
- ◆ put - copies a file from your computer to the remote computer (it requires appropriate permissions)
- ◆ mkdir - Allows you to create a directory on the remote computer (it requires appropriate permissions)

Lecture 5

Lists in HTML

Objective

This lecture provides an insight into the following:

- ❖ Use of Lists in our HTML documents
- ❖ How to display text in lists
- ❖ Using Ordered Lists to organize data

Lecture - 5

- 5.1 Snap Shot
- 5.2 Lists in HTML
- 5.3 Ordered Lists
- 5.4 Using Ordered Lists
- 5.5 Short Summary
- 5.6 Brain Storm

5.1 Snap Shot

If presenting information in Web pages precisely and clearly is an art, organizing them elegantly is a yet another art. HTML provides Lists to organize the information and tables to organize the data.

5.2 Lists in HTML

Displaying text in lists

Lists are everywhere. On the refrigerator for groceries, on a scratch pad for to-do lists, in the front of books as a table of contents, in lists of directions for accomplishing tasks. Lists are one of the most natural ways to organize information.

HTML has a special set of tags just for displaying lists with a host of special attributes to give greater control over their appearance.

At the most basic level, lists are divided into two categories:

- ❖ **Ordered Lists** - These lists are typically used to indicate a sequence of events or priorities. They're also used to specifically identify sections and relationships when creating outlines.
- ❖ **Unordered Lists** - These lists are typically used to display a group of items that are somehow related, but necessarily in a hierarchical fashion. Three special HTML subsets of unordered lists are illustrated later lectures - definitions, directories, and menus.

The next sections look at each type of list and the ways to customize them to according to specific circumstance.

5.3 Ordered (or Numbered) Lists:

A list is defined by its opening and closing tags. For ordered list, they are and . However mere opening and closing tags are not enough. For example, the following listing, which is a list encased with the ordered list tags, is viewed on a browser as in Figure 5.1

```
<HTML>
<HEAD>
<TITLE>Basic CPR</TITLE>
</HEAD>

<BODY>
Basic CPR
<OL>
```

```
Use Body Substance Isolation Precautions
Determine Unresponsiveness
Open Airway
Look, Listen and Feel for Breathing
Ventilate Twice
Check for Pulse
If No Pulse, Begin Compressions
</OL>

</BODY>
</HTML>
```

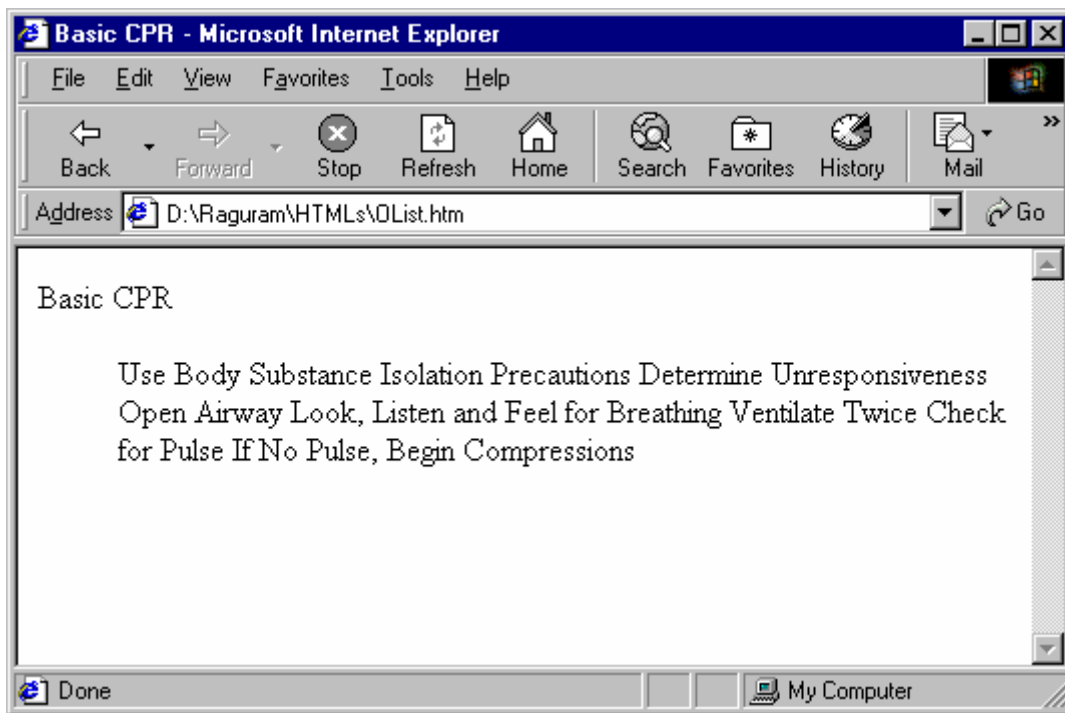


Figure 5.1 A list of Information that doesn't result in an HTML list.

How come the list in Figure 5.1 doesn't have numbers? Remember that HTML doesn't recognize line breaks unless it's told explicitly where they occur with the <P> or
 tags. The same principle applies to list, although a different tag is used to tell the browser where each specific item begins. This is the list item tag: .

With the and tags in hand, the basic syntax for an ordered list is as follows:

```
<OL>
<LI>ListItem1
<LI>ListItem2
<LI>ListItem3
...
<LI>ListItemN
```

```
</OL>
```

ListItem is each separate item in the list. The following listing is same as the proceeding listing except that the tag is added.

```
<HTML>
<HEAD>
<TITLE>Basic CPR</TITLE>
</HEAD>
<BODY>
Basic CPR
<OL>
<LI>Use Body Substance Isolation Precautions
<LI>Determine Unresponsiveness
<LI>Open Airway
<LI>Look, Listen and Feel for Breathing
<LI>Ventilate Twice
<LI>Check for Pulse
<LI>If No Pulse, Begin Compressions
</OL>
</BODY>
</HTML>
```

This listing produces the document as in Figure 5.2

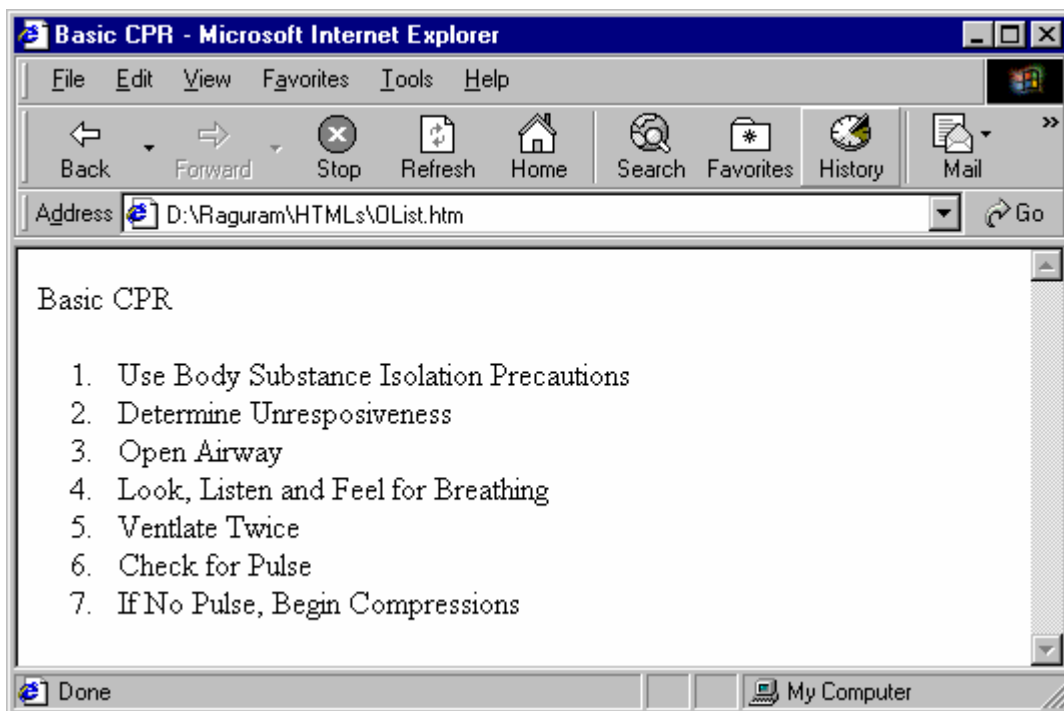


Figure 5.2 Lists with Intends and numbers.

The line break tags - <P> and
 - are also allowed within the body of a list to further control its appearance and formatting, as given in the following listings.

```
<HTML>
<HEAD>
<TITLE>Basic CPR</TITLE>
</HEAD>
<BODY>
Basic CPR
<OL>
<LI>Use Body Substance Isolation Precautions <P>
If available, be sure to use latex or vinyl gloves,
Goggles and a barrier device with one-way valve. <BR>
<B>BSI is crucial to your safety and the safety of your
patient.</B>
<LI>Determine Unresponsiveness
<LI>Open Airway
<LI>Look, Listen and Feel for Breathing
<LI>Ventilate Twice
```

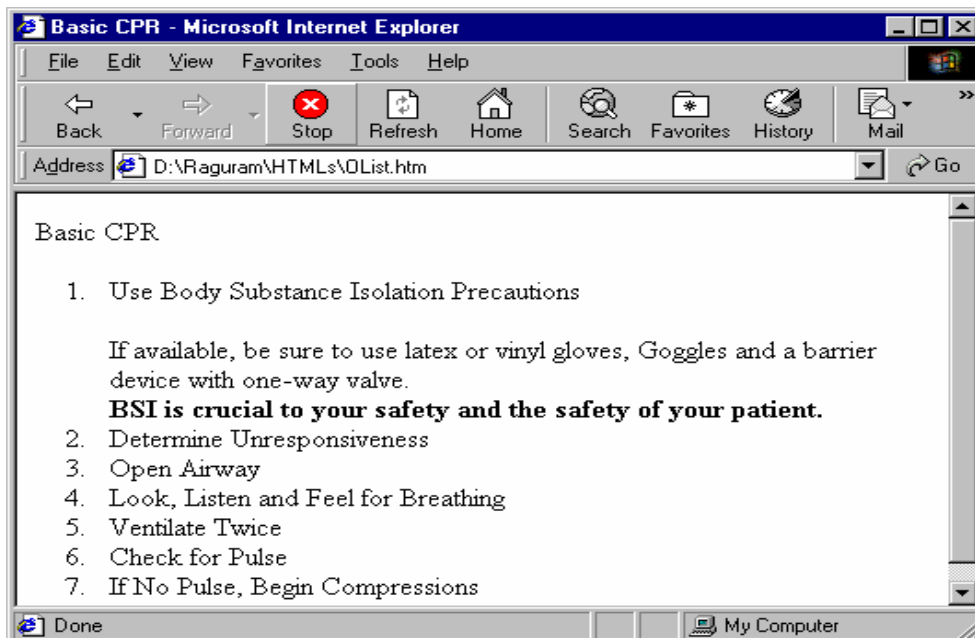


Fig
re 5.3 List with additional text.

```
<LI>Check for Pulse
<LI>If No Pulse, Begin Compressions
```

```
</OL>
</BODY>
</HTML>
```

When viewed on a browser (Figure 5.3), the tags affecting line breaks don't affect the line numbering. When the list begins with the tag, only an tag causes the browser to start a new line with the next number.

5.4 Using Ordered Lists

Where to start a list

Sometimes it's necessary to interrupt a list for some explanatory text or other material where indenting is not needed. Look at the following listing, which shows two ordered lists interrupted by some explanatory text, and its corresponding Figure 5.4.

```
<HTML>
<HEAD>
<TITLE>Basic CPR</TITLE>
</HEAD>

<BODY>
<H2>Basic CPR</H2>
<H3>Airway and Breathing</H3>
<OL>
<LI>Use Body Substance Isolation Precautions <P>
If available, be sure to use latex or vinyl gloves,
Goggles and a barrier device with one-way valve. <BR>
<B>BSI is crucial to your safety and the safety of your
patient.</B>
<LI>Determine Unresponsiveness
<LI>Open Airway
<LI>Look, Listen and Feel for Breathing
<LI>Ventilate Twice

</OL>

<P> Look for chest rise and listen for exhalation while ventilating. If no air enters,
reposition the airway and try again. If the airway is blocked, attempt to clear it using
appropriate airway obstruction maneuvers. If you can't get air into the patient,
circulation won't help. If the airway won't clear, you will remain in steps 3 through
5 until you can successfully ventilate the patient, or they are transported to a
hospital.</P>

<H3>Circulation</H3>

<OL>
```



```

<LI>Check for Pulse

<LI>If No Pulse, Begin Compressions

</OL>

</BODY>

</HTML>

```

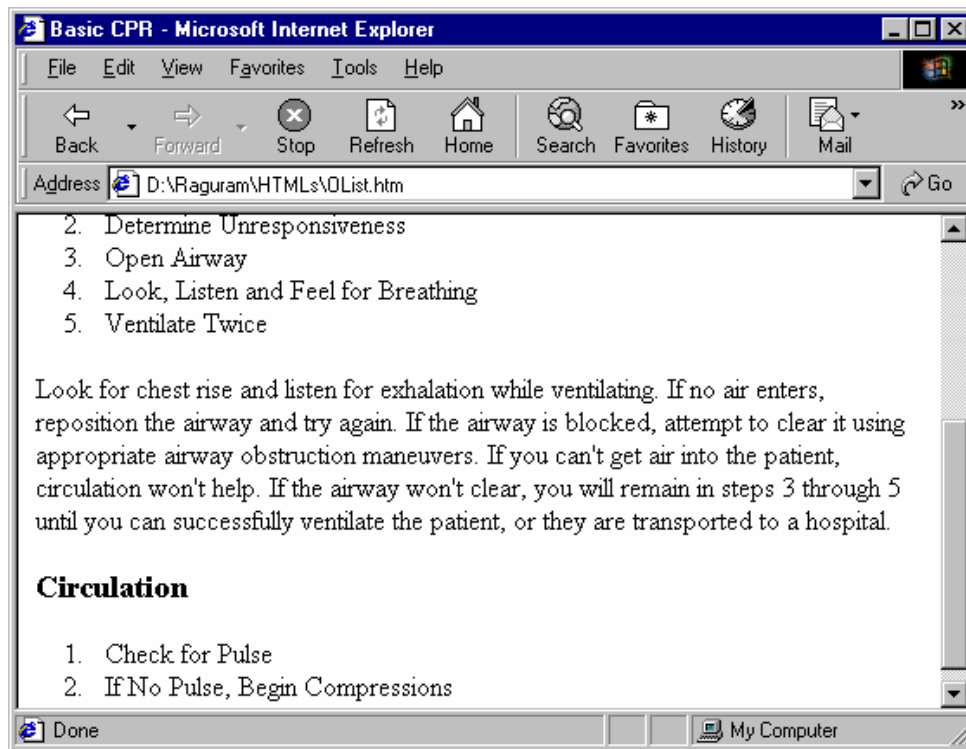


Figure 5.4 Appearance of two lists in which the numbers are not consecutive.

In the above Figure, list numbering isn't consecutive between individual lists. A new list starts the numbering process over from the beginning.

Two sets of tags are used – one for the first half of CPR (airway and breathing), and another for the second half (circulation). Because each portion of the list is contained within its own pair of ordered list tags, the browser treats each portion as a separate list and begins numbering from 1 for the second list. The browser doesn't know it's really a continuation of the first list.

To work around this problem, the START attribute of the tag is used. This enables to start numbering from anywhere. This attribute can be used to override default list numbering when a list broken into multiple parts.

In the above listing make the following change (indicated in bold).

```

<H3>Circulation</H3>

```

`<OL START = 6>`

Now view the same listing in the browser. The “Check for Pulse” item will begin with the number 6.

Using the start attribute makes it possible to create lists that are interrupted by blocks of text or other material and then resumed with appropriate numbering in a subsequent list.

This is the syntax for start:

`<OL START=number>`

In this attribute, number is any integer from -2147483648 to 2147483647. Be sure not to use commas in numbers; browsers and other user agents do not recognize them.

5.5 Short Summary

- ◆ HTML has a special set of tags just for displaying lists with a host of special attributes to give greater control over their appearance.
- ◆ Ordered Lists are typically used to indicate a sequence of events or priorities.
- ◆ Unordered Lists are typically used to display a group of items that are somehow related, but necessarily in a hierarchical fashion.
- ◆ The tags essential for an ordered list are `` tags and ``, the list item tag.
- ◆ Using the start attribute makes it possible to create lists that are interrupted by blocks of text or other material and then resumed with appropriate numbering in a subsequent list.

5.6 Brain Storm

1. Select the listing tags from the following
 - a. Ordered
 - b. Unordered
 - c. Heading
 - d. Menu & Dir
2. The type attribute is supported by Ordered List
 - a. True
 - b. False
3. Is it possible to give `START = 27` in an ordered list? If possible write down the numbered list

4. Write the output of the following tag?

```
<OL TYPE = A START = 27>  
<Li> Oracle  
<Li> VB  
</OL>
```

Lab Unit (2 Real Time Hours)

On completion of this session you will get familiar with using ordered lists in your web page.

Heading – Strategies for the Enterprise

Content

As organizations increasingly learn that cash on power of knowledge is critical to achieving competitive differentiation, Knowledge Management has gained predominance.

There are two types of knowledge

1. Tacit

Tacit knowledge, as the phrase implies, is knowledge, which is clearly understood but may not be clearly articulated. Much of the knowledge held by individuals is tacit. Every knowledge directly referred to from books and journals may be tacit, for example, a professional has about a dozen reference books on a particular subject. In general we may define Tacit knowledge in terms of

1. Clearly understood
2. Experience
3. Simultaneous Interaction

2. Explicit

Explicit knowledge has well defined flow-charted methods to arrive at a desired output. The inputs are limited to a set of logically related activities, which are processed in sequence. Outcomes of previous steps are important to subsequent steps in the flow. Explicit knowledge may be described in terms of

1. Clearly Articulated
2. Rationality

3. Sequential

Having understood the characteristics of tacit and explicit knowledge, the KM architect needs to understand how knowledge transmission and transformation take place and what organizations need to do to facilitate this transformation.

1. TACIT TO TACIT
2. TACIT TO EXPLICIT
3. KEY ENABLERS
4. EXPLICIT TO EXPLICIT
5. EXPLICIT TO TACIT
6. CREATIVE CHAOS AND TENSION
7. REDUNDANCY
8. REQUISITE VARIETY

Lists in HTML

Objective

This lecture provides an insight into the following:

- ❖ Netscape Extensions to the tag
- ❖ Listing using unordered lists
- ❖ Different Types of Bullets in unordered lists
- ❖ Use of the over the deprecated <DIR> and <MENU>.
- ❖ The tag Create used to crate a glossary-style listing - <DL>
- ❖ Embedded combinations of all of the list types.

Lecture - 6

- 6.1 Snap Shot
- 6.2 Using Netscape Extensions
- 6.3 Unordered Lists
- 6.4 Directory Lists
- 6.5 Definition Lists
- 6.6 Combining List Types
- 6.7 Short Summary
- 6.8 Brain Storm

6.1 Snap Shot

When Browsers like Mosaic became available, the world has turned upside down. Then came Netscape. Netscape added features to its browser to enhance the presentation of an HTML document. These extensions are only viewable via a Netscape Browser. Representing a set of items that are somehow related to one another but don't necessarily need to follow a specific order are done using unordered lists. We also discuss the ways to represent menu and definitions in a web document.

6.2 Using Netscape Extensions

Netscape provides useful extensions to the tag supported by their own Netscape Navigator as well as Microsoft Internet Explorer. In the early days of HTML, there was only one type of ordered list, with numbers beginning at 1 and ending wherever the list stopped. Then, as people began stretching the use of lists to things such as online books, they began wanting to use something other than numbers. The TYPE attribute was developed to meet this need; here is its syntax:

```
<OL TYPE=numberingSystem>
```

In this attribute, *numberingSystem* is one of five characters: 1, A, a, I, or i. Examples of the latter four numbering units are illustrated in the following Table.

Value	Style	Example
1	Arabic	1,2,3...
A	Uppercase alpha	A, B, C...
a	Lowercase alpha	A,b,c...
I	Uppercase Roman	I, II, III...
i	Lowercase Roman	i, ii, iii...

The ability to use something other than numbers leads to a useful feature of ordered lists: *nested lists*, which are lists within lists that can extend several levels. To create a nested list, include a new set of ordered list tags with the current list tags as shown in the following listings. The browser begins a new list for the new tags while remembering where the parent list left off after the nested list ends (Figure 6.1).

```
<HTML>
<HEAD>
<TITLE>Patient Assessment</TITLE>
</HEAD>

<BODY>
<!--BEGIN MAIN LIST -->
<OL TYPE=A>
<LI>Safety Considerations
```

```

        <!--BEGIN SUB LIST -->
<OL TYPE = 1>
    <LI>Body substance isolation
    <LI>Scene safety
    <LI>Initial size-up
</OL>

<LI>Initial Patient Assessment
    <!--BEGIN SUB LIST 2-->
<OL TYPE = 1>
    <LI>General Impression
    <LI>Unresponsiveness
    <!--BEGIN SUB LIST 1 OF SUB LIST 2 -->
    <OL TYPE=i>
    <LI>Alert to person, place and time
    <LI>Verbal response to audible stimuli
        <LI>Pain evokes verbal or physical response
        <LI>Unresponsive to all stimuli
            </OL>
        </OL>
    <LI> Patient Critical Needs
    <!-- BEGIN SUB LIST 3 -->
<OL TYPE=1>
<LI>Airway
<LI>Breathing
<!--BEGIN SUB LIST 1 OF SUB LIST 3-->
<OL TYPE=1>
    <LI>Use Oxygen if indicated
    <LI>Verbal response to audible stimuli
<LI>Consider use of assisting with bag
    valve mask
</OL>
    <LI>Circulation
    <LI>Bleeding
        </OL>
    <!-- END MAIN LIST -->
<OL>
</BODY>
</HTML>

```

One of the main problems with nesting lists is the confusion they can generate. It's easy to lose track, of which list is which, and what's subordinate to what after the second or third embedded set of items. Remember that extra leading and trailing spaces are stripped from HTML along with carriage returns. Feel free to use extra returns and indenting with comment tags to make the page's source code easier to read and edit.

It should be noted that the START attribute always has an integer value regardless of the type specified. For example, to begin an ordered list with the capital letter D, the opening tag is specified as <OL TYPE=A START=4>. The browser will translate the starting position into the appropriate letter.

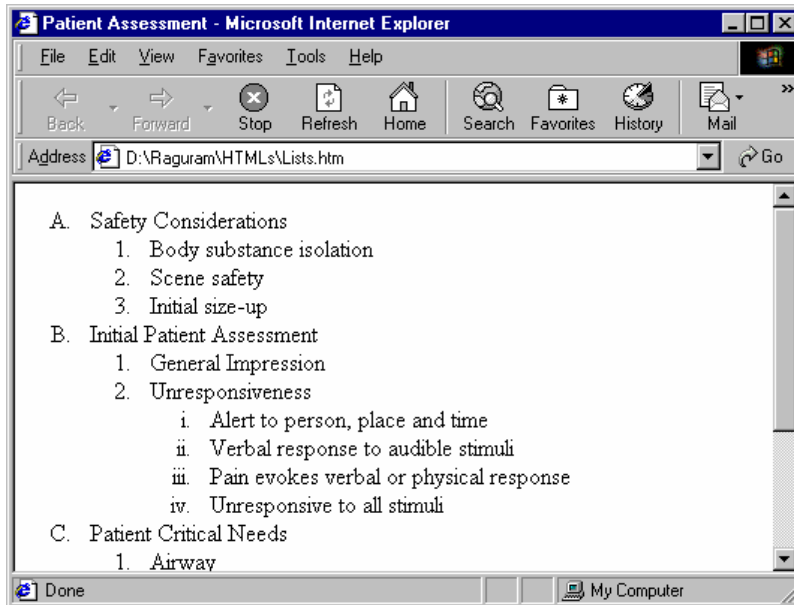


Figure 6.1 Different types of lists in nested fashion.

So far, we have defined the types and variations of ordered lists. The following sections explain its half-sister – the unordered list.

6.3 Unordered Lists:

Unordered lists are typically used to represent a set of items that are somehow related to one another but don't necessarily need to follow a specific order. The syntax is similar to that of the ordered list.

```
<UL>
<LI>ListItem1
<LI>ListItem2
<LI>ListItem3
....
<LI>ListItemn
</UL>
```

ListItem is separate item in the list as given in the following listing.

```
<HTML>
<HEAD>
```

```
<TITLE>Lives</TITLE>
</HEAD>
<BODY>
<H2>Lives Encyclopedia</H2>
<UL>
<LI> Animals
<LI> Reptiles
<LI> Pisces
</UL>
</BODY>
</HTML>
```

This listing produces document as in Figure 6.2

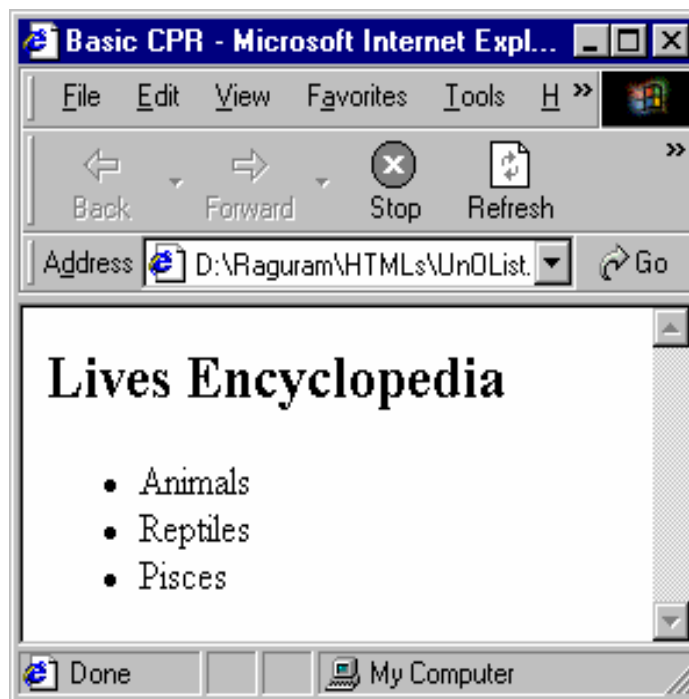


Figure 6.2 Unordered List.

The above shows that the unordered list is typically represented the same way as an ordered list, except bullets are used instead of numbers.

The actual appearance of the bullet varies from browser to browser. Internet Explorer uses small text bullets, and others use graphical bullet.

Numbering is obviously not an issue for an unordered list. However, HTML offers some additional choices for the default bullet appearance with the TYPE attribute.

Using different types of Bullets

Three basic types of bullets are supported by HTML, although not all browsers support all three. If a browser doesn't recognize the attribute, the default bullet representation is used. The TYPE attribute's syntax is given below:

```
<UL TYPE=bulletType>
```

In this attribute, *bulletType* is one of three values: **circle**, **square**, or **disc**. These values are used in the following listing.

```
<HTML>
<HEAD>
<TITLE>Lives</TITLE>
</HEAD>

<BODY>
<H2>Lives Encyclopedia</H2>
<UL TYPE= "Circle" >
<LI> Earth Creatures
    <UL TYPE= "Disc" >
    <LI> Animals
        <UL TYPE= "Square" >
        <LI> Jackal
        <LI> Leopard
        <LI> Lion
        </UL>
    <LI> Reptiles
        <UL TYPE= "Square" >
        <LI> Ant
        <LI> Cobra
        <LI> Lizard
        </UL>
    </UL>
</UL>
<LI> Water Creatures
    <UL TYPE= "Disc" >
    <LI> Whale
    <LI> alligator
    </UL>
</UL>
</BODY>
</HTML>
```

This listing produces the document as in Figure 6.3.

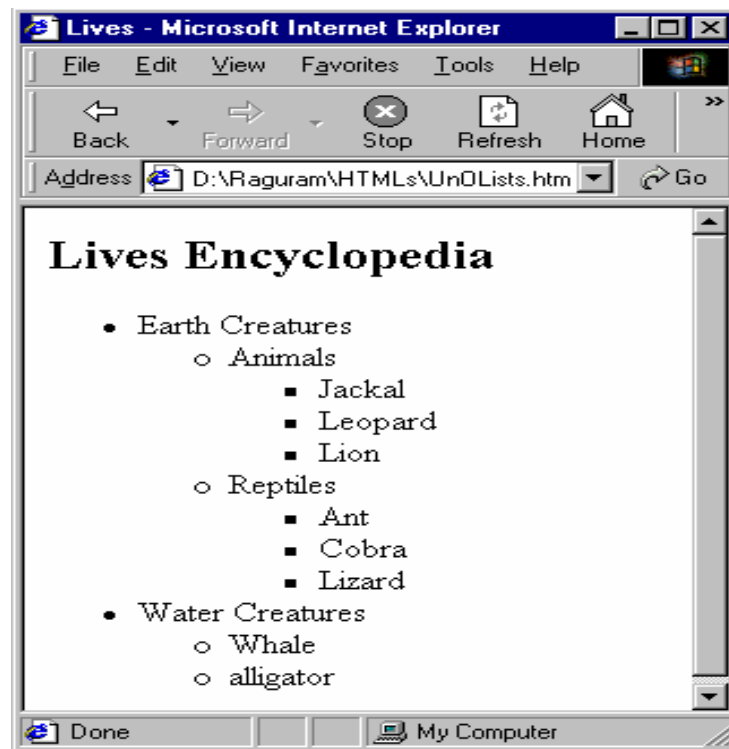


Figure 6.3 Unordered list with different styles of bullets.

Also like the ordered list, an unordered list can contain other lists within a list. This nesting helps show the relationship among items, even though there isn't an underlying hierarchy.

6.4 Directory Lists

The last set of list tags supported by HTML is `<DIR>` and `<MENU>`, originally designed for directory listings and user menus. However, their use has fallen by the wayside, and they are deprecated in the HTML 4 standard. Use of the `` is recommended over the deprecated `<DIR>` and `<MENU>`.

Both `<DIR>` and `<MENU>` use the `` tag to mark each separate item. The syntax for the two lists is the same as for an unordered list (which is part of the reason they are slated for eventual removal from the HTML standard):

```
<DIR> <!-- or MENU -->
<LI>ListItem1
<LI>ListItem2
....
<LI>ListItem3
</DIR> <!-- or /MENU -->
```

Both items are typically rendered as unordered lists.

6.5 Definition Lists

The last specialty list tag is `<DL>`, which stands for definition list. This tag is used to create a glossary-style listing, which is handy for items such as dictionary listings and Frequently Asked Question pages.

The `<DL>` tag is used similarly to the unordered list tag, except that it doesn't use the `` tag to mark its entries because a definition list requires two items for every entry: a term and its definition. Marking these two items is done with the corresponding `<DT>` and `<DD>` tags.

Syntax

```
<DL>
<DT>Term1<DD>Definition1
<DT>Term2<DD>Definition2
...
<DT>Term3<DD>Definition3
</DL>
```

In this tag, *Term* is the word requiring a definition and *Definition* is the block of text that supplies the definition.

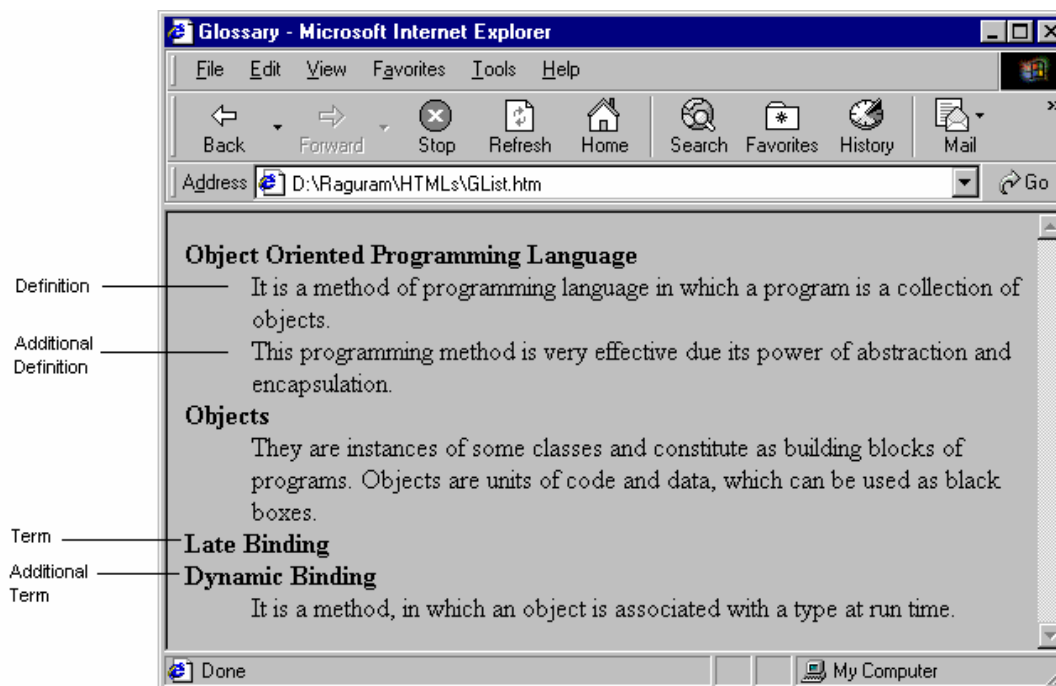


Figure 6.4 Typical glossary prepared using Definition List.

The `<DL>` tag's implementation can vary a bit from browser to browser, but a common method of displaying the content is shown in Figure 6.4.

Like the other list tags and styles, it's also possible to include additional physical formatting tags in the terms or definitions:

```
<DT><B>Term</B>
```

```
<DD><I>Definition</I>
```

A popular method of implementing the definition list is to give the term a bold style and use regular or italic type for the definition as in Figure 6.4.

All the major styles of creating and displaying lists are covered. Now it is time to take a look at the last two styles.

6.6 Combining List Types

There are applications, which use sublists of more than one type within a single list. For instance, we may have a numbered list that includes a list as one of the numbered elements. Instead of just creating an ordered sublists, which numbers each of its items, we might prefer to display an unordered list to differentiate the sublists (while avoiding ordering the information as well) HTML supports embedded combinations of all of the list types.

6.7 Short Summary

- ◆ Netscape provides useful extensions to the tag supported by their own Netscape Navigator as well as Microsoft Internet Explorer.
- ◆ *Nested lists* are lists within lists that can extend several levels
- ◆ Unordered lists are typically used to represent a set of items that are somehow related to one another but don't necessarily need to follow a specific order.
- ◆ Three basic types of bullets supported by HTML are circle, square and disc
- ◆ The last set of list tags supported by HTML is <DIR> and <MENU>, originally designed for directory listings and user menus.
- ◆ Use of the is recommended over the deprecated <DIR> and <MENU>.
- ◆ Definition list tag is used to create a glossary-style listing, which is handy for items such as dictionary listings and Frequently Asked Question pages.

6.8 Brain Storm

1. Select all the values for the type attribute of an unordered list
 - a. Square
 - b. Circle
 - c. Rectangle
 - d. Disc
2. Is it possible to give a start attribute for unordered list?
3. Write a note on Definition Lists.
4. How do you represent sublists of more than one type within a single list?

Lab Unit (2 Real Time Hours)

a. This session helps you to organize unordered lists as well as combined list types, which we often ought to use in our web pages.

The Heading is “New Wave in IT Careers”

The output screen should appear as follows:

As an established provider of e-business solutions to primarily the banking, finance and health care industries, we are looking for talented professional to fill several engineering and development positions to work on existing client projects and also to enhance our portfolio. We offer a high quadrant compensation package, stock options and excellent benefits for a person with right skills

- **Engineering**
 - **eCommerce Architect**
 - You will work with the latest Java technologies like EJB, JSP, CORBA and other component architectures
 - Masters degree in Computer Science or related discipline
 - Sound knowledge of OO Programming, Design Patterns and UML
 - 5+ years in total experience (4+ with Masters) and 2+ years in Java
 - **Sr. Software Engineer**
 - 4+ experience in Software Engineering
 - Must have good knowledge of Database Modeling tools like Erwin, S-Designer and relational databases Sybase & Oracle
 - Knowledge of database connectivity technologies (JDBC, ODBC)
- **Testing/QA**
 - **GUI Test Engineering**
 - 3+ working experience, with 1 of automation tools experience (SunTest, Seque tools, etc)
 - Programming experience in C/C++ or Java
 - Position requires UNIX & NT experience

b. This session deals with understanding definition lists. Your output screen should be similar to the given one.

Heading - Definitions

Object Oriented Programming Language

It is a method of programming language in which a program is a collection of objects. (Definition)

This programming method is very effective due to its power of abstraction and encapsulation (Additional Definition)

Objects

They are instances of some classes and constitute of building blocks of projects. Objects are units of code and data, which can be used as black boxes.

Late Binding (Term)

Dynamic Binding (Additional Term)

It is a method, in which object is associated with a type at runtime

Lecture 7

Graphics and Web Page

Objective

This lecture provides an insight into the following:

- ❖ Graphics in a Web page
- ❖ Advantages and disadvantages of Graphics in a Web page
- ❖ Image Formats and the browsers that support them
- ❖ Adding images to a Web page
- ❖ Using images as hyperlinks
- ❖ What exactly is an Image map
- ❖ Working with an image map

Lecture - 7

- 7.1 Snap Shot
- 7.2 Graphics and Web Pages
- 7.3 Image Formats and Browsers
- 7.4 Graphics and HTML Documents
- 7.5 Images and Hyperlink anchors
- 7.6 Image Maps
- 7.7 Short Summary
- 7.8 Brain Storm

Lab Unit

7.1 Snap Shot

An attractive feature in HTML is Graphics, which is by far one of the most popular elements used in the designing of Home pages. You can use graphics on your Web page to provide information, artwork, or pictures. Including pictures in your document is very easy. There are number of image formats available. When deciding which image format to use, you need to look at the type of image you are storing. These and other elements used to create clickable image maps are also discussed in this lecture.

7.2 Graphics and Web Pages

Graphics is by far one of the most popular elements used in the designing of Home pages. You can use graphics on your Web page to provide information, artwork, or pictures. In addition to being decorative, graphics can be useful, for example as navigational buttons. Using graphics is a great way to break up large amount of text. For example, it would more interesting for users to get information of a company's profit from a bar chart rather than from documents written in a number of paragraphs.

However it is important to take a note of the following. When you add graphics such as pictures, animations, and videos to your web, the size of your web will quickly increase. The size, quality, type and number of graphics are important considerations if your Internet service provider has limited the size of your web. This may be overcome by resizing or resampling a large picture and make it smaller or convert the picture to JPEG format.

Graphics can take a lot of time to download, especially for site visitors with slow Internet connections. This may be prevented by either resizing or resampling large pictures or by creating a thumbnail i.e., a small version of the picture that downloads quickly.

However the user has to decide putting graphics in an HTML document. Chances are, though that we will probably make some use of images in our home page, but not too much. Using only a few images is generally a good approach, but how much?

7.3 Image Formats and Browsers

You can add pictures with the following file formats: GIF (standard and animated), JPEG (standard and progressive), BMP (Windows and OS/2), TIFF, TGA, RAS, EPS, PCX, PNG, PCD (Kodak Photo CD), and WMF. The formats that are generally used for Web pages are GIF, JPEG, and PNG.

GIF (Graphics Interchange Format), formally known as GIF 87a, was originally designed by CompuServe. It is probably one of the most widely used and supported graphics format around. GIF is the only file format that can be viewed by almost every graphical browser around. You can add animated GIFs and videos to Web pages.

An animated GIF, which is a sequential display of GIF pictures, can be created in a graphics program, and you can find animated GIFs on the World Wide Web. Pictures in GIF format can contain up to 256 colors. One useful aspect of GIF pictures is that you can select one color to be transparent.

The JPEG (Joint Photographic Experts Group) format is commonly used for photo-realistic pictures containing thousands or millions of colors. JPEG pictures are useful because you can control the file compression by changing the picture quality. The lower you set the quality, the higher the file compression will be, and, as a result, the file size is decreased. JPEG is widely used by most people and is rapidly gaining more and more acceptance. Older graphical browsers do not support JPEG natively, but the newer ones do.

Netscape, a company that constantly changes the face of Web browsers, has introduced Progressive JPEG. Visually, this new file format appears to behave similar to Interlaced GIF format. Interlaced GIF format is one in which, as more of the file is downloaded, the picture becomes clearer. This capability is useful because it allows users who are not interested in a particular graphic to avoid downloading the whole image.

The PNG format is an alternative to GIF that supports transparency for pictures containing thousands or millions of colors. However, some Web browsers cannot display PNG pictures without a special plug-in.

Generally images in the format .GIF or .JPG are used in Web pages. Because they are compressed format and so can travel over the Internet quickly.

7.4 Graphics and HTML Documents

In order to create exciting Web pages and to provoke users to eagerly visit to the Pages again and again it must be attractive. Without impressive images this can't be attained. HTML provides a tag to add images to Web pages.

Syntax

<IMG

[ALIGN = TOP | MIDDLE | BOTTOM | LEFT | RIGHT]

[BORDER = pixels]

[HEIGHT = pixels]

SRC = path or URL of image to include

[WIDTH = pixels] >

The attributes are described below:

- ALIGN** This attribute controls the horizontal alignment of the image with respect to the page. The default value is LEFT. Only the Netscape and Internet Explorer implementations support the **ABSBOTTOM**, **ABSMIDDLE**, **BASELINE** and **TEXTTOP** attributes.
- BORDER** This attribute indicates the width in pixels of the border surrounding the image.
- HEIGHT** This attribute indicates the height in pixels of the image.
- SRC** This attribute indicates the URL of an image file to be displayed.
- WIDTH** This attribute indicates the width in pixels of the image.
- ALT** supplies a text-based alternative for the image for browsers that do not support graphics.

The following listing, which uses the tag produces a document as in Figure 7.1

```
<HTML>
<HEAD>

<TITLE>
InnerTrail
</TITLE>

</HEAD>

<BODY>

<CENTER>
<U>
<H2>Experience the Inner Pathway</H2>
</U>
<IMG Src="C:\Raguram\Images\Innerpath.jpg" ALT =
"alternative_text_description">
</CENTER>
```

```
</BODY>
```

```
</HTML>
```



Figure 2.9 Image in HTML document.

7.5 Images and Hyperlink anchors

Basically there are two classifications of Hyperlinks - Hypertext and Hypermedia. Text that forms a hyperlink are called as Hypertext and Images that forms the hyperlink are called as Hypermedia.

You can set a default hyperlink for a graphic, such as a picture, animated GIF, or video. When a site visitor clicks the graphic, the Web browser displays the destination of the hyperlink. For example, to create a button that displays your home page, add a button graphic to a page, and then set the default hyperlink for it to go to your home page.

All that's required to turn a image as hyperlink is to substitute the textual portion of the anchor element (<A>) with an tag as in the following.

```
<A href="C:\MS_jan\cow.htm">  
<IMG Src="Innerpath.jpg">  
</A>
```

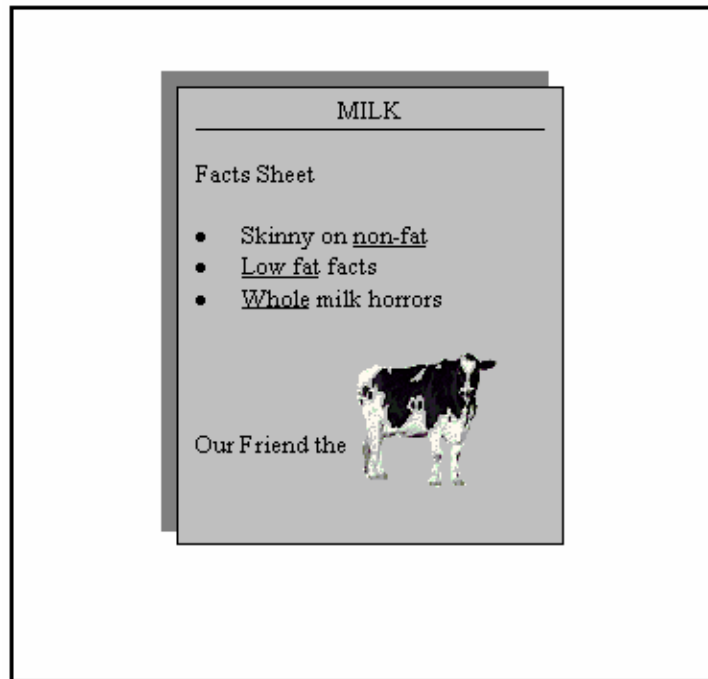


Figure 7.2 *Image as a hyperlink*

Clicking on the cow in the above figure takes us to a page on cows.

7.6 Imagemaps

Imagemaps have a universal appeal over textual menus, perhaps because human being has the ability to interpret visual images. Whatever the reason, imagemaps have taken over as the first choice for menu development on the World Wide Web.

What exactly is an imagemap? An imagemap is an extended version of an image hyperlink. Instead of having an image represent only one hyperlink, the image can represent several different links depending on where in the image is clicked.

A single-link image anchor poses no particular processing difficulty. An imagemap, with its multiple hyperlinks, requires a method for determining which link to follow. This is accomplished by tracking the position of the mouse pointer in relation to the upper-left corner of the image which has the x, y coordinates 0,0. The x coordinate increases from left to right, counting (in pixels) to the space occupied by the image. Likewise, the y coordinate increases from top to bottom to the size of the image (see Figure 7.3)

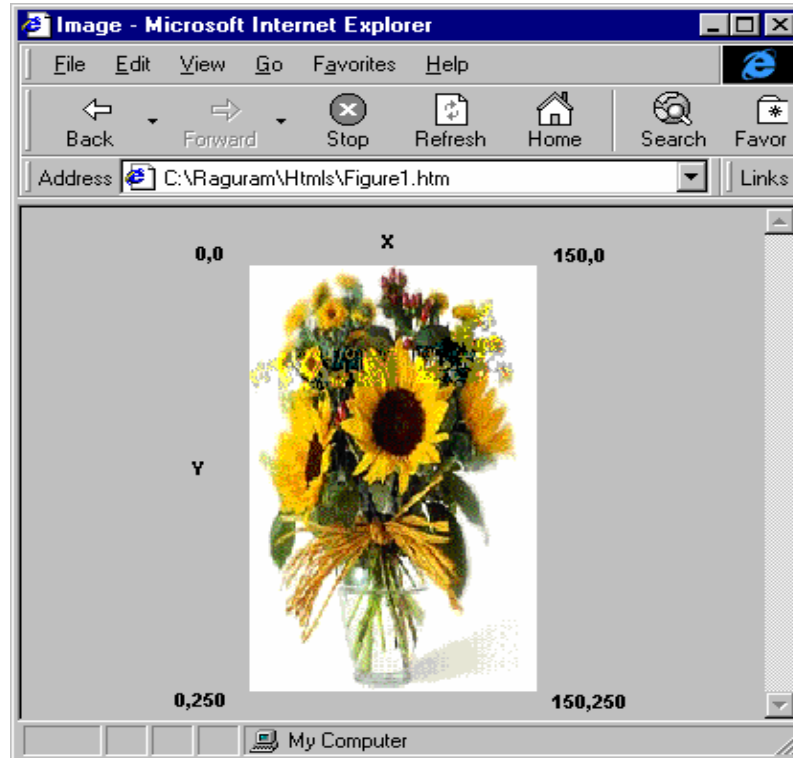


Figure 7.3 Image coordinates of an image 150 X 250.

Active areas (called *hot spots*) were originally limited to rectangular shapes. If the mouse pointer is in the area between 0,0 and 20, 10, a different hyperlink is activated than if the pointer is between 0,11 and 20,20. As imagemap technology has improved, circles and polygons have also been added to the kinds of shapes that are allowed for hyperlink hot spots.

Designing Imagemaps

The first thing is, of course, an image. The image used for the image map is inserted into the HTML page just like a normal image, but with one important difference: the USEMAP attribute. Here is an example of inserting an imagemap into an HTML page:

```
<IMG SRC= "Sky.gif" USEMAP= "#skymap">
```

The USEMAP attribute tells the browser that it's dealing with an imagemap and then directs the browser to where coordinate and URL information for this imagemap can be found. That named anchor will be a MAP element, which contains the same information as a text-based menu, plus a definition of the area on the imagemap that will activate them. The value of USEMAP must be same as the value of NAME in the MAP element.

MAP elements are contained within the body of the document and have only one attribute, NAME. The NAME attribute identifies the particular MAP element referenced with the USEMAP attribute. The AREA element is used within the MAP to delineate a hyperlink and the shape and coordinates of its corresponding hot spot in the image.

```
<AREA SHAPE = value COORDS = "shape_dependent" HREF = "hyperlink_URL">
```

There are three different shapes that can be defined with the SHAPE attribute: **rect**, **circle**, and **poly**. They refer to, respectively, rectangles, circles, and polygons. If the shape is not specified the attribute defaults to **rect**.

- ◆ Rectangle is the default shape for the AREA element. It requires two sets of coordinate pairs. The first pair gives the coordinates of the upper-left corner, and the second pair gives the coordinates of the lower-right corner:

```
COORDS= "upper_left_x, upper_left_y, lower_left_x, lower_left_y"
```

- ◆ Circles require one set of coordinate pairs, which gives the coordinates of the center point, and a third number gives the radius of the circle:

```
COORDS= "center_x, center_y, radius"
```

- ◆ Polygons are more complex shapes than either rectangles or circles and thus require a series of coordinate pairs that define points along the outline of the figure:

```
COORDS= "first_point_x, first_point_y, second_point_x, second_point_y
```

```
..., last_point_x, last_point_y"
```

If two different areas overlap and user clicks on the overlapping part, the one listed first in the MAP element is the link that will be activated. That's because the WEB browser reads the listing from the top down when looking for an AREA element that encompasses the coordinates the coordinates that were clicked; once it finds a match, it stops looking.

For example, look at the following Figure 7.4

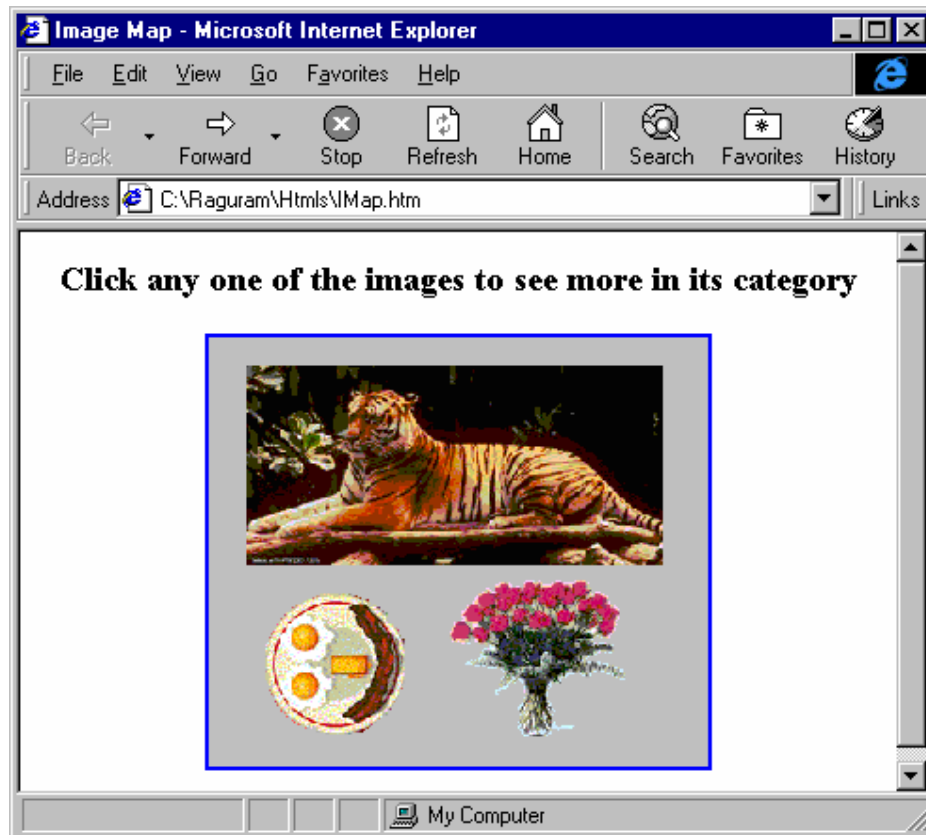


Figure 2.12 A image map.

The image in the above figure can be mapped for three areas - tiger (rectangular area), smiling face (circular area) and flower (polygonal area). To achieve the following code is used.

```
<IMG SRC="C:\Shared\Images\IMaps\Imap.bmp" USEMAP="#ImageMap">
<MAP NAME="ImageMap">
  <AREA SHAPE = "rect" COORDS = "20, 10, 250, 120"
    HREF = "C:\Raguram\Htms\India.htm">
  <AREA SHAPE=circle COORDS = "68,175,40"
    HREF="C:\Raguram\Htms\table.htm">
  <AREA SHAPE=poly COORDS = "160, 210, 125, 165, 130, 125, 200, 125,
    240,140" HREF="C:\Raguram\Htms\indianstates.htm">
</MAP>
```

The SHAPE attribute has a fourth possible value, but it's not actually a shape. That value is **default**, which establishes a URL that will be activated if a user clicks inside the -imagemap but outside any of the areas.

7.7 Short Summary

- ◆ Using graphics is a great way to break up large amount of text.
- ◆ Creating a thumbnail view or a small version of the picture allows graphics to downloads quickly.
- ◆ GIF is the only file format that can be viewed by almost every graphical browser around.
- ◆ An animated GIF is a sequential display of GIF pictures in a graphics program
- ◆ Text that forms a hyperlinks are called as Hypertext and Images that forms the hyperlinks are called as Hypermedia.
- ◆ An imagemap, an extended version of an image hyperlink can represent several different links depending on where in the image is clicked.
- ◆ If two different areas in an image overlap and user clicks on the overlapping part, the one listed first in the MAP element is the link that will be activated.

7.8 Brain Storm

1. How to embed an image into an HTML document and turn it as a hyperlink?
2. What are the attributes of an Image Tag?
3. What is an Image Map? How to map an image?
4. What are the shapes available in an Image Map?
5. Is it possible to have more than one image map in an HTML document?

Lab Unit (2 Real Time Hours)

Design a web page that should provide information about your career plan and with an image at bottom of the page. An example image is given for you below:



This image should enable for navigation to any other two pages you have already created.

An image may be downloaded from a web site as follows

1. Right click the mouse over the image
2. Select “Save picture as”
3. Give a file name. It will automatically be stored as a .gif or .jpeg file.

HTML Tables & Frames

Objective

This lecture provides an insight into the following:

- ❖ Basic structure of Tables
- ❖ Aligning Table Elements within the cell
- ❖ Spanning of Rows and Columns
- ❖ Enhancements by the Netscape Navigator to Tables
- ❖ Using Frames in a Web Page

Lecture - 8

- 8.1 Snap Shot
- 8.2 HTML Tables
- 8.3 Aligning Table Elements
- 8.4 Row and Column Spanning
- 8.5 Netscape Table Enhancements
- 8.6 Frames in HTML
- 8.7 Frameset Container
- 8.8 Short Summary
- 8.9 Brain Storm

Lab Unit

8.1 Snap Shot

Tables can be used for more than just displaying a table of data. Table can also be used as a formatting tool. The data in a table can be text or images. Because many HTML documents rely less on text than their printed equivalents, Web-based tables have become an important way to structure documents. Frames allow the programmer to divide the browser window into sections, each section acting independently of each other. These two concepts are discussed in this lecture

8.2 HTML Tables

A table represents information in a tabular way, like a spreadsheet: distributed across a grid of rows and columns. In printed documents, tables commonly serve a subordinate function, illustrating some point described by an accompanying text. Tables still perform this illustrative function in HTML documents.

Basic Table Structure

Like other block elements, a table is marked at its beginning with a <TABLE> tag and its end with a </TABLE>. A table can contain a variety of row and column definitions within. At its simplest level, a table consists of the <TABLE> tag and one row with one cell:

```
<TABLE>
  <TR> <!-- table row -->
    <TD>Content <!-- table data -->
  </TABLE>
```

In a word, this simple table is called “pointless.” However, there are a few things to note. First, only the <TABLE> tag needs to be closed with </TABLE>; otherwise, the browser doesn’t know when to stop with table formatting. Second, the table row (<TR>) and table data (<TD>) cells can accept their respective closing tags, but they are not required. A new row or data cell automatically marks the end of the previous row or data, except in legacy versions of some browsers.

The <TABLE> tag has several attributes, which are given in the following syntax.

```
<TABLE [ BACKGROUND = path of an image file ]
  [ BGCOLOR = color name | #RRGGBB ]
  [ BORDER = pixels ]
  [ BORDERCOLOR = color name | #RRGGBB ]>
```

The BACKGROUND and BGCOLOR are same as in the <BODY> tag. The BORDER attribute sets the width of the border around table in pixels. Generally 0 is assumed to this attribute, i.e no border will appear. The last attribute BORDERCOLOR specifies a color for the table’s border. The following listing builds a simple table that appears as in Figure 8.1.

```

<HTML>
<HEAD>
<TITLE>
Simple Table
</TITLE>
</HEAD>

<BODY>
<TABLE BORDER=3>
<TR>
<TD>1.1<TD>Cell 1.2<TD>Data Cell 1.3<TD>Very Big Data Cell 1.5
<TR>
<TD>2.1<TD>Cell 2.2<TD>Data Cell 2.3<TD>Very Big Data Cell 2.5
<TR>
<TD>3.1<TD>Cell 3.2<TD>Data Cell 3.3<TD>Very Big Data Cell 3.5
</TABLE>
</BODY>
</HTML>

```

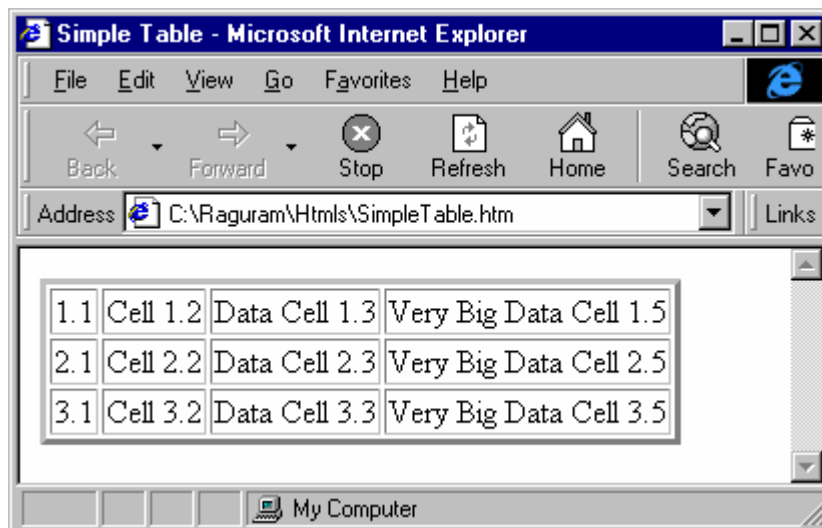


Figure 8.1 A simple table.

8.3 Aligning Table Elements

Cell Content Alignment: **ALIGN** and **VALIGN**

It is possible, through the use of the **ALIGN** and **VALIGN** attributes to align table elements within their cells in many different ways. These attributes can be applied in various combinations to the **<CAPTION>**, **<TR>**, **<TH>** and **<TD>** elements.

Syntax

```
<TD ALIGN = left | center | right VALIGN = top | middle | bottom >
```

In these attributes, **ALIGN** controls horizontal justification and **VALIGN** controls the vertical justification.

Explaining the Table: <CAPTION>

An important element for a table is a caption to identify it. The <CAPTION> tag is to the table what the <TITLE> tag is to be <HEAD> of the document, except that it is used to specify whether the caption appears at the top or bottom of the table.

If included, a caption must appear immediately after the open <TABLE> tag before any table rows or cells are defined.

Syntax

```
<TABLE>  
  <CAPTION [ ALIGN=alignment] >Caption Text</CAPTION>  
  ...Table contents  
</TABLE>
```

The ALIGN attribute is used slightly differently from other elements. In this case, it indicates where the caption is placed in relation to the table. Its possible values are **top, bottom, left, or right**.

Beginning to Include Data: <TR>

The beginning of a row is marked with a <TR> tag.

Syntax

```
<TR [ ALIGN = horizontalAlign ] [ VALIGN = verticalAlign ] >
```

<TR> marks the beginning of a row of cell definitions and can set default display settings for its cells. The ALIGN attribute sets the horizontal spacing for cells on the row and can be one of the following values: **left, right, or justify**.

The VALIGN attribute sets the vertical placement of information in the cell and takes one of these values: **top, middle, or bottom**.

Individual Data Cells and Headings: <TD> and <TH>

With the beginning of the row marked with <TR>, it's time to finally get down to the work of filling each cell. Two types of cell tags are used in a table. The first, <TH>, marks a header cell, which is similar to a heading tag on a Web page. Most browsers include a different font style for header cells to help emphasize their purpose to the user.

Syntax

```
<TABLE>  
<TR>
```

```
<TH>Cell Header</TABLE>
```

The *Cell Header* is the cell's header content. Generally this tag (<TH>) is used in defining a table's first row.

The other tag is a data cell tag, <TD>, used for the body of the table.

Syntax

```
<TD [ROWSPAN = numRows] [COLSPAN=numCols]
    [alignment]> cellContent
```

The attributes for <TD> are discussed in the following sections. A closing </TD> or </TH> is not required for individual cells, although many HTML designers and editing programs use it for clarity and compatibility with some older browsers.

8.4 Row and Column Spanning

Cell Size Attributes: ROWSPAN and COLSPAN

The first set of attributes determines its capability to merge with an adjacent cell. The ROWSPAN and COLSPAN attributes are used to combine adjacent cells into larger cells. It's important to note that when these attributes are used, the adjacent cells aren't eliminated; they're just "hidden" while the acquiring cell uses their space.

Syntax

```
<TD [ROWSPAN = numRows] [COLSPAN=numCols]>
```

In this attribute, numRows is the number of rows, including the current cell, that are joined. Likewise, numCols is the number of columns joined together. The default for both values is 1.

Spanning is a little tricky to plan and carry out. For example, start with look at the following listing, which has a 3 X 3 table, which requires three rows with three cells each.

```
<HTML>
<HEAD>
<TITLE>
Cells Span
</TITLE>
</HEAD>
```



```

<BODY>
<CENTER>
<TABLE BORDER=3>

<TR>
    <TD>Cell 1.1 <TD>Cell 1.2 <TD>Cell 1.3
<TR>
    <TD>Cell 2.1 <TD>Cell 2.2 <TD>Cell 2.3
<TR>
    <TD>Cell 3.1 <TD>Cell 3.2 <TD>Cell 3.3

</TABLE>
</CENTER>
</BODY>
</HTML>
    
```

Now it is time to merge the cells Cell 1.1 and Cell 1.2. Because this method reaches across columns, it is called *column span*. To do this change the data tag of the Cell 1.1 in the above listing as given below:

```

<TD COLSPAN=2>Cell 1.1
    
```

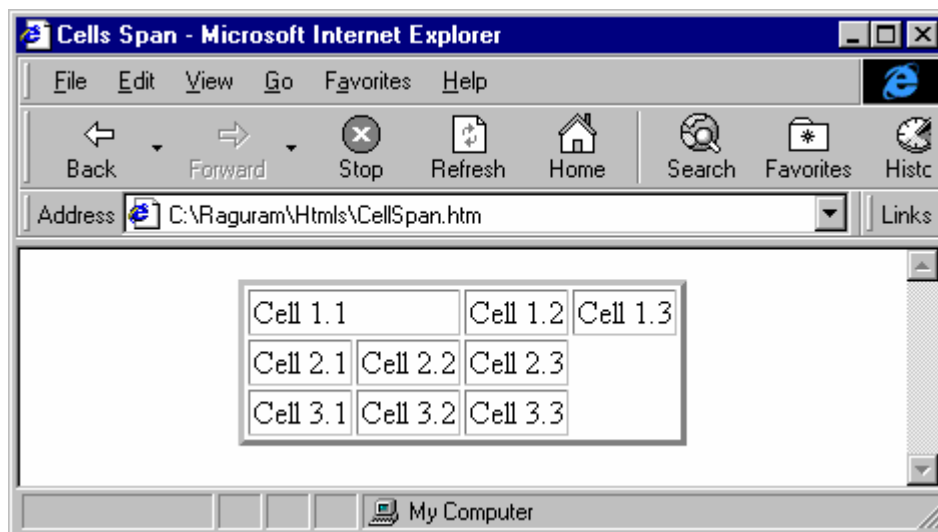


Figure 8.2 The revised table with Column span.

This change produces document as in Figure 8.2.

Notice the bottom two rows of cells in Figure 8.2. This reveals that spanning cells isn't quite simple as just adding the span attributes. Here's what happened: when the browser encountered COLSPAN = 2; it knew Cell 1.1 needed to take up the space of two cells, and it provided the space accordingly. Then, it continued across the row and finished adding the next two cells. The result was space for a total of four cells (two joined and two individuals).

At the next row tag, the browser started a new row by adding three cells. The browser didn't encounter a fourth <TD> tag, even though there was room in the table for one. Instead of adding a cell that the user didn't specify, it simply filled in with blank space. The final result is a 3 X 4 table with the bottom two rows only partially defined.

This feature of cell spanning can cause many headaches when working with tables. For each cell span, the corresponding cell definition is to be removed.

Remove the cell definition <TD> Cell 1.2 in the previous listing and view it in the browser. It will appear as in Figure 8.3.

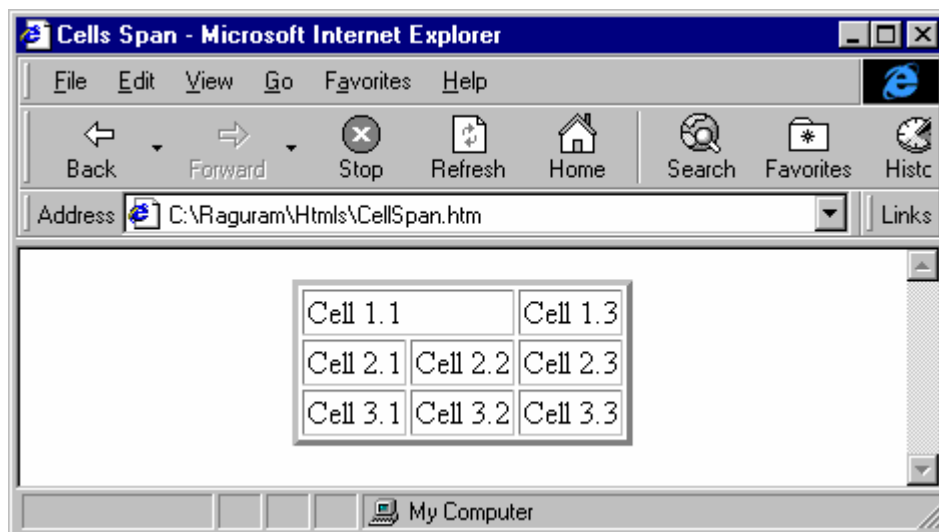


Figure 8.3 Table's appearance after the adjoining cell removal.

The same rules also apply for the ROWSPAN attribute, except cells below the originating tag should be removed. For example, in the following listing three adjoining cell tags are removed. It's also possible to combine the two attributes for other effects (see Figure 8.4).

```
<HTML>
<HEAD>
<TITLE>
Cells Span
</TITLE>
</HEAD>

<BODY>
<CENTER>
<TABLE BORDER=3>

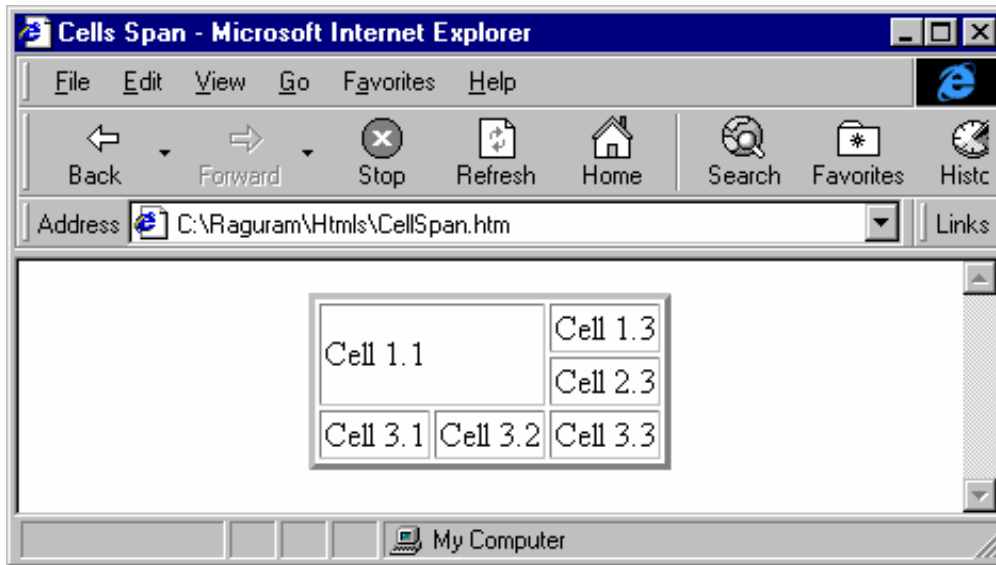
<TR>
    <TD ROWSPAN=2 COLSPAN=2>Cell 1.1 <TD>Cell 1.3
<TR>
    <TD>Cell 2.3
<TR>
    <TD>Cell 3.1 <TD>Cell 3.2 <TD>Cell 3.3
```

```

</TABLE>
</CENTER>
</BODY>
</HTML>

```

This produces the table as in Figure 8.4.



Figures 8.4 Table in which four adjoining cells are combined together.

When spanning in two directions at once, it is required to delete the all the data cells in the path of the merge, which means removing cells from the right, left, and diagonal directions.

8.5 Netscape Table Enhancements

Netscape Navigator has introduced several enhancements to HTML tables to increase the degree of control. This allows HTML authors to have design how their documents are displayed. The Netscape table enhancements are as follows

WIDTH attribute – This enables the author to specify the width of the table, either in pixels or as a percentage of the width of the browser window

HEIGHT attribute – This enables the author to specify the height of the table, either in pixels or as a percentage of the height of the browser

BORDER attribute – This attribute exists in the draft HTML 3.0 specification and puts a border around the table, and in that respect is supported by most Web browsers with table support. The enhancement also enables it to be used as a numerical attribute, `BORDER=<num>`, which makes the border `<num>` pixels wide

It should be noted here that, when using the Netscape BORDER=<num> table enhancement, it is possible to specify a table with no borders by including BORDER=0 in the <TABLE> element. While this will give a borderless table when viewed with Netscape Navigator, Web browsers that do not support this enhancement will ignore this “=0” and display the table with a border. So, to use a borderless table that will work on all browsers that support tables, include the <TABLE> element without specifying a BORDER attribute at all.

CELLPADDING and CELLSPACING – These numerical attributes include extra space within each cell in the table and/or within the borders of the table. If the border is not being displayed, they are equivalent.

Nested Tables – Netscape Navigator enables tables to be included as elements within other tables.

Using the STYLE attribute

The STYLE attribute can also be used (as used in <P> tag) for table elements to format the table, rows and columns independently. The following listing shows how to use this attribute.

```
<HTML>
<HEAD>
<TITLE>
Projects of Cyber World
</TITLE>
</HEAD>

<BODY>
<CENTER>
<H2>Department of Projects, Cyber World</H2>

<HR>
We have developed awesome projects, as this table shows. <BR>

<TABLE Cols=3 Border Bordercolor="Black" Style="color:green;background-color:Yellow">
<CAPTION Style="font-weight:bold;font-size:15">Project Details</CAPTION>

<TR>
<TD Rowspan=2 Style="text-align:center;font-weight:bold;background-color:Goldenrod">Projects</TD>
```

```
<TD Colspan=2 Style="text-align:center;font-weight:bold;background-color:Goldenrod">Team</TD>
</TR>

<TR Style = "text-align:center;font-weight:bold">
<TD Style = "background-color:Goldenrod">Leader</TD>
<TD Style = "background-color:Goldenrod">Programmers</TD>
<TR>

<TR>
<TD>Computer Based Training Development Platform </TD>
<TD>Saravana Kumar</TD>
<TD>Kalathi, Antony</TD>
</TR>

<TR>
<TD>Net Train</TD>
<TD>Balaji</TD>
<TD>Kannan, Ramanathan</TD>
</TR>

<TR>
<TD>256 Color Graphics Library</TD>
<TD>Raguram</TD>
<TD>Ganesh Sudhakar, Mythili</TD>
</TR>

</TABLE>
</CENTER>

</BODY>
</HTML>
```

This listing produces a table as in Figure 8.5.



Figure 8.5 Appearance of the Table due to the STYLE attribute.

8.6 Frames in HTML

Though tables, images and hyperlinks add attraction to Web pages, these alone are not enough to design pretty Web pages. Frames come handy in many ways to design web pages. It enables to design sophisticated user interfaces. A framed document divides a browser window into multiple panes, or smaller window frames. Each frame may contain a different document.

The benefits of this approach are obvious. Users can view information in on frame while keeping another frame open for reference, instead of moving back and forth between pages. The contents of one frame can be manipulated, or linked to the contents of another. This allows designers to build sophisticated interfaces. For example, one frame can contain links that produce a result in another frame. A Figure for such an interface is given in the next section.

Overview of Frames

A frame is an independent scrolling region, or window, of a Web page. Every Web page may be divided up into many individual frames, which can even be nested within other frames. Fixed screen sizes limit how many frames can realistically be

used at once. Frames provide navigation facilities. Figure 8.6 provides a visual overview of the components of a framed document.

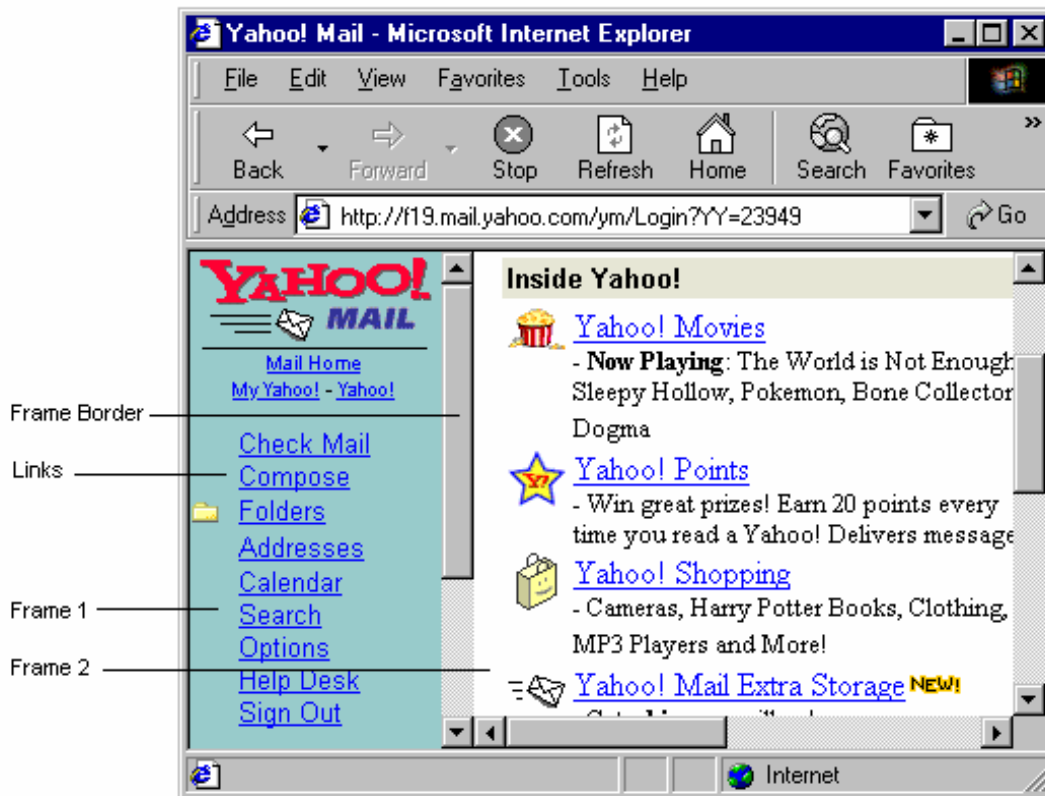


Figure 8.6 Frame road map.

8.7 Frameset Container

The first point to remember is that a framed document is composed of several documents. To illustrate, a page with two frames will actually involve three files:

- ◆ The framing document that defines the framing relationship
- ◆ The file that contains the contents of frame one
- ◆ The file that contains the contents of frame two

A simple two-frame document is shown in Figure 8.7.

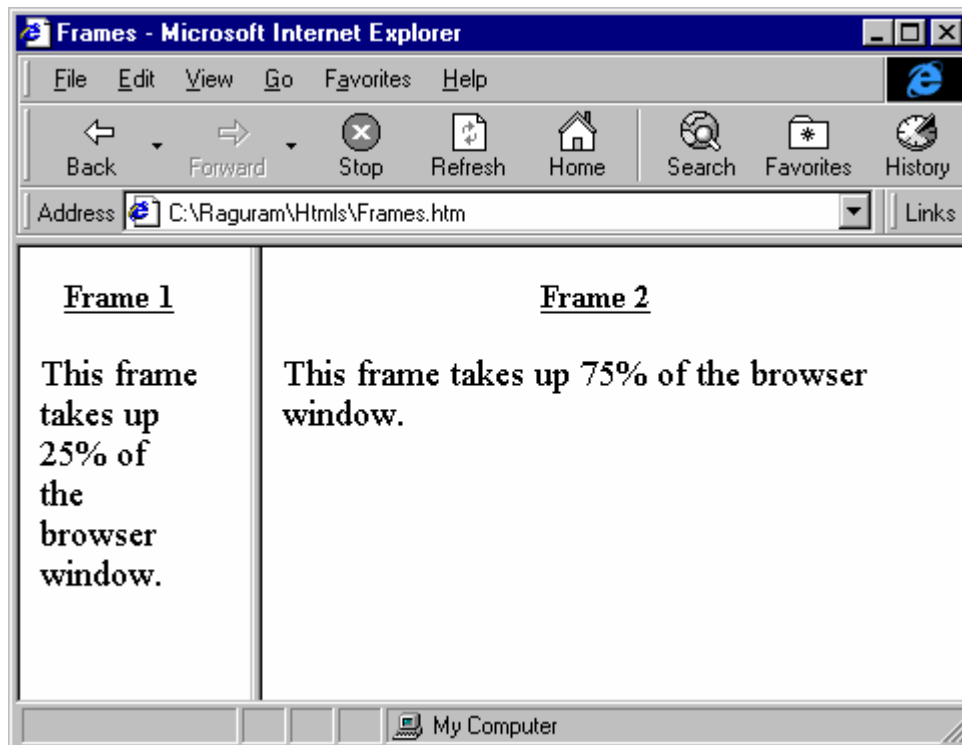


Figure 8.7 A framed document.

The first frame, on the left, takes up around 25 percent of the browser window and the right frame takes up the other 75 percent of the browser window. For the purpose of this example, the left frame is named as Frame1 and the right frame as Frame2.

To specify a framing document, the <FRAMESET> element is used in the HTML file instead of the <BODY> element. This element defines the set of frames that make up the document.

Syntax

```
<FRAMESET
  [ BORDER = pixels ]
  [ BORDERCOLOR = color ]
  [ COLS = columns width ]
  [ FRAMEBORDER = NO | YES ]
  [ FRAMESPACING = pixels ]
  [ ROWS = row heights ]
  [ TITLE = advisory text ]
<FRAME>....<FRAME>
</FRAMESET>
```

The attributes of the <FRAMESET> element are described below:

Border This attribute set the width in pixels of frame borders within the frame set. Settings BORDER=0 eliminates all frame borders.

Bordercolor This attribute sets the color for frame borders within the frame set using either a named color or color specified in the hexadecimal #RRGGBB format.

Cols This attribute contains a comma-delimited list, which specifies the number and size of columns contained with a set of frames. List items indicate columns, left to right. Column size is specified in three formats, which may be mixed. A column can be assigned a fixed width in pixels. It can also be assigned a percentage of the available width, such as 50 percent. Last , a column can be set to expand to fill the available space by setting the value to *, which act as a wildcard.

Frameborder This attribute controls whether or not frame borders should be displayed. Netscape supports YES and NO values. Microsoft uses 1 and 0 as well as YES and NO.

Framespacing This attribute indicates the space between frame in pixels.

Rows This attribute contains a comma-delimited list that specifies the number and size of rows contained within a set of frames. The number of entries in the list indicates the number of rows. Row size is specified with the same formats used for columns.

The Frame Tag

Once the frame layout is specified with the <FRAMESET> element, the contents of each frame must be specified using the <FRAME> element in the order that the frames were defined in the ROW or COL attribute. In the case of <FRAMESET COLS = "25%, 75%">, the contents of the first <FRAME> element encountered is loaded in the 25% column, and the contents of the second <FRAME> element in the 75% column.

Syntax

```
<FRAME  
    [ NAME = string ]  
    [ NORESIZE ]  
    [ SCROLLING = YES | NO | AUTO ]  
    [ SRC = URL of frame contents ] >
```

The attributes of the <FRAME> element are described below:

Name This attribute assigns the frame a name so that it can be the target destination of hyperlinks.

Noresize This attributes overrides the default ability to resize frame and gives the frame a fixed size.

Scrolling This attribute determines if the frame has scroll bars. YES value forces scroll bar, a NO value prohibits them, and an AUTO lets the browser decide. When not specified, the default value of AUTO is used.

Src This attribute contains the URL of the content to be displayed in the frame. If absent nothing will be loaded in the frame.

The code that displays the document as in Figure 8.7 is given below:

```
<HTML>
<HEAD>
<TITLE>Frames</TITLE>
</HEAD>
<FRAMESET Cols = 25%,75% >
    <FRAME Name = "Frame1" Src = "Frame1.htm" >
    <FRAME Name = "Frame2" Src = "Frame2.htm">

    <NOFRAMES><P>This document uses frames. Please follow this link to a
    <A HREF = "noframes.htm">noframes</A> version.
    </NOFRAMES>

</FRAMESET>
</HTML>
```

The above listing uses the <NOFRAMES>...</NOFRAMES> element within the frame set. This element provides information to be displayed in browsers that do not support frames. Although this approach seems like a good idea, the page author may need to maintain both frame and no frame version of a site in order to accommodate different browsers.

More about Frames (*For self learning*)

Frame Targeting

When using frames it may be desirable to make the links in one frame target another frame. This way, when a user activates a link in one framed document, the requested page loads in another frame.

The first part of link targeting is to ensure frame naming by setting the **NAME** attribute in the <FRAME> element to a unique name. The next part of linking is to use the **TARGET** attribute in the <A> element to set the target for the anchor. For example, a link like,

```
<A HREF = "http://www.excite.com" TARGET = "Frame2">
```

would load the site specified by the HREF attribute into the window called "Frame2", if there is such a frame. If the target specified by the name does not exist, the link loads over the window it is in. Some particular values for the TARGET attribute have special meaning; these are summarized in the following Table.

Value	Meaning
<code>_new</code>	Load the page into a new, generally unnamed, window.
<code>_self</code>	Load the page over the current frame.
<code>_parent</code>	Load the link over the parent frame.
<code>_top</code>	Load the link over all the frames in the window.

Reserved **TARGET** values.

The `_parent` value is not often encountered, because it is only useful when frames are nested to great degree. The `_parent` value makes it possible to overwrite the parent frame that contains the nested frame.

The following listing, which is contained in the file `Photos.htm`, constructs nested frames. First it splits the window into two rows by setting the **ROWS = 2**. The first row displays the contents of the file `Title.htm`, which displays the title "Photo Exhibition". This row can't be resized as its corresponding `<FRAME>` element has the **NORESIZE** attribute. The second row is further divided into two columns by using yet another `<FRAMESET>` element which has **COLS = 2**. The first column displays the contents of the file `Links.htm`, which contains links to photos. When a link is activated, an HTML file that contains its corresponding photo is targeted to the second column. This column is named as "Photo" and the links in the `Links.htm` file use this name for their **TARGET** attribute.

```
<HTML>
<HEAD>
<TITLE>Photos</TITLE>
</HEAD>
<FRAMESET ROWS = 20%,* >
    <FRAME Name = "Title" Src = "Title.htm" NORESIZE>
    <FRAMESET COLS = 20%,* >
        <FRAME Name = "Links" Src = "Links.htm" >
        <FRAME Name = "Photo">
    </FRAMESET>
</FRAMESET>
</HTML>
```

The listing of the `Title.htm` is given below.

```
<HTML>
<BODY BGCOLOR = "silver">
<H2 ALIGN=Center><U>Photo Exhibition</U></A>
</BODY>
</HTML>
```

The listing of the `Links.htm` is given below:

```
<HTML>
<BODY BACKGROUND="C:\Shared\Images\backgrd.gif">
<Center><B>Click any one of the following:</B></center>
<BR>
```

```
<A HREF = "Desert.htm" TARGET="Photo"> Desert </A> <BR>
<A HREF = "Sea.htm" TARGET="Photo"> Sea </A> <BR>
<A HREF = "Clouds.htm" TARGET="Photo"> Clouds </A> <BR>
<A HREF = "Mountains.htm" TARGET="Photo"> Mountains </A>

</BODY>
</HTML>
```

The listing of the Desert.htm is given below:

```
<HTML>
<BODY>
<IMG ALIGN=Center SRC= "Desert.jpg" >
</BODY>
</HTML>
```

The HTML files for the other links are similar to the Desert.htm with one exception. It will contain different image source.

When the Photos.htm is viewed in the browser it will appear as in Figure 8.8.



Figure 8.8 A typical framed document.

Floating Frames: <IFRAME>

Up until this point, all the frames shown have been attached to the sides of the browser (left, right, top, or bottom). Another form of frame, called a *floating frame*, is introduced by Microsoft. The idea of the floating frame is to create an inline framed region. And in this region text or image can be flowed around it. An inline frame is defined by the <IFRAME> element and may occur anywhere within the <BODY> of an HTML document.

Syntax

```
<IFRAME  
  [ ALIG N = LEFT | RIGHT | TOP | BOTTOM ]  
  [ BORDER = pixels ]  
  [ BORDERCOLOR = color ]  
  [ FRAMEBORDER = YES | NO ]  
  [ HEIGHT = pixels or percentage ]  
  [ NAME = frame name ]  
  [ NORESIZE ]  
  [ SRC = URL of frame contents ]  
  [ WIDTH = pixels or percentage ] >
```

... Contents for the browser that do not support floating frames

```
</FRAME>
```

The major attributes to set for the <IFRAME> element include SRC, HEIGHT, and WIDTH. The SRC is set to the URL of the file or image to load, while the HEIGHT and WIDTH are set to the pixel or percentage value of the screen that the floating frame region should consume. The other attributes are same as in <FRAMESET> element.

Note that unlike the <FRAME> element the <IFRAME> comes with the close tag </IFRAME>. <IFRAME> and </IFRAME> should contain any HTML markup code and text that should be displayed in browsers that do not support floating frames.

The following listing gives example to use <IFRAME> element to display a big image, which appears as in Figure 8.9.

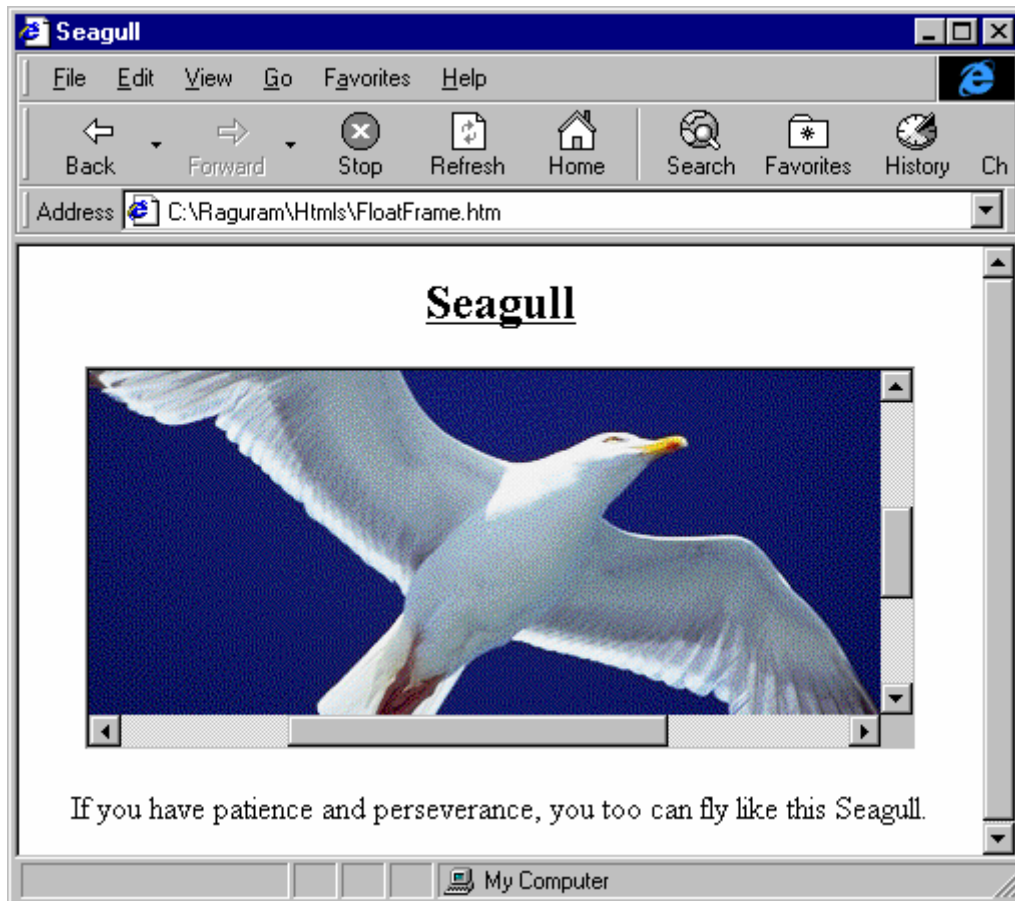


Figure 8.9 Floating frame containing an Image.

If the URL of an HTML file is assigned to the **SRC** attribute of the IFRAME element, the HTML file will be displayed in side the floating frame.

8.8 Short Summary

- ◆ A table represents information in a tabular way, like a spreadsheet: distributed across a grid of rows and columns.
- ◆ Cell Content Alignment can be performed using ALIGN and VALIGN
- ◆ The ROWSPAN and COLSPAN attributes are used to combine adjacent cells into larger cells.
- ◆ A framed document divides a browser window into multiple panes, or smaller window frames.
- ◆ A frame is an independent scrolling region, or window, of a Web page.
- ◆ To specify a framing document, the <FRAMESET> element is used instead of the <BODY> element.

8.9 Brain Storm

1. Is it possible to create tables without border? Using border attribute is it possible to have table without border?

2. Is it possible to align the table in the center of the browser window

- a. Yes
- b. No

If Yes what is the attribute?

3. Is it possible to align the contents of a particular row inside a table without

affecting other Row's contents

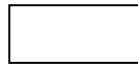
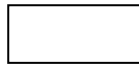
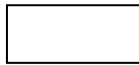
- a. Yes
- b. No

4. Is it possible to get a table structure like the following

Table 1

Table 2

Table 3



- a. Yes
- b. No

5. What is the need for a frame?

6. How many body tags are possible inside a frame set?

7. Which of the following browsers support frame?

- a. Internet Explorer 3.0
- b. Netscape Navigator 4.0
- c. Internet Explorer 4.0
- d. Netscape Navigator 3.0
- e. Mosaic

8. What is the significance of noresize attribute?

9. What is the difference between Yes and auto in a scroll attribute?

10. What is a floating frame?

11. Is it possible to have more than one floating frame in an HTML document?

12. What will happen if a browser does not support frames?

13. What is the use of no frames?

14. When parent value can be used for the target attribute, where the result is placed?

Lab Unit (2 Real Time Hours)

This Exercise will make you understand completely about Table and Links. The web page you are about to create is for LIFE STYLE -Shopping complex.

Heading LIFE STYLE - SUPER MALL

The contents -

LIFE STYLE is a chain of International shopping mall Estd. in 1980 in CHENNAI. Promoted by Business tycoons from Singapore with an investment of nearly 100 crores. In the very first Year of its launch LIFE STYLE has captured a major share in the market and they have opened 20 Branches across 5 countries.

The Details about their Branches:

LIFE STYLE 's INTERNATIONAL MARKET

COUNTRY	PLACE
INDIA	CHENNAI
	BANGALORE
	HYDERABAD
AUSTRALIA	SYDNEY
	MELBOURNE
CANADA	TORONTO
AMERICA	NEW YORK

All the leading Corporate are is doing business of their products through LIFE STYLE.

The following are available for the customers in LIFE STYLE

FOOD & BEVERAGES

CLOTHING

AUDIO / VIDEO

VARIETY ITEMS

The listed items are HYPERLINKS to other pages.

Create corresponding web page, which gives more detailed info about the selected link items.

For example if you click AUDIO/VIDEO link it takes you to.../audiovideo.htm

<i>Customer's Choice</i>		
BRAND NAME	PRODUCT DESCRIPTION	PRICE
<i>AIWA</i>	<i>DVD PLAYER</i>	<i>\$19000</i>
<i>PANASONIC</i>	<i>VCD HI-FI MINI PLAYER</i>	<i>\$12,800</i>
<i>THOMSON</i>	<i>21" FFST T.V</i>	<i>\$8000</i>
	<i>29" HOME THEATRE SYSTEM</i>	<i>\$14,980</i>
	<i>WALKMAN</i>	<i>\$20</i>
<i>LG</i>	<i>GOLDEN EYE</i>	<i>\$13,777</i>

Design the Web Pages according to your requirement. Use all the possible TAG options.

Lecture 9

HTML Forms

Objective

This lecture provides an insight into the following:

- ❖ Using Forms to create interactive Web pages
- ❖ The Form elements
- ❖ What are dynamic documents
- ❖ Including background graphics and color to our Web page
- ❖ Playing background sound in our Web page
- ❖ Marquees as a tool for scrolling

Lecture - 9

- 9.1 Snap Shot
- 9.2 HTML Forms
- 9.3 The <Input> Tag
- 9.4 Dynamic documents
- 9.5 Background Graphics and Color
- 9.6 Microsoft Internet Extensions
- 9.7 Font Tag Enhancements
- 9.8 Scrolling Marquees
- 9.9 Short Summary
- 9.10 Brain Storm

Lab unit

9.1 Snap Shot

It is very important to note that, most Web pages do more than just rendering information. E-commerce, the emerging technology is done through the Web. HTML offers Forms to design Web pages that can be used for order processing on a retail site or they can be set up to get customer feedback.

9.2 HTML Forms

Forms are constructed in the HTML documents by using the <FORM> element. This element contains several other elements, called controls that have a variety of methods for gathering information. When a form is completed and submitted, the information in its active controls is passed to a program that takes whatever action the form has been designed to perform. Each element in the form has both a name and a value, thus the data that's passed for processing is in the form of name/value pairs.

The processing of data is done by scripting languages or CGI program. CGI programs are written in the languages - Perl, Java and C. This chapter doesn't deal with the data processing but shows the construction of Forms using its controls.

The FORM element

The FORM element has three main attributes - Name, Action and method.

Syntax

```
<FORM
    [ NAME = string ]
    [ ACTION = URL ]
    [ METHOD = GET | POST ] >
    ... Form elements
</FORM>
```

The NAME attribute gives a name to the Form. The ACTION attribute gets an URL that specifies the address of the program used for processing the data, as in the following example:

```
<FORM ACTION = "http://www.cybershopping.com/getmoney.pl"
    METHOD = "POST">
.....
</FORM>
```

The METHOD attribute can have either GET or POST as its value; GET is the default value choice. It submits the name/value pairs to the URL specified in the METHOD attribute as an appendage to the URL itself; POST, on the other hand, sends them as a separate section following the HTTP header. This separate section is called the

entity body. But the last two attributes are required only when dealing with Active Server Pages.

9.3 The <Input> Tag

The INPUT Element is the most critical to using forms. It's entirely possible to build an entire form using no other elements due to the variety of widgets available through the **TYPE** attribute (see Figure 9.1).

Syntax

```
<INPUT [ TYPE = text | password | checkbox | radio | submit | reset | image | button ]
      [ NAME = control name ]
      [ VALUE = control value ]
      [ CHECKED ]
      [ DISABLED ]
      [ READONLY ]
      [ SIZE = control width ]
      [ MAXLENGTH = word length ]
      [ SRC = URL      ]
      [ ALIGN = left | center | right ]
      [ TABINDEX = tab number ] >
```

Acceptable values for the TYPE attribute are listed in Table given below

Form Controls	Values for the TYPE attribute
Custom push button	Button
Off/On check box	Checkbox
Graphic files	Image
Masked text entry	Password
Radio buttons	Radio
Reset button	Reset
Submit button	Submit
Text entry boxes	Text

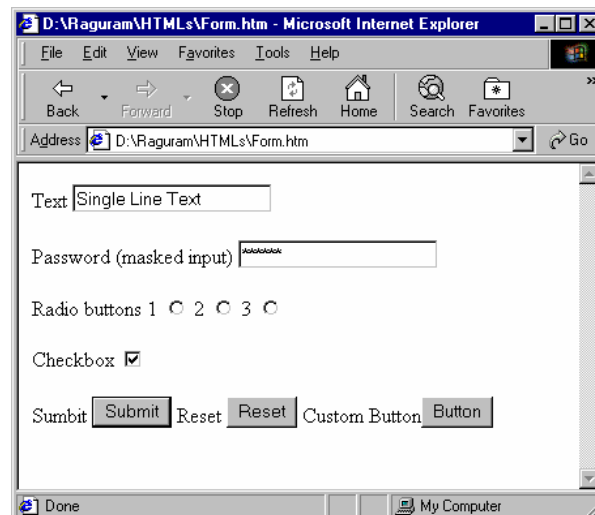


Figure 9.1 Form built with only INPUT element.

The following sections examine each type of button in more detail.

The Button value

Before HTML 4, the only buttons available were the Submit and Reset buttons whose meaning and actions were predetermined. The button input type, however, has no default function; its function is defined by the author in a script. A typical button declaration looks like this:

```
<INPUT TYPE="Button" NAME="button_01" VALUE="Click Me">
```

Each control is assigned a unique identifier with the NAME attribute; without this identifier, it would be impossible to tell which control was assigned what value. The VALUE attribute's text is displayed on the button.

The Reset Value

The reset value for the type attribute creates a button with only one purpose: all form entries are cleared to their default entries or left blank if no default is specified. It's declared with the following line:

```
<INPUT TYPE="Reset">
```

The Submit Value

The Submit value of the TYPE attribute creates a button that, like the Reset button, has only a single purpose. In this case, it's to send the name/value pairs of the active form elements to the URL specified in the FORM declaration.

Strangely enough, the HTML specification enables multiple Submit buttons to exist. The Submit button is declared with the following code:

```
<INPUT TYPE= "Submit">
```

The CheckBox Value

The CheckBox value of the TYPE attribute is a Boolean input device; it's either off or on. It looks like a hollow box that, when selected is filled with a check mark to indicate its active state. A check box is extremely versatile and can be used in several different ways. The basic code for declaring a check box is as follows:

```
<INPUT TYPE= "Check" NAME = "choice1" VALUE = "Cricket" CHECKED >
```

The **CHECKED** attribute is optional; it causes the check box to be "on" (that is, filled with a check mark) when the form is first created. Of course, if a user clicks on the filled check box, it switches to its empty state. The CHECKED attribute doesn't establish a permanent state; it just sets the Default State of the check box.

To enable user to make multiple selections on the same topic, several check boxes can be used with the same NAME and a different VALUE. For example, to request different choices on hobby, the code will be something like this:

```
<H3> Choose your hobbies </H3>
```

```
<FORM>
```

```
<INPUT TYPE= "Checkbox" NAME= "HOBBY" VALUE = "Games">Playing Games <BR>
```

```
<INPUT TYPE= "Checkbox" NAME= "HOBBY" VALUE = "Songs">Hearing Songs <BR>
```

```
<INPUT TYPE= "Checkbox" NAME= "HOBBY" VALUE = "Books">Reading Books <BR>
```

```
<INPUT TYPE= "Checkbox" NAME= "HOBBY" VALUE = "Shopping" CHECKED> Shopping
```

```
</FORM>
```

The result of this code is shown in Figure 9.2.

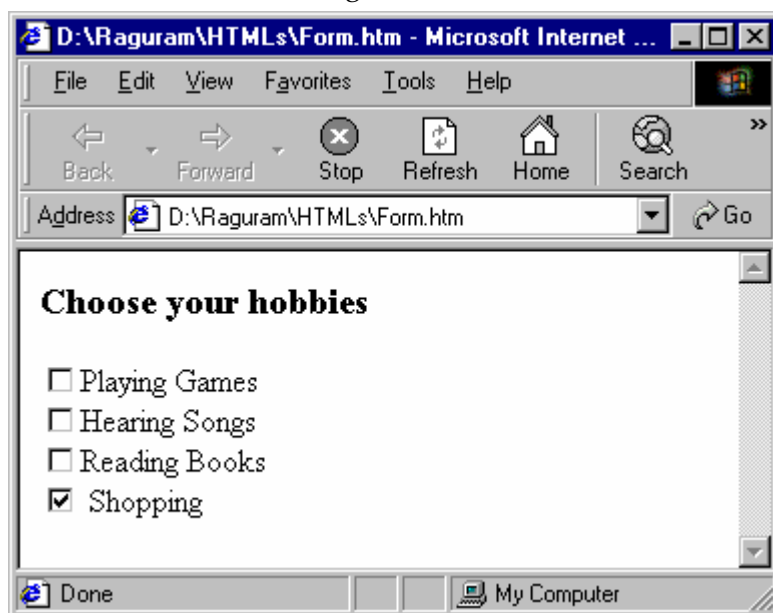


Figure 9.2 A Form with multiple check boxes.

The Radio Value

The radio value of the TYPE attribute is very similar to the Checkbox type. The difference between the two is that radio buttons are mutually exclusive meaning that selecting one turns the others off. So only the final selection is sent along with the other form data when the form is submitted. Example of designing radio buttons is given below:

```
<H3> Choose your pet </H3>
<FORM>
<INPUT TYPE= "Radio" NAME= "Pet" VALUE = "Dog" CHECKED>Dog <BR>
<INPUT TYPE= "Radio" NAME= "Pet" VALUE = "Cat">Cat <BR>
<INPUT TYPE= "Radio" NAME= "Pet" VALUE = "Dove">Dove <BR>
</FORM>
```

The result of this code is shown in Figure 9.3.

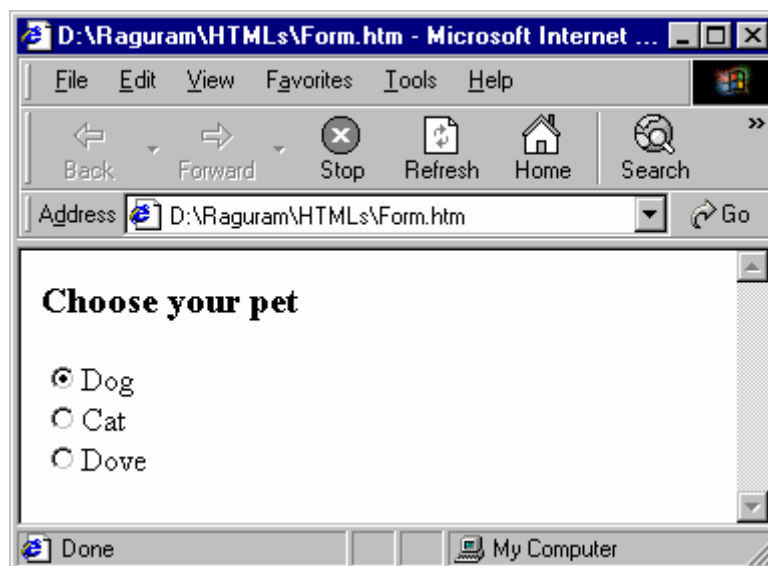


Figure 9.3 A set of radio buttons.

The Text Value

It asks for input in the form of a single-line, typed response of a given length. The code for adding text response for a user's first name is as follows:

```
<INPUT TYPE= "Text" NAME= "FirstName">
```

A number of text responses could be strung together to create a meaningful form, as shown in the following code and illustrated in Figure 9.4.

```
<H3> Your Address Please </H3>
<FORM>
Name : <INPUT TYPE= "Text" NAME= "Name" SIZE = "50"> <BR>
```



```
Address: <INPUT TYPE= "Text" NAME= " Address" SIZE = "40">
Phone: <INPUT TYPE= "Text" NAME= "Phone" SIZE = "20"> <BR>
</FORM>
```

The size attribute sets the length of the text box in characters. It is also possible to set the maximum number of characters that can be entered by using the MAXLENGTH attribute. The code for setting MAXLENGTH IS looks like this:

```
<INPUT TYPE= "Text" NAME= "Name" SIZE = "50"
MAXLENGTH = "50">
```

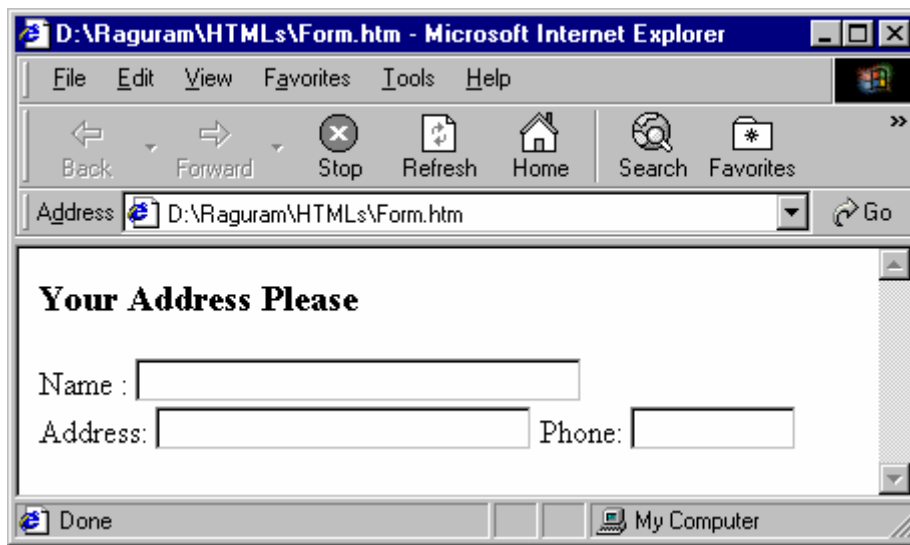


Figure 9.4 Text boxes used to gather visitors input.

Text can be locked against change by the READONLY attribute. The value of such text is included with the values of other active elements when the form is submitted. Read-only elements are capable of receiving focus and are included in tabbing navigation. This kind of text enables the user to see what's being sent.

The Password Value

The Password value of the TYPE attribute is exactly like the TEXT attribute, except that the visible response on the form is masked with an asterisk or similar character so that no one looking over the shoulder of the user can read what is typed. Here's an example of the code for setting Password:

```
<INPUT TYPE= "Password" NAME= "Secret" SIZE= "20">
```

9.4 Dynamic Documents

Server push and client pull are Netscape 2.0's innovative new techniques for automatically updating Web pages. We shall now consider a few uses for these "dynamic documents"

- ◆ Creation of automated slide shows by loading sequence of pages at timed intervals

- ◆ Can automatically advance viewer to another site. This is useful, especially if our Web site moves to a new URL. “Slipper” pages can be created, which is displayed but disappear after a few seconds. We might use this feature for a “teaser” that shows users what is available at a site but that would not stay put unless the user enters passwords.
- ◆ Creation of auto surf documents that advance viewers to a new random page every few seconds.

Server push applications keep a connection open from the server to the browser and “push” a stream of data at the browser. Server push is often used for data-driven applications such as animating icons, though it is rapidly being replaced for such purposes by Java applets.

Client pull, on the other hand, is implemented using the <META> tag and a browser capable of performing it, such as Netscape 2.0. Compared to server push, client pull is relatively easy to set up. Though simple in its implementation and somewhat obscure in syntax - client pull is a powerful new Netscape addition to HTML

9.5 Background Graphics and Color

We can include background graphics and color to our Web page, which makes it attractive for any viewer. The <BODY> tag allows two extensions to define tileable background graphics and custom background colors for Web pages. Though these extensions originated in Netscape, most browsers now support these features.

```
<BODY BACKGROUND = “flower.gif” BGCOLOR = “#FFFFFF”>
```

Here the BACKGROUND attribute fills the background of the browser window with the specified graphic image file. If the image is smaller than the page, it is tiled to fill the window. We also have an alternative to using hexadecimal color values.

Example:

```
<BODY BGCOLOR=“BLUE”> gives a blue screen.
```

9.6 Microsoft Internet Explorer Extensions

Background Sound

BGSOUND tag of the Microsoft Internet Explorer aids us to create pages with background sounds. When a page with this tag is loaded, the sound plays automatically, and a LOOP attribute controls the number of times it is to be played. Sounds can consist of .wav or .au sampled sound files or MIDI (.mid) music files.

```
<BGSOUND SRC = “xmascarol.wav”>
```

The SRC attribute contains the URL of the source file to be played. The sound "xmascarol.wav" in the preceding example plays once when the page is loaded. Here are two examples that play more than once using the LOOP attribute:

```
<BGSOUND SRC="xmascarol.wav" LOOP=5>
```

```
<BGSOUND SRC="xmascarol.wav" LOOP=INFINITE>
```

The first example plays five times; the second plays until we leave the page. LOOP can have any integer value or be INFINITE; -1 is the same as specifying INFINITE.

9.7 Font Tag Enhancements

Like Netscape, Explorer adds the BASEFONT tag, as well as the FONT tag with COLOR and SIZE attributes. But it adds one more attribute FONT FACE, which is used to define the font, or typeface for the tagged text. Here's the generic format:

```
<FONT FACE="Times New Roman", "Courier New", "Arial">Hello!</FONT>
```

In this example, the message Hello! Would be displayed in the Times New Roman font, if available. If not, Courier New would be used, or Arial in case Courier New does not exist.

9.8 Scrolling Marquees

We may get the impression that many of Explorer's extensions to HTML are meant to compete directly with the HotJava demos being shown by SunMicrosystems. They typically display small animations and scrolling marquees. Explorer has added a new MARQUEE HTML tag just for creating the latter. A variety of special attributes exist for controlling Explorer marquees.

Syntax

```
<MARQUEE> Scrolling Scrolling Scrolling </MARQUEE>
```

Any text within the <MARQUEE>...</MARQUEE> tags scroll sideways. The BEHAVIOR attribute lets us pick how the text moves. With a value of SCROLL, text scrolls in from one side and off the other. SLIDE scrolls in from one side and stops as soon as the text touches the opposite margin. ALTERNATE bounces text back and forth within the marquee.

Beyond the basics, MARQUEE offers an incredible amount of control over the way our marquee looks and acts.

Attributes of MARQUEE:

- ◆ BEHAVIOR=SCROLL | SLIDE | ALTERNATE: Defines how the text moves.
- ◆ SCROLL wraps, SLIDE stops and ALTERNATE ping-pongs.

- ◆ DIRECTION=LEFT|RIGHT: Specifies the direction in which the text moves to and not the direction it comes from.
- ◆ ALIGN=TOP|MIDDLE|BOTTOM: Determines whether surrounding text aligns with the TOP, MIDDLE or BOTTOM of the marquee.
- ◆ BGCOLOR= "color": Defines background color as a hexadecimal value ("#FFFFFF") or color name ("blue")
- ◆ HEIGHT=n|n%: Specifies marquee height in pixels (HEIGHT=50) or as a percentage of screen height (HEIGHT=33%)
- ◆ WIDTH=n|n%: Determines marquee width in pixels or a percentage of screen height (See HEIGHT)
- ◆ HSPACE=n: Defines left and right outside margins in pixels (HSPACE=35)
- ◆ VSPACE=n: Specifies top and bottom outside margins in pixels (See HSPACE)
- ◆ LOOP=n|INFINITE: Determines how many times a marquee loops.
- ◆ SCROLLAMOUNT=n: Defines how far the marquee moves each step, in pixels
- ◆ SCROLLDELAY=n: Specifies the delay between moves in milliseconds.

Let us now consider an example, which uses most of these attributes. The output is given in figure 9.5

```
<HTML>
<HEAD>
<TITLE> World of Marquess</TITLE>
</HEAD>
<BODY>
<FONT SIZE =10>
<A HREF = "marq.html">
<MARQUEE BEHAVIOR=SCROLL DIRECTION=LEFT LOOP=INFINITE BGCOLOR=
"blue" HEIGHT=250 WIDTH=85% HSPACE=20 VSPACE=10 BORDER=20
SCROLLDELAY=25 SCROLLAMOUNT=3 ALIGN=MIDDLE> This is the Wonderful World
of Marquess! </MARQUEE></A></FONT>
</BODY>
</HTML>
```

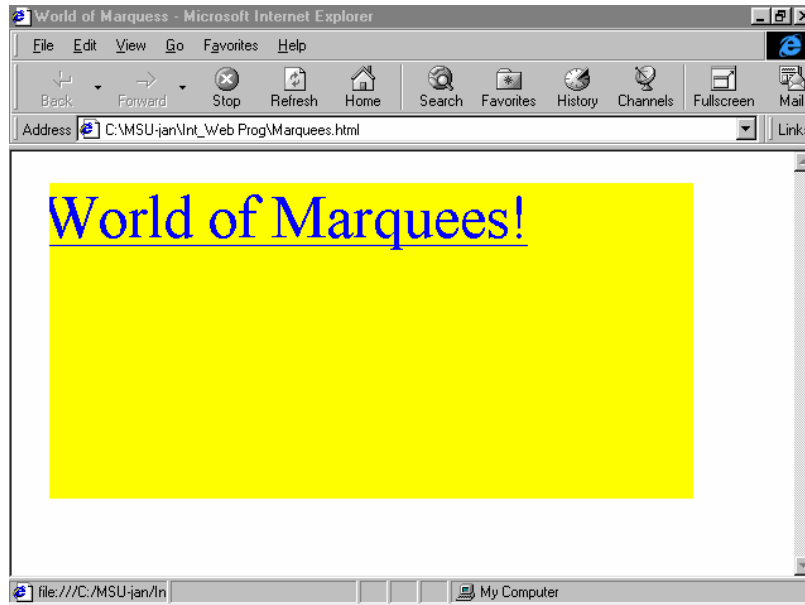


Figure 9.5 An Example for Scrolling Marquees.

Other form Attributes (for self learning)

Closely related to READONLY is the DISABLED attribute. When an element is disabled, its contents are not only unalterable, but unusable, so it's taken out of the tab order. When a form is submitted, the name/value pairs of disabled form elements are not included. If the disabled element is a button (whether custom, Reset, or Submit), the button can't be activated.

Although READONLY is limited to the **text** and **password** types within the INPUT element, DISABLED can be applied to any INPUT type, as well as to the TEXTAREA, SELECT, OPTION, LABEL and BUTTON elements.

The BUTTON Element

The BUTTON element takes three possible values for the TYPE attribute: submit, reset and button. These values are all duplicates of their counterparts in the INPUT element.

Syntax

```
<BUTTON
    [ NAME = control name ]
    [ VALUE = control value ]
    [ TYPE = button | submit | reset ]
    [ DISABLED ]
    [ TABINDEX = tab number ] > Button Text </BUTTON>
```

The SELECT and OPTION Elements

The SELECT element is used to create a list of choices either as a drop-down menu or a list box. Each of the choices in the list is an OPTION element.

Syntax

```

<SELECT
    [ NAME = control name ]
    [ SIZE = control width ]
    [ MULTIPLE ]
    [ DISABLED ]
    [ TABINDEX = tab number ] >
  <OPTION
      [ SELECTED ]
      [ DISABLED ]
      [ VALUE = control value ] >
      .....Text label or value
  </OPTION>
  ..... other option elements
</SELECT>

```

A selection list is created by adding one or more OPTION elements within the SELECT element, much like an HTML list:

```

<SELECT NAME = "Pets" >
  <OPTION> Dog </OPTION>
  <OPTION> Cat </OPTION>
  <OPTION> Dove </OPTION>
</SELECT>

```

The end tags on the OPTION element are optional. If they're not used, the element automatically terminates at the beginning of the next OPTION element or at the end of the SELECT element that contains it.

The OPTION element can also have a specified VALUE assigned to it, but that's not required. If it's absent, the contents of the OPTION element will become the "VALUE" part of the name/value pair.

The result of the proceeding code is shown in Figure 9.6.



Figure 9.6 The SELECT element drop-listing its OPTIONS.

Using `SELECTED` attribute with options causes the selected element to be highlighted. The user can accept it or deselect it at will, and more than one option can be selected.

By default, the `SELECT` element displays a drop-down menu in which only the first element is displayed (as in Figure 9.6). However, the `SIZE` attribute can be used to make more of the options visible. The following code illustrates this:

```
<FORM>
<SELECT NAME = "Pets" SIZE=5 >
    <OPTION> Playing Games </OPTION>
    <OPTION> Hearing Songs </OPTION>
    <OPTION> Reading Books </OPTION>
    <OPTION> Shopping </OPTION>
</SELECT>
</FORM>
```

When the `SIZE` attribute is specified the appearance of the menu will slightly be different. The menu is displayed as a simple list instead of as a drop down list (see Figure 9.7).

Users can select only a single choice from the menu, but using the `MULTIPLE` attribute will enable them to select a range of choices. Of course, they can still choose only one if that's their wish, but they can now choose any or all available options. If more than one option is chosen, multiple name/value pairs are sent when the form is submitted. Each has the same name, but a different value. The `SELECT` element that uses the `MULTIPLE` attribute is given below.

```
<FORM>
<SELECT NAME = "Pets" SIZE=5 MULTIPLE>
    <OPTION> Playing Games </OPTION>
    <OPTION> Hearing Songs </OPTION>
    <OPTION> Reading Books </OPTION>
    <OPTION> Shopping </OPTION>
</SELECT>
</FORM>
```

The result of this code is shown in Figure 9.7.

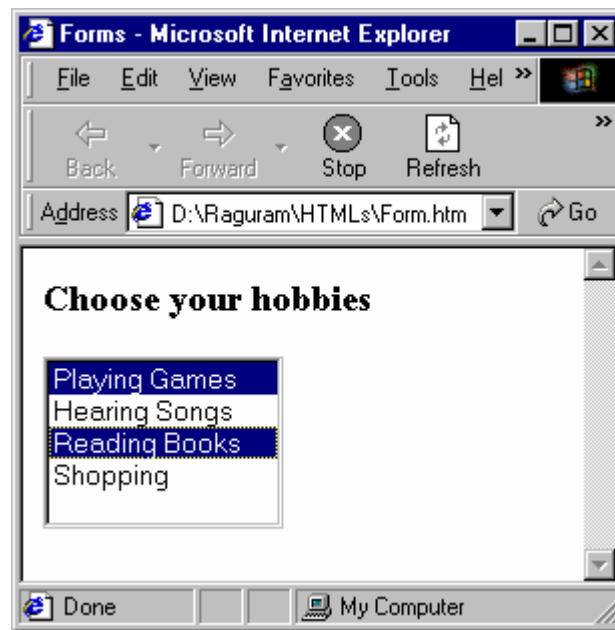


Figure 9.7 *SELECT* element enabling multiple selection.

The TEXTAREA Element

The TEXTAREA element is similar to the INPUT element's text type. The difference is that users can type a larger section of text than they can with the text boxes. Instead of a single line of text, there is a large window where multiple-lines responses can be typed.

Syntax

```
<TEXTAREA
    [ NAME = control name ]
    [ ROWS = number of rows ]
    [ COLS = number of columns ]
    [ DISABLED ]
    [ READONLY ]
    [ TABINDEX = tab number ] >
```

```
...Text...
</TEXTAREA>
```

TEXTAREA is typically used for comments or "delivery" instructions - anything that requires more than a simple response. The dimensions of the window are specified with the ROWS and COLS attributes. Number of rows means number of lines and number of columns means number of characters per line. The following code illustrates the use of the TEXTAREA element; the results are shown in Figure 9.8.

```
<TEXTAREA NAME= "Comments" ROWS= "4" COLS = "15">
  This is the place to type your comments.
</TEXTAREA>
```

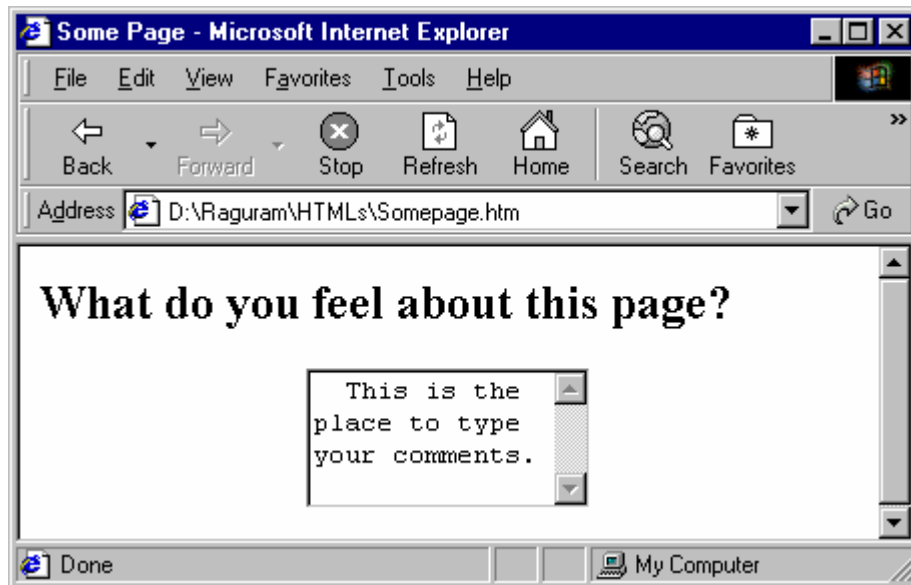



Figure 9.8 TEXTAREA in a Web page.

TEXTAREA requires both start and end tags. As with the INPUT element's TEXT, TEXTAREA elements can use the READONLY attribute. If it's specified, then the user can't alter any initial content provided by the author.

The LABEL Element

As the name implies, the LABEL element is the text that labels a control. Unlike normal text, however, the label and its associated control both share the same focus. In other words, if the label is clicked the effect will be same as though the control is clicked.

Syntax

```
<LABEL  
    [ FOR = control name ]  
    [ DISABLED ] >  
</LABEL>
```

Labels are associated with controls in one of two ways, either implicitly or explicitly. In implicit association method, the associated element is contained in the LABEL element as illustrated in Figure 9.9 and the following code:

```
<LABEL> Your nick name  
    <INPUT TYPE= "Text" NAME= "NickName" SIZE=15>  
</LABEL>
```

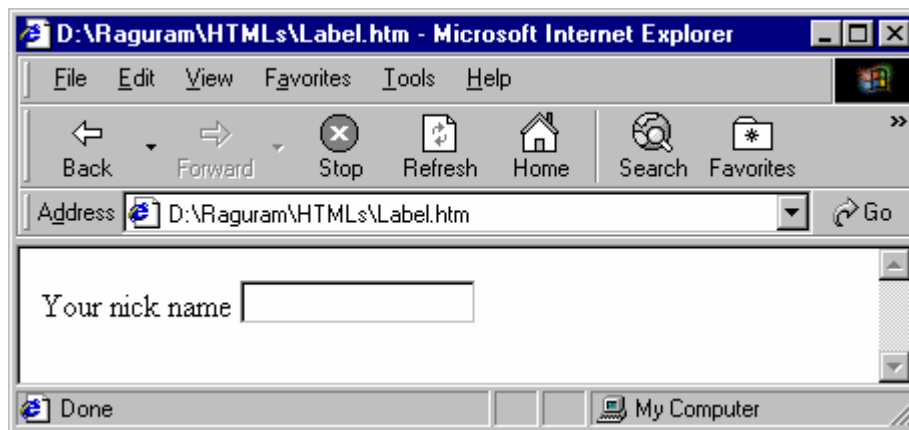


Figure 9.9 A Label associated with a text box.

With an explicit association, the label is tied in by using the control element's ID attribute. The FOR attribute of the LABEL element is assigned the value of the control element's ID attribute, as shown in the following:

```
<LABEL FOR= "NName" > Your nick name </LABEL>
<INPUT TYPE= "Text" NAME= "NickName" SIZE=15 ID= "NName">
```

The LABEL element must still come before the control for the display to look right; a logical association has nothing to do with visual placement.

The FIELDSET and LEGEND Elements

The FIELDSET and LEGEND elements create border around grouped controls, just as panels do in standard programming, and legends are label that refer to overall field set.

Syntax

```
<FIELDSET>
  <LEGEND>Legend Text</LEGEND>
  ....Control elements
</FIELDSET>
```

Both elements require start and end tags. The use of field sets and their legends is illustrated in Figure 9.10 and the following code:

```
<FIELDSET>
<LEGEND><B>Address Details</B></LEGEND>
<LABEL> Name :
<INPUT TYPE= "Text" NAME= "Name" SIZE = "30">
</LABEL> <BR>
<LABEL> Address:
<INPUT TYPE= "Text" NAME= "Address" SIZE = "15">
</LABEL>
```

```
<LABEL> Phone:
<INPUT TYPE= "Text" NAME= "Phone" SIZE = "10"> <BR>
</LABEL>
</FIELDSET>
```

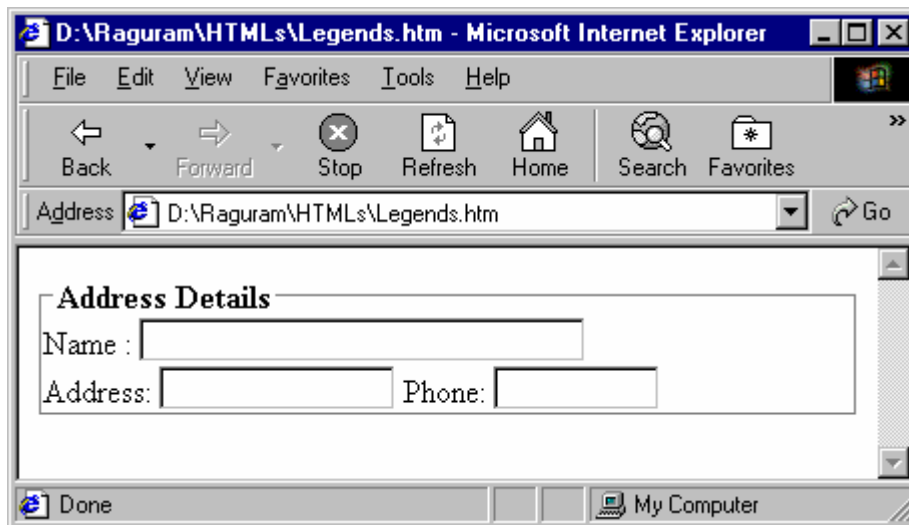


Figure 9.10 Result of FRAMESET element.

There is no provision in the HTML specification for controlling the size of fields sets; they stretch across the entire screen, regardless of the space taken up by the controls they contain.

Tab Navigation

The TABINDEX attribute is for navigating through a form by using the Tab key to move from one element to the next. This attribute is not required, if the form is to be navigated in the order that the elements appear, as is usually the case. If TABINDEX isn't specified, then that's the default behavior of the tab key.

However, if the navigation is required in some other order (such as vertically through adjacent control element), this attribute is useful. To set the tab order for a pair of text input boxes, the following code is used:

```
<INPUT TABINDEX= "1" TYPE= "Text" NAME = "FirstName">
```

```
<INPUT TABINDEX= "2" TYPE= "Text" NAME = "LastName">
```

If two elements have the same TABINDEX value, the navigation order is the order in which they appear.

It is not necessary for the assigned values to be sequential; all that's required is that, they be in ascending order corresponding to the navigation pattern. Therefore, the following code has the same effect as the previous example:

```
<INPUT TABINDEX= "10" TYPE= "Text" NAME = "FirstName">
```

```
<INPUT TABINDEX= "20" TYPE= "Text" NAME = "LastName">
```

In fact, it is good idea to give non-sequential number values to the TABINDEX attributes of successive elements. This technique gives room for the new elements. New elements can be inserted in the tab order, with out the need of reordering the tab index of other already existing elements.

The TABINDEX attribute is supported by the BUTTON, INPUT, SELECT, and TEXTAREA elements.

9.9 Short Summary

- ◆ Forms are constructed in the HTML documents by using the <FORM> element.
- ◆ Scripting languages or CGI program does the processing of data
- ◆ Server push and client pull are Netscape 2.0's innovative new techniques for automatically updating Web pages.
- ◆ The BACKGROUND attribute fills the background of the browser window with the specified graphic image file.
- ◆ Any text within the <MARQUEE>...</MARQUEE> tags scroll sideways.

9.10 Brain Storm

1. Discuss the result of a GET or POST value to the METHOD Attribute.
2. What is the significance of the NAME Attribute
3. Explain Server push and Client pull
4. How do you secure your information?
5. What are the different types in which a background graphic can be put in.
6. How can we restrict the background sound in our page?
7. MARQUEE offers an incredible amount of control over the way our marquee looks and acts - Discuss

Lab Unit (2 Real Time Hours)

Using Tables design a web page of Yahoo mail options.

This is a web page designed by myself Mr./Ms.on

Personalization	Mail Management	Delivery Services
Account Information Change your password and edit user information. Update your profile.	Block Addresses Block addresses from which you don't want to receive mail.	Yahoo! Delivers Discounts and special offers direct to your Inbox. Personalize your interests!
Mail Preferences Customize your Inbox view. Change your outgoing name and address. Get a bigger mailbox!	Check Other (POP) Mail Retrieve mail from all your other accounts into your Yahoo! Mailbox!	Yahoo! Subscriptions Have Yahoo! content delivered directly to your Inbox.
Signature Attach a custom signature to your outgoing messages.	Filters Sort your incoming mail automatically into designated folders or to your mobile device. Filter out unsolicited email.	Reminders Set up Email, Messenger, or Pager reminders for any event!
Vacation Response Send a custom, automatic message response when you are away.	POP Access & Forwarding Use Yahoo! as your permanent email address. Forward to another mail account, or download your Yahoo! messages to your POP3 mail client.	

Click here to see my previous exercise

Lecture 10

Programming Techniques

Objective

This lecture provides an insight into the following:

- ❖ The concept of data and Programs
- ❖ The various stages in a programming process
- ❖ The concept of Programming Specification

Lecture - 10

- 10.1** *Snap Shot*
- 10.2 Data
- 10.3 Steps in Programming Process
- 10.4 Programming Specification
- 10.5 Short Summary
- 10.6 Brain Storm

10.1 Snap Shot

A program is a set of instructions to be executed by the computer to accomplish a particular task. It is a multi-set process involving identification and definition of a problem; developing computer solutions for the same and preparing a sequence of instructions that can be run on the computer. We shall begin with defining the basic unit of a program, data.

10.2 Data

The term computer programming is defined as “the entire series of steps involved in solving a problem on a computer. The basic unit of a computer program is Data. The term data comes from the word datum, which means a fact. In computers, data is simply the value assigned to a variable. The term variable refers to the name of a memory location that can contain only one data at any point of time.

For example, in a business environment, data can be the number of hours worked, name of employees, stock inventory levels or employee pay details. The data can be of different types. The data types can be Character, Number or Boolean.

Information is a set of processed data that convey the relationship between data considered. Processing means to do some operations or computations on the data of different variables to relate them so that these data, when related convey some meaning. Thus, information is a group of related data conveying some meaning.

10.3 Steps in Programming Process

The programming process consist of developing and writing sequences of instruction to solve problems on the computer, from the initial rough problem statement, to the finished, verified and documented computer program. We shall now consider the phases involved in the development of correct and efficient computer programs. Details of the work that must go on prior to the actual coding of the program such as specification, organization and design; the work involved during the coding of the program, and the work after the program is implemented, such as verification, documentation and maintenance are covered. To be specific the steps or phases in the programming process can be listed as follows:

1. Program Specification
2. Problem Definition
3. Requirement Analysis
4. Design a Program Model
5. Determine the correctness of an Algorithm
6. Code Program

7. Program Testing
8. Debugging
9. Documentation

10.4 Programming Specification

Programming or Problem Specification includes two activities namely the requirement specification and developing a clear, concise, and unambiguous statement of the exact problem to be solved, in other words the problem definition.

10.5 Short Summary

- ◆ The entire series of steps involved in solving a problem on a computer is a computer program
- ◆ The basic unit of a computer program is Data
- ◆ The term data comes from the word datum, which means a fact
- ◆ Information is a set of processed data that convey the relationship between data considered
- ◆ Problem Specification includes requirement specification and problem definition.

10.6 Brain Storm

1. Define “Computer Programming”
2. Relation between data and a variable
3. Discuss the basic steps involved in a programming process
4. In what terms do we define a given problem
5. Discuss the importance of planning in a programming process
6. Discuss the components of the Problem Specification phase

Lecture 11

The Programming Process

Objective

This lecture provides an insight into the following:

- ❖ Need for Problem definition
- ❖ How do you define a problem
- ❖ The concept of Requirements Analysis
- ❖ Components of Requirements Analysis
- ❖ Designing a problem model

Lecture - 11

- 11.1 *Snap Shot*
- 11.2 Problem Definition
- 11.3 Requirement Analysis
- 11.4 Design a Program Model
- 11.5 Short Summary
- 11.6 Brain Storm

11.1 Snap Shot

Schedule slippage, cost overruns, poor quality and higher maintenance needs for programming are caused by the lack of planning. At the beginning of the programming, precise information concerning the programming goals, customer needs and product constraints is not available. This makes the early planning an impossible one. Even then the purpose of planning is to clarify goals, needs and constraints. This is achieved by defining the problem to be solved in a clear and unambiguous manner. Also, the technical requirements of the program are specified completely and concisely in an unambiguous manner. Formal notations are used as appropriate. The process of Design according to Webster, involves “conceiving and planning out in the mind” and “making a drawing pattern or sketch of”.

11.2 Problem Definition

We must know exactly what we want to do before we can begin to do it. But this phase is too often overlooked or omitted by programmers. A clear understanding of exactly what is needed is absolutely necessary for creating a workable solution. This phase involves developing and clarifying the exact specification of the problem. The problem definition should be in user language, and the problem should be described from the user’s point of view. The problem definition is typically expressed in English or some other natural language and may incorporate charts, figures, tables, and equations of various kinds. The exact notations used in the problem definition are highly dependant on the problem area.

Problem definition requires a thorough understanding of the problem domain and the problem environment. Techniques for gaining this knowledge include customer interviews, observation of problem tasks and actual performance of the tasks by the planner. The planner must be highly skilled in the techniques of the problem definition because different customer representatives will have different viewpoints, biases and prejudices that will influence their problem area. In addition, customer representatives are seldom able to formulate their problems in a manner that yields to logical, algorithmic analysis.

11.3 Requirement Analysis

Requirements describe in detail what a program is supposed to do, and constitute the first step toward a solution. This basically is an input/output document. Explicit requirements keep you from guessing what the user wants. Specifying requirement adequately is a key to program success.

To completely specify the input to a program we need to provide the following five pieces of information

1. What values will be provided to the program in what order?
2. How many input values will there be and what is the end of the input?

3. What is the format of data including type, accuracy and units?
4. What is the input device?
5. What is the legal range of allowable inputs including an upper and lower limit?

To specify the output of a program completely, we need to provide answers to the following four questions

1. What results are needed from the program - This has to be clearly and unambiguously specified
2. What format is needed for these values including type, accuracy and units?
3. What is the output device?
4. How these values are to be displayed including spacing heading titles and layout?

In addition, it is important that we itemize special conditions that must be checked and that require correction and recovery. The input, output and special processing specification represents the most important information collected during the problem definition phase.

11.4 Design a Program Model

Having got a clear concise and unambiguous problem statement we ready to design and build a program to solve it. An algorithm is the specific method used to solve a problem. We will define an algorithm as an ordered sequence of well-defined and effective operation that, when executed, will always produce a result and terminate in a finite amount of time. By “ordered sequence” we mean that after the completion of each step in the algorithm, the next step is unambiguously defined.

An algorithm is a procedure for performing a particular task. The algorithm must have the following characteristics

1. Upon completing the execution of each step, we will always know the identity of the step to be executed next
2. There is a single clearly defined starting point and one or more clearly defined stopping points
3. In all cases, the algorithm will terminate after a finite number of steps
4. The algorithm is composed of effective primitives whose meaning is clear and unambiguous to the persons or machine executing it.

Tools used in developing a solution and in the preparation of an algorithm are flowchart and pseudo code among others. Flowcharts provide a visual and graphical representation of the solution while pseudo code means writing the program logic in simple English-like language. Logic depicted using these tools can then be written using any programming language. In other words these are generic tools.

11.5 Short Summary

- ◆ We must know exactly what we want to do before we can begin to do it
- ◆ Problem Definition involves developing and clarifying the exact specification of the problem
- ◆ Specifying requirement adequately is a key to program success
- ◆ The input, output and special processing specification represents the most important information collected during the problem definition phase
- ◆ An algorithm is the specific method used to solve a problem
- ◆ Tools used in developing a solution and in the preparation of an algorithm are flowchart and pseudo code among others

11.6 Brain Storm

1. What are the important points to be considered when defining a problem
2. List some special conditions that are required during the requirement analysis
3. “Algorithm is the best method to solve a given problem effectively” - Can you justify this statement

The Programming Process

Objective

This lecture provides an insight into the following:

- ❖ The efficiency of an Algorithm
- ❖ Coding phase of the program development
- ❖ Structured coding as the efficient way to coding
- ❖ The concept of testing the program
- ❖ The different types of Program testing

Lecture - 12

12.1 *Snap Shot*

12.2 Determine the correctness of an algorithm

12.3 Code Program

12.4 Program Testing

12.5 Short Summary

12.6 Brain Storm

12.1 Snap Shot

In the previous lecture we have seen that the best way to design a given problem is by algorithms. It is very important that we ensure that the algorithm we have designed is efficient. The actual coding of the design follows this phase. The program is not complete once we have completed the coding process. The code has to be tested to demonstrate that the program meets its requirements.

12.2 Determine the correctness of an Algorithm

One of the most difficult, and sometimes most tedious step in the development of an algorithm is ensuring and proving that the algorithm is correct. Correctness of an algorithm or a program is a measure of the output that is obtained and determining if it does what it is supposed to do. This can be checked by any one of the following methods

- ◆ Dry Run
- ◆ Independent inspection
- ◆ Structure walk-through

12.3 Code Program

Only after unambiguously defining the problem, organizing a solution, and sketching out step-by-step details of the algorithm can we consider beginning to code. Your choice of which computer language to use will probably be dictated by three considerations, such as the nature of the problem, the programming languages available on your computer and limitation of your particular computer installation.

Structured coding is the most effective way to realistic coding. The advantages of structured coding are many and in fact it is the application of principles of the top-down design and stepwise refinement to the implementation of individual program modules.

Structured coding achieves clarity and readability within individual program modules and thus increases the maintainability of that module. Structured code helps during debugging and testing of individual programs. We shall deal with Structured programming at a later stage

12.4 Program Testing

Program Testing, also called empirical testing, involves determining the correctness of a program by trying it on a large number of carefully chosen data sets and then seeing if it produces correct answers in all those cases. If debugging can be said to be

the process of “finding the error you know is there”, then testing can be viewed as “finding the error you don’t know is there”. There are three types of Program Testing

Module testing

Module testing or Unit testing checks for errors in each individual module and the errors are corrected for each individual unit. The testing may be conducted in parallel for several modules at a time. Here the common everyday functions of the program unit, i.e., each legal data values on which the program operates are tested.

Testing of the program on null case is also done and it is important to make sure that your program works correctly for the degenerate case of “nothing to do”. The program is then tested for how it responds in the area of illegal data. And finally we test the program’s behaviour under the stress of extreme conditions in either its operating environment or the quantity of data we provide.

System Testing:

Errors may exist in the interface between two modules, and this fact may not become apparent until they are combined and tested as a complete unit. Therefore, additional testing of the overall program is essential for determining whether the correct individual modules fit together properly into a correct whole. This phase of testing is typically called system testing. Errors not discovered in unit testing but discovered in system testing

- a. Errors in the interface between two modules
- b. Errors or misunderstandings in interpreting the specifications of a module
- c. Errors caused by the side effects of a module

Acceptance Testing

The very last phase of testing is called acceptance testing, and this is where we determine whether or not the program is considered finished and can be released for general use. Acceptance testing differs from unit or module testing and system testing in two very important ways

- a. It is typically done by the end-user, not the programmer
- b. It is carried out without the knowledge of the internal structure and organization of the program

Thus, acceptance testing is not simply more testing but testing from a different viewpoint, which can more extensively check such aspects of the program as user friendliness, robustness and on-line assistance.

12.5 Short Summary

- ◆ Correctness of an algorithm or a program is a measure of the output that is obtained
- ◆ Structured coding is the most effective way to realistic coding
- ◆ Testing can be viewed as “finding the error you don’t know is there”
- ◆ Module testing or Unit testing checks for errors in each individual module of the program
- ◆ Errors that exist in the interface between two modules are checked using System Testing
- ◆ Acceptance Testing determines whether or not the program is considered finished and can be released for general use

12.6 Brain Storm

1. In what terms do we relate the efficiency of algorithms
2. During Program coding, what are the important aspects that must be looked into?
3. Significance of the various types of program testing

Lecture 13

The Programming Process

Objective

This lecture provides an insight into the following:

- ❖ The process of locating and correcting errors in a program
- ❖ The types of errors - syntax errors, run-time errors and logic errors.
- ❖ Need for documentation and its types
- ❖ The concept of structured programming
- ❖ Introduction to the basic control structures

Lecture - 13

- 13.1 *Snap Shot*
- 13.2 Debugging
- 13.3 Documentation
- 13.4 Structured Programming Techniques
- 13.5 Short Summary
- 13.6 Brain Storm

13.1 Snap Shot

In this lecture we begin with the process of determining the correctness of the program. This is very important to assure that the program implemented demonstrates accuracy. Here we look into the three types of errors that are commonly encountered, namely Syntax Errors, Runtime Errors and Logic Errors. One most important entity in any programming process is the documentation. This is not a single-phase process and is to be present in every single phase of the programming process. As we have seen earlier, the best method of programming is the structured effort. All these are discussed in this lecture.

13.2 Debugging

Program implementation is getting the code working correctly. Correctness is not two-step process of having an incorrect program immediately making it correct. Debugging is the process of locating and correcting errors in a program in the presence of explicit and demonstrated incorrect behavior. The most important thing to remember about debugging is "The best debugging tool in the world is the avoidance of bugs from the beginning. The easiest bug to correct is the one that never got into your program in the first place."

The approaches to debugging can be classified according to the errors encountered. The following sections treat syntax errors, run-time errors and logic errors.

Syntax Errors

A syntax error is any violation of the syntactic rules of the language. Syntax errors are common and relatively easy to correct. These errors are encountered during compilation. There is usually not a one-to-one correspondence between error messages from the compiler and corrections to be made. Often a single error will generate multiple error messages. Frequently, the compiler overlooks a syntax error because of the presence of other syntax errors on the same line. Therefore, it may require more than one additional run to eliminate all of them.

Run-time Errors

Even when a program unit is syntactically correct, there is still the possibility of a run-time error, which causes abnormal program termination. Common run-time errors include the following:

- ◆ Using a variable before assigning a value to it
- ◆ Dividing by 0
- ◆ Assigning values outside the prescribed bounds of the declared type of a subrange variable

- ◆ Using an index outside the prescribed bounds of the array's lower and upper bound
- ◆ Reading a piece of input data whose syntax is improper for the declared data type of the variable in the read list. (e.g., a letter within an integer value)

Logic Errors

Logical errors in an incorrect translation of either the problem statement or the algorithm, which causes the program to produce incorrect results. These errors can be extremely difficult to locate, certainly much more difficult than the syntax and run-time errors mentioned. The main difficulty is that the computer will not print any error-message and will produce the wrong result. These errors can be detected by doing a dry run.

13.3 Documentation

The documentation of the program is a continuous process. The program specifications, the design of a model and the code program itself can all be considered part of the documentation of a program. Documentation falls into two distinct classes – user documentation and technical documentation

User documentation is needed solely for end-users. These people may or may not be familiar with computer or even the program they are trying to run. Therefore, user documentation concentrates almost exclusively on the input/output characteristics of the program and presents a general overview of what it does. The major piece of user documentation is a separately bound book, write-up called the user's guide or user's manual.

Though the exact format and contents are a matter of personal writing style, but the following information are to be included – The program name, the input data required by the program, the normal output of the program when presented with valid data, the exception reports, program limitations and the name, address and telephone number of the person responsible for providing assistance.

Technical documentation is the material the maintenance programmer needs to change, correct or understand the program. This material is intended for technical specialists. The structure of the program, not simply its input/output characteristics, is of primary importance to these people.

Technical documentation must describe two distinct aspects of a program – low level coding details and high-level program structure. Again the exact format of a program design documents is a matter of personal taste and company policy but it should include the following information – Program name and purpose, the historical development of the program and its current status, the overall program

structure, description of each module, description of key data structures, built-in maintenance aids and testing/acceptance criteria.

13.4 Structured Programming Techniques

Structured programming technique involves constructing individual program units. The single most important design aid is the technique called top-down program design. It is also known as modular development or stepwise refinement. In a top-down design, we initially describe the problem we are working on at the highest, most general level. The description of the problem at this level will usually be concerned with what must be done – not with how it must be done. We must take all of the operations at this level and individually break them down into simpler steps that begin to describe how to accomplish the tasks.

Structured programs are designed using three basic control structures

- a. Sequential: Steps are performed in a strictly sequential manner, each step being executed exactly once
- b. Selection: One of several alternative actions is selected and executed
- c. Repetition: One or more steps are performed repeatedly.

These three control mechanisms are individually quite simple, but in fact they are sufficiently powerful that any program can be constructed using them.

13.5 Short Summary

- ◆ Debugging is the process of locating and correcting errors in a program in the presence of explicit and demonstrated incorrect behaviour
- ◆ The approaches to debugging can be classified according to the errors encountered – Syntax, Run-Time or Logic errors
- ◆ The documentation of the program is a continuous process
- ◆ Documentation falls into two distinct classes – user documentation and technical documentation
- ◆ Structured programming technique involves constructing individual program units
- ◆ Top-down program design is also known as modular development or stepwise refinement
- ◆ The three basic control structures are sequential, selection and repetition

13.6 Brain Storm

1. Differentiate between syntax, runtime and logic errors
2. Documentation of a program is a continuous process - Explain
3. Advantages of structured coding

Lecture 14

Program Tool

Objective

This lecture provides an insight into the following:

- ❖ The pictorial representation of an algorithm
- ❖ The basic symbols in a flowchart
- ❖ Need for structured programming
- ❖ An introduction to the basic structures in structured programming

Lecture - 14

- 14.1 Snap Shot
- 14.2 Program Tool - Flowchart
- 14.3 Why Structured Programs
- 14.4 Structures
- 14.5 Short Summary
- 14.6 Brain Storm

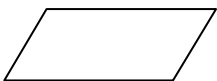
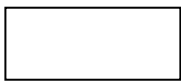



14.1 Snap Shot

In the previous lectures we looked at the various steps in a programming process. We also discussed that algorithms can be represented in pictorial form (flowcharts). Here the representation schemas used are of fundamental importance. Good notation can clarify the interrelationship and interactions of interactions, while poor notation can complicate and interfere with good programming. Let us now look at Flowcharts in detail

14.2 Program Tool - Flowchart

A flowchart is a pictorial representation of logic required to accomplish a task. It includes all necessary steps to be included in a program. It is named so, because it charts the flow of a program. It is a symbolic representation of input, output and the processing steps. Besides being a good method of writing down the algorithm, it is also a part of program documentation and helps in understanding, debugging and maintenance of programs

In the flowcharting technique, each action is represented by an appropriate symbol. Each symbol is connected to the other by arrows to illustrate the sequence of operations. The following are the basic symbols used in a flowchart

Symbol	Purpose
	This symbol is used to represent the input/output operation (I/O operations) such as read and write.
	The process symbol is used to represent processes like assigning a value to a variable or adding a number.
	This terminator symbol indicates the beginning or the end of the flowchart
	This connector symbol is used to maintain links between two or more flowcharts when the flow chart runs longer than one page or when the same diagram is continued at different locations on the same page. It is indexed with a marker.
	This flowline symbol represents the direction of flow of data in a flowchart. These are straight lines with arrowheads. These are normally drawn from top to bottom and left to right or right to left.

14.3 Why Structured Programs?

Novices are inclined to ask this question for they are satisfied as long as their programs run and give the expected result. If however, your programs are to be used by another person, which is usually the case, then your idiosyncratic approach to programming would seal the fate of your program. Programs, which are structured consciously, offer a host of advantages over the unstructured ones.

Program coding does not end with getting the required results, the code has to be maintained and improved. The principles of structured programming, when applied will help programmers to easily modify and adapt the program for a particular situation.

Structured coding achieves clarity and readability within individual program modules and thus increases the maintainability of that module. Structured coding allows the coding of the outer block first and then the inner blocks. Each successive block allows us to narrow our focus and concentrate on finer and finer details of the program. Thus we work from the general aspects of the code to the more specific.

14.4 Structures

Structured programming involves the deployment of the following five basic structures in coding of algorithms. Using any other structure violates the principles of structured programming. Incidentally, two Italian computer scientists, Bohm and Jacopini, proved it way back in 1964 that any program however complex can be written using only the five basic structures to be discussed here

- ◆ Sequence program flow
- ◆ Decision (two branch) program flow
- ◆ Decision (single branch) program flow
- ◆ Pre-test iteration program flow
- ◆ Post-test iteration program flow

14.5 Short Summary

- ◆ Flowchart is a symbolic representation of input, output and the processing steps
- ◆ The basic symbols in a flowchart are symbol to represent input/output, process symbol, terminator symbol, connector symbol and flowline symbol
- ◆ Program coding does not end with getting the required results, the code has to be maintained and improved
- ◆ Structured coding allows us to work from the general aspects of the code to the more specific

14.6 Brain Storm

1. Construct a simple flowchart to illustrate the use of its various symbols
2. Compare and contrast Structured Programming with the unstructured programming
3. Any program however complex can be written using only the five basic structures given above - Discuss

Basic Programming Structures

Objective

This lecture provides an insight into the following:

- ❖ The sequential flow of program
- ❖ Need and types of decision structures
- ❖ The concept of Iteration and its types

Lecture - 15

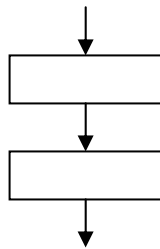
- 15.1 Snap Shot
- 15.2 Sequence Program Flow
- 15.3 Decision Structure
- 15.4 Iteration Structures
- 15.5 Short Summary
- 15.6 Brain Storm

15.1 Snap Shot

Structured coding achieves clarity and readability within individual program modules and thus increases the maintainability of that module. The structured coding of an algorithm involves five basic structures. Apart from the sequential flow of program it is often necessary to alter the flow of the program sequence. This arises when certain conditions are to be tested before proceeding in a particular sequence. Repetition of a part of a program for a specified number of time or until a particular condition is satisfied also commands importance in programming. This lecture covers the basic structures of structured programming in detail.

15.2 Sequence Program Flow

This is the simplest of the structures used in structured programming. Several logically coherent statements can be grouped under a single sequence structure provided they are all simple statements as in the case of assignment statements or declaration statements. In fact, all other structures can be represented by this primitive structure.

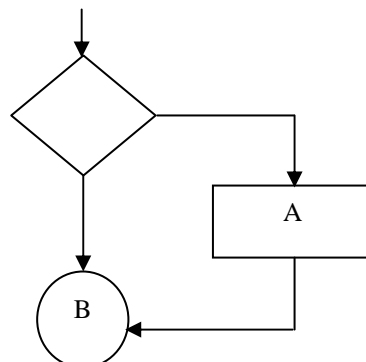


15.3 Decision Structure

In many cases we may require to alter the course of program flow. This structure serves the purpose. There are three variants of this structure viz.

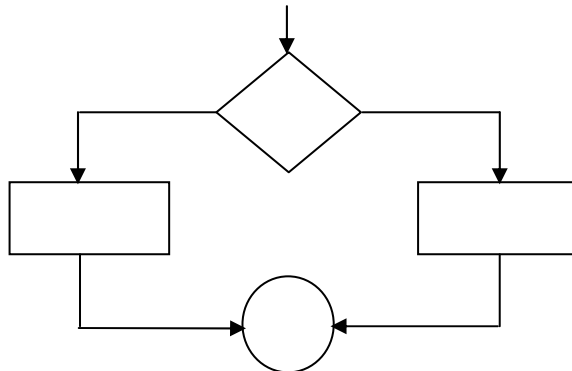
15.3.1 One Way Branch

The one-way branch decision structure as depicted below serves to transfer control on one result of the decision to block A and then to B which is the place where the control is transferred if the result of the decision is otherwise. This structure is mostly represented as IF...THEN in pseudo code.



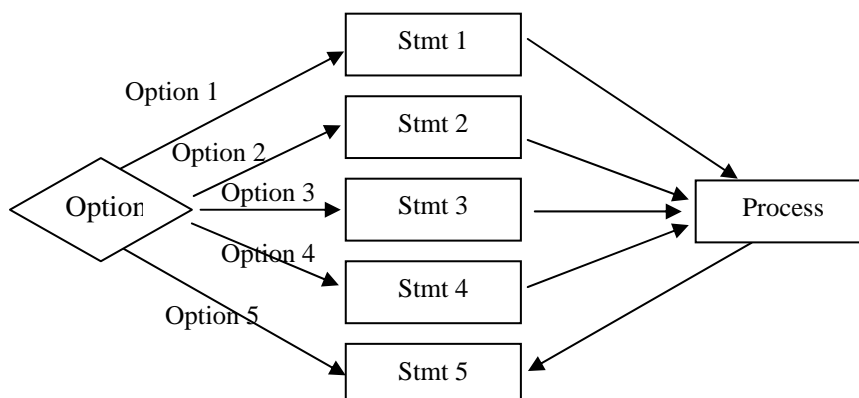
15.3.2 Two-way branch decision structure

This variant of the decision structure helps in executing one of the two alternative blocks depending upon the result of a decision and then continues execution in the normal sequence. This construct is implemented in the famous IF...THEN...ELSE construct.



15.3.3 Case or Multi-way decision structure

In this structure depending upon the value of a key (a variable or expression) the control branches to one of the many alternatives. This structure is compact in form but can also be substituted by a series of the decision structures discussed.

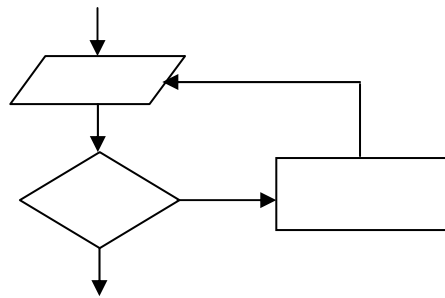


15.4 Iteration Structure

Iteration is the process of repeating a statement or a group of statements a specified number of times. Iteration structures are most used of the structures and are powerful in nature. Iteration structures come in two flavours, viz.

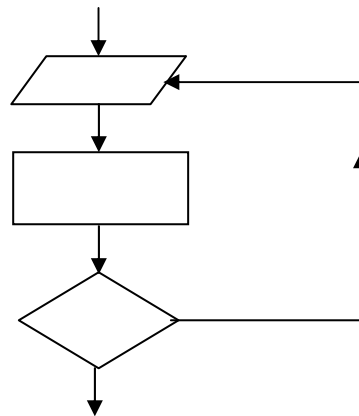
Pre-test iteration structure

This involves a check before the iteration commences. Therefore there is a possibility that the block is never executed if the condition is not valid for iteration at the very first encounter. A good example of such a structure is the FOR loop.



Post-test iteration structure

This structure differs from the pre-test iteration in the fact that the iteration termination condition is tested only after the block is executed. This ensures that the block is executed at least once because the iteration termination condition is tested only after the execution of the block. This is implemented as DO...WHILE and REPEAT...UNTIL loops.



15.5 Short Summary

- ◆ Logically coherent simple statements can be grouped under a single sequence structure
- ◆ One way branch structure is mostly represented as IF...THEN in pseudo code
- ◆ Two-way branch decision structure is implemented in the famous IF...THEN...ELSE construct.
- ◆ Case or Multi-way decision structure is used to branch control depending upon the value of a key
- ◆ Iteration is the process of repeating a statement or a group of statements a specified number of times
- ◆ Pre-test iteration structure involves a check before the iteration commences

- ◆ In post-test iteration structure the iteration termination condition is tested only after the block is executed

15.6 Brain Storm

1. Explain the significance of Decision structures
2. Compare and contrast pre-iteration structure with post-iteration structure
3. What is the need for decision and iteration structures

Structured Charts and Pseudo code

Objective

This lecture provides an insight into the following:

- ❖ Representing structured relationship among modules
- ❖ Differential aspect of structure chart and flow chart
- ❖ Structured programming through Pseudocode
- ❖ An example for a Pseudocode

Lecture - 16

- 16.1 Snap Shot
- 16.2 Tools for Structured Programming
- 16.3 Short Summary
- 16.4 Brain Storm

16.1 Snap Shot

To write structured programming, tools are of utmost importance. There are a great many number of methodologies for displaying the overall design structure like the structured charts, the data flow diagrams, bubble charts, HIPO (hierarchy-input-process-output) Charts, procedure templates and decision tables.

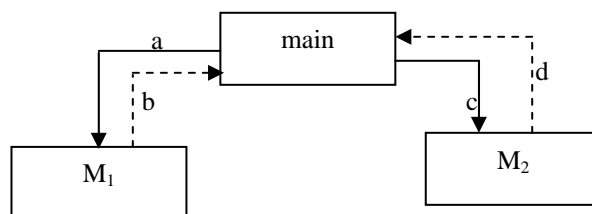
16.2 Tools for Structured Programming

Structured Charts

This is a pictorial representation of the structural relationships among modules and between modules and data structures. Consider that a very low-level routine is called by a great number of modules throughout the system then it cannot be represented by a rigidly hierarchical tree structure.

We shall see a highly simplified version of the above named techniques, which uses a directed graph to represent the overall design. Here any module may be related to any other module. A structure chart differs from a flow chart in two ways: a structure chart has no decision boxes and the sequential ordering of tasks inherent in a flow chart can be suppressed in a structure chart.

Consider the notation below, which represents a program structure.



This contains three modules, called main, M₁ and M₂. Modules M₁ and M₂ are activated by main and return to it. M₁ is given the value a and returns the result b while M₂ is given the value c and returns the result d.

Pseudocode

Pseudocode (sometimes called metacode) is a shorthand notation for the control structures and other programming entities deliberately kept small and incomplete.

The Pseudocode can be written in simple English without any bias towards any specific programming language. The steps should convey their meaning and should desirably avoid any language dependent feature.

If a programmer adheres to the rule of not introducing any language dependent information, the same Pseudocode can be given to programmers working with

different programming languages. This thereby increases the productivity and cuts short the development time. The Pseudocode goes one step further after the structure charts to explain the method of solution. The Pseudocode is easily maintained and modified by using a text editor.

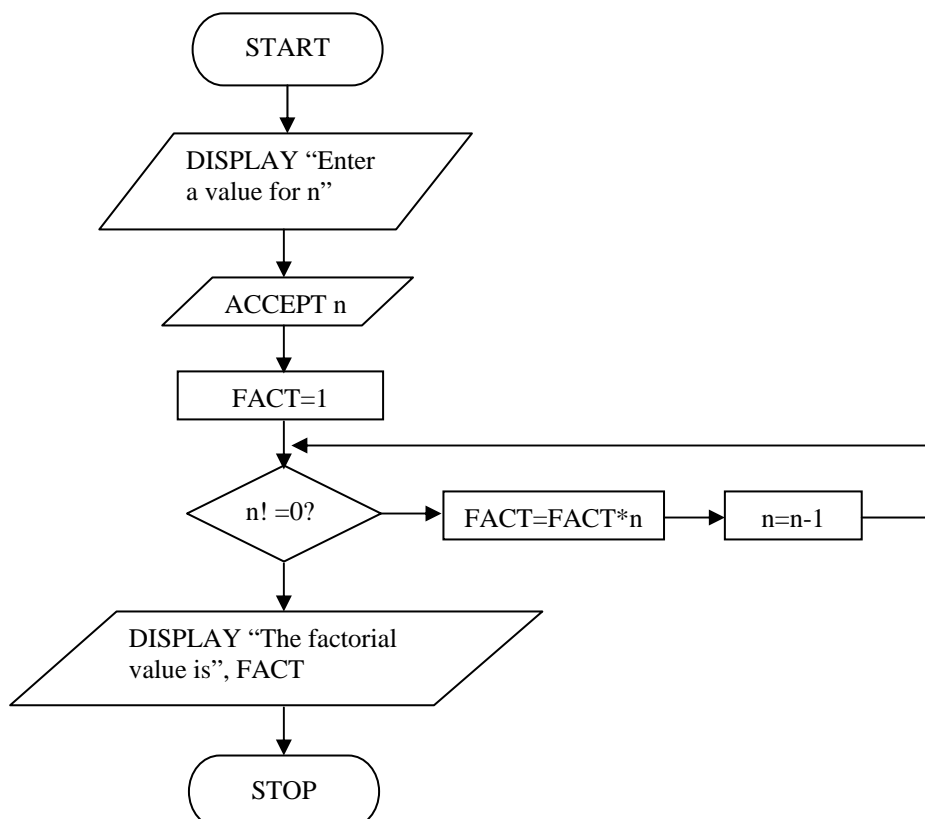
An example below, illustrates both Pseudocode and flowchart. Following is a Pseudocode to find the factorial for any given number. Here the factorial of a number, which is the product of the positive integers 1,2,3 upto and including the given integer, is calculated. Factorial is derived by applying the following equation:

$$n! (\text{factorial of } n) = n * (n-1) * (n-2) * (n-3) \dots\dots *3*2*1$$

Pseudocode

```

START
DISPLAY "Enter a value for n"
ACCEPT n
FACT = 1
WHILE n!=0
DO
    FACT=FACT*n
    N=n+1
ENDDO
DISPLAY "The factorial value is:", FACT
END
  
```



16.3 Short Summary

- ◆ Structure charts are used during architectural design to document hierarchical structure, parameters and interconnections
- ◆ A structure chart has no decision boxes
- ◆ Pseudocode is also called metacode
- ◆ Pseudocode is a shorthand notation for the control structures and other programming entities deliberately kept small and incomplete.
- ◆ The Pseudocode is easily maintained and modified by using a text editor

16.4 Brain Storm

1. What are Structure Charts
2. Differentiate between structure chart and flow chart
3. Explain the significance of Pseudocode

Lecture 17

Structured Programming Issues

Objective

This lecture provides an insight into the following:

- ❖ Major issues in structured programming
- ❖ The concept of program maintenance
- ❖ Events that cause changes in the user programs
- ❖ Techniques that reduce maintenance cost
- ❖ Portability of a program
- ❖ Importance of the readability of a program
- ❖ Formal and Informal verification
- ❖ Modularity of a program
- ❖ Issues regarding the size of a program module

Lecture - 17

- 17.1 Snap Shot
- 17.2 Structured Programming Issues
- 17.3 Maintenance
- 17.4 Portability
- 17.5 Readability
- 17.6 Program Verification
- 17.7 Modularity
- 17.8 Short Summary
- 17.9 Brain Storm

17.1 Snap Shot

Getting results from a program is not enough. We must guarantee that they are correct results. Programs are not static entities. They frequently become outdated as errors are discovered, new problems need to be solved, or new equipment becomes available. So we need to determine if the program we have written will survive the test of time. It should also be considered that the program we have developed is independent of any machine environment. The readability of a program should also be ensured. All these issues are dealt with in this lecture.

17.2 Structured Programming Issues

We will now look into the major issues concerning structured programming for better insight towards developing good application software. These include

- ◆ Maintenance
- ◆ Portability
- ◆ Readability
- ◆ Program verification
- ◆ Modularity

17.3 Maintenance

Program maintenance is the process by which programs are corrected and updated to reflect changes in the problem specification. If the programs are not structured, you will find it difficult to incorporate changes in your own programs leave alone other programmers who are supposed to maintain them. The most common maintenance operation is tracking down bugs that went unnoticed during the testing phase. Structured programs enable you to maintain your programs with the least effort and can, to a greater extent, be maintained by others better than the unstructured ones. Events that cause changes to the original program include:

Newly discovered bugs:

Errors that were not detected during the original testing phase may be discovered during the actual program operation

Program improvements

User's may not be happy with the way the program currently operates. The program needs to be improved to rectify user complaints

Specification changes

The original problem changes because of new laws, new discoveries, different users or changing consumer demand. The program must be modified to reflect this change.

New Equipment

The program must be written to take advantage of a newer computer or compiler

The time and costs involved with program maintenance are usually grossly underestimated by most programmers. Too often they think only of the one-time costs associated with the original design and implementation.

Studies have shown that typical programmers will spend only about 40% of their time on designing and implementing new software, while the other 60% of their time is spent on maintaining or enhancing existing software products.

The magnitude of these maintenance costs has led to a demand for techniques that reduce maintenance cost, speed up changes and minimize the risk of introducing errors. In reviewing these techniques, we can now appreciate why they are so critical.

Clarity and reliability

The program listing is the primary piece of technical documentation. The clarity of that listing significantly affects the time it takes to implement a program change

Portability and generality

Software tends to last longer than hardware. Therefore, a program written with portability in mind will create much less havoc when the inevitable new computer arrives. Likewise, when user needs change, a truly general program will require less programmer time to adapt the new specification

Structured Code

The most common maintenance operation is tracking down bugs that went unnoticed during the testing phase. A program implemented as a series of singly entry-singly exit blocks will facilitate debugging and testing

Robustness

A robust program will not “bomb out” without any helpful error message. Instead, it will terminate gracefully with useful information that will help the maintenance programmer or the user to locate and correct the problem.

Logical coherence and independence

When a change can be localized to a single module, it makes maintenance much easier. We do not need to worry about the effects of that change rippling throughout other modules in the system.

Debugging and testing

A formalized debugging and testing scheme reduces the likelihood of undetected bugs getting into the finished program, thus reducing the maintenance required on that program.

Documentation

The technical documentation of a program is necessary because many other programmers will be working with and maintaining that program over its life span.

The stylistic guidelines and implementation methods we have stressed make the most sense when these key points are remembered

Programs change often. They are not static entities

Programs tend to be used for long periods of time, typically 5, 10, even 15 years. The people that developed the original program may not be available to explain or clarify the design

A program will be maintained by many different programmers over its lifetime, and most of them will be initially unfamiliar with its contents or structure

The most expensive component of any computer system is personal costs

17.4 Portability

Portability implies program independence from the hardware, or a specific operating system or compiler. A portable program is independent of any particular machine environment. On the other hand, generality implies program independence only from a particular data set.

The portability is a critical characteristic because of the rapid changes in the computers and also due to exchange and sharing of programs. The most important guideline in writing portable programs is strict adherence to the standard version of a programming language.

The group that directs most of the standardization efforts is the International Standards Organization (ISO), an agency of United Nations. ISO publishes a standard that specifies the syntax and semantics (actions) of each statement in the program. A standard compiler for a language accepts and correctly compiles the standard language. But if we use a language feature that is a local extension to our compiler, there is no guarantee that our programs will run when moved.

Also, we should avoid the machine dependent features that could present our program from running on another computer system.

17.5 Readability

It is found in general that it takes a longer time to read a program and understand it than writing it out. Therefore one has to make his or her programs more readable i.e., easily comprehensible to be useful. Structured programs are found to be more readable because, the constructs by themselves are quite natural in expressing any logic. It is also required of the programmer to follow a generally accepted stylistic pattern of coding to make the program more readable. The programs have to be well documented to clarify things, which are not obvious. Both scant and verbose documentation don't add to the readability of programs. It is important to make your comments short and crisp.

17.6 Program Verification

Testing can only detect the presence of errors; it can never prove their absence. All we can say about the program is that for a wide range of carefully chosen data cases, it has worked properly and, from this experience, we extrapolate to the statement that the program will work correctly under all circumstances, even for those test cases that were not explicitly tested. This extrapolation is not based on either mathematical formalities or physical laws but on empirical observation and human judgement.

This is a very weak form of correctness, which is quite different from the type of correctness proven in such classical discipline as mathematics or physics. Our technique of formal verification is based on describing, for every statement S in a programming language, what conditions we know to be true before S is executed, called the preconditions, and what conditions we know to be true after S has been executed, called the post conditions. This is usually written as follows:

$$\{P\} S \{Q\}$$

where

P are the Boolean preconditions

Q are the Boolean pos conditions

S is any statement in the language

Together $\{P\}$ and $\{Q\}$ are called assertions because they assert that a given condition is true at a given point in the program. The important thing, however, is not to make an assertion about the behaviour of a single statement, but to study the behaviour of entire programs. This can be done by using the composition laws of Boolean logic, which say that

$$\{P\} S_1 \{Q\} \text{ and } \{Q\} S_2 \{R\} \text{ implies } \{P\} S_1 : S_2 \{R\}$$

Thus, by knowing the preconditions and post conditions associated with individual statements, we can begin to make claims about what happens when sequences of statements are executed – that is, when we run a program. In the above example, if we know that the Boolean predicate $\{P\}$ is initially true, then when we execute the “program,” $S_1 : S_2$ we know that condition $\{R\}$ will be true upon completion.

Until formal verification of every program unit becomes a practical reality, there is one other thing that programmers can learn from these techniques and use in their own modules – the idea of informal verification. Informal verification means that you argue and rationalize the correctness of your program as it is being written, just as with formal methods, but without necessarily using the mathematical notations and proof techniques of formal verification.

Structured programs can be verified theoretically as being correct or otherwise, whereas it is near to impossibility with the other type of programming. Even though formal verification may not, at the present time, be a completely feasible tool for widespread use, there are still a number of realistic things that can be done to minimize the number of bugs that get into the code:

1. Reason logically about the behaviour of code as it is being written. Convince yourself of its correctness immediately, rather than waiting until testing
2. Include, as comments in the code, your assertions about the program’s behaviour and the reasoning you used to justify the correctness of the code.
3. Show these assertions to one or more professional computer scientists and convince them, by reviewing your arguments that your assertions are indeed correct

These steps, along with a well-planned and well-organized empirical testing scheme, will go a long way toward eliminating bugs before they occur, quickly and efficiently removing the few that are present and ensuring the correctness of the finished software product.

17.7 Modularity

Modularity is the attribute that says that a program can be decomposed into several independent segments of code or modules. Each module is written separately and tested by using mock data by employing a test driver program. Such a development procedure removes lot of errors that might be hard to do with the complete program.

Modularity is also emphasized by the Top-down approach to problem solving. Generally users feel that modular approach is for large problems, but using the modular approach can efficiently solve even small problems.

How large should a program module be? While it is impossible to fix a specific number of instructions, there are useful guidelines to manage module size. Some

organizations have established rules to manage module size. A common one is that no module should contain more than 50 instructions. Another is that the listing of source code for a module should fit on a single printed page. In some situation, these rules are appropriate, but in others they result in arbitrary decisions that miss the point of managing module size.

Another important feature to consider here is the span of control. Span of control refers to the number of subordinate modules controlled by a calling module. In general, we should seek to have no more than five to seven subordinate modules. It should be noted that excessive span of control meaning a high number of subordinate modules creates considerable overhead in determining which module to invoke under certain conditions.

17.8 Short Summary

- ◆ Program maintenance is the process by which programs are corrected and updated to reflect changes in the problem specification
- ◆ Typical programmers spend only about 40% of their time on designing and implementing new software, while the other 60% of their time is spent on maintaining or enhancing existing software products.
- ◆ A portable program is independent of any particular machine environment
- ◆ Generality implies program independence from a particular data set
- ◆ Programmers need to follow a generally accepted stylistic pattern of coding to make the program more readable
- ◆ There are two types of verification namely formal and informal verification
- ◆ Modularity is the attribute that says that a program can be decomposed into several independent segments of code or modules
- ◆ Modularity is emphasized by the Top-down approach to problem solving

17.9 Brain Storm

1. What are the events that may affect our programs
2. List the techniques that reduce maintenance cost
3. Discuss the portability factor of a program
4. Discuss in general the criteria involved in making a program more readable.
5. Testing can only detect the presence of errors; it can never prove their absence – Discuss
6. What is formal verification?
7. How is modularity implemented in a program
8. Explain Span of Control

Lecture 18

Problem solving Approaches

Objective

This lecture provides an insight into the following:

- ❖ What do you mean by solving a problem
- ❖ Top Down approach to solve a problem
- ❖ Brute Force Approach to problem solving

Lecture - 18

- 18.1 Snap Shot
- 18.2 Problem Solving Approaches
- 18.3 Top Down Approach
- 18.4 Brute-force Approach
- 18.5 Short Summary
- 18.6 Brain Storm

Lab Unit

18.1 Snap Shot

Once we get a clear concise and unambiguous problem statement we are ready to design and build a program to solve it. This involves the intellectual and administration techniques for managing the overall development of large size and complexity of the real world problems. A careful planning and a good program design is required to effectively overcome many problems encountered in large programming project and lead to successful completion of quite large programs.

18.2 Problem Solving Approaches

During the past few years, several techniques have been developed for designing. These techniques include stepwise refinement, levels of abstraction, structured design and top-down structure. Although these techniques are often called “design methodologies” or “problem solving techniques”, they are infact viewpoints and guidelines for the design process.

18.3 Top-Down approach

Top-Down program Design involves starting from the problem specification document and subdividing the original problem into collections of one or more modules. Each of these lower level modules is smaller and simpler than the original task. We proceed from the top-overall high level goals to the bottom detailed low-level solution methods.

This Top-down design approach is not a one-step process. The decomposition and simplification of the problem is performed over and over again on the successive sub modules until finally we are left with a task that is so elementary that it cannot be simplified any further. This repeated simplification of a task into a collection of simpler sub tasks is called stepwise refinement.

18.4 Brute-force approach

Brute-Force approach is one in which the programmer tries to solve the problem at hand as a whole. This works fine when the program is small. When the program has to be extended the programmer has to remember a lot of details and the approach fails the programmer. The approach fast loses ground when a team of programmers is entrusted with a large problem. Thus this method is restored to only when the problem is small and there is absolutely no growth envisioned in the future.

18.5 Short Summary

- ◆ Top-Down program Design involves starting from the problem specification document and subdividing the original problem into collections of one or more modules
- ◆ In Brute-Force approach the problem at hand is solved as a whole

18.5 Brain Storm

1. Discuss the Top Down approach to problem solving with an example
2. What is Brute-Force Approach?

Lecture 19

Program Testing

Objective

This lecture provides an insight into the following:

- ❖ Testing the program developed
- ❖ Black box approach to testing
- ❖ Glass-box approach to testing

Lecture - 19

- 19.1 Snap Shot
- 19.2 Testing methods
- 19.3 Black box Approach
- 19.4 Glass-box Approach
- 19.5 Short Summary
- 19.6 Brain Storm

19.1 Snap Shot

Testing and Debugging can be identified as two separate processes. If debugging can be said to be the process of “finding the error you know is there”, then testing can be viewed as “finding the error you don’t know is there”. In this lecture we shall define the black-box and glass-box approach to testing.

19.2 Testing Methods

We have already indicated that the philosophy behind testing is to find errors. Test cases are devised with this purpose in mind. A test case is a set of data that the system will process as normal input. However, the data are created within the express intent of determining whether the system will process them correctly. For example, test case of inventory handling should include situations in which the quantities to be withdrawn from inventory exceed, equal and are less than the actual quantities on hand. Each test is designed with the intent of finding errors in the way the system will process.

19.3 Black box Approach

The programmer treats the program as something that will accept some input and produces some meaningful information. He does not bother with the internal working of the programs or modules.

In this manner, each of the several program modules can be individually tested and then interfaced together smoothly. The selection of test data is critical and has to be done with care and a fairly good amount of deliberations. The test data should be simple realistic values, which also test for extreme condition. This method is ideally suited for large programs.

19.4 Glass-box approach

This is a method of testing a program with regard to its internal workings. The test data is so chosen that all the control flows of the various structures in a program are examined for correctness and each and every alternative is tested by suitable combination of the input test data.

The glass-box approach requires more effort and is not suited for large programs because of the complexity and magnitude of data involved.

19.5 Short Summary

- ◆ A test case is a set of data that the system will process as normal input
- ◆ In black box approach the programmer does not bother with the internal working of the programs or modules

- ◆ Glass-box approach is a method of testing a program with regard to its internal workings
- ◆ The glass-box approach requires more effort and is not suited for large programs because of the complexity and magnitude of data involved

19.6 Brain Storm

1. What is the need for program testing
2. Explain the black-box approach to program testing
3. Explain the glass-box approach to program testing
4. Compare and contrast the two methods of testing discussed here in this lecture

Lecture 20

Introducing VB Script

Objective

This lecture provides an insight into the following:

- ❖ Client side and Server side programming languages
- ❖ Introduction to the VBScript
- ❖ Using VBScript in your web page
- ❖ Variables and its declaration

Lecture - 20

20.1 *Snap Shot*

20.2 Client-Side and Server-Side programming languages

20.3 The <SCRIPT> Tag

20.4 Variables

20.5 Short Summary

20.6 Brain Storm

20.1 Snap Shot

Microsoft Visual Basic Scripting Edition, the newest member of the Visual Basic family of programming languages, brings active scripting to a wide variety of environments, including Web client scripting in Microsoft Internet Explorer and Web server scripting in Microsoft Internet Information Server.

20.2 Client-Side and Server-Side Programming Languages

VBScript acts as both a client-side and server-side programming language. A client-side programming language is a language that can be interpreted and executed by a browser. Jscript/JavaScript are additional examples of client-side programming languages. When a program written in any of these languages is loaded into a compatible browser, the browser will automatically execute the program.

No scripting language including VBScript has disc-accessing features and so it is accepted to be a safe language to deploy in the web. If you already know Visual Basic or Visual Basic for Applications, VBScript will be very familiar. Even if you don't know Visual Basic, once you learn VBScript, you're on your way to programming with the whole family of Visual Basic languages.

It is a free language and so it can be used anywhere. We can write our VBSCRIPT program in notepad and execute it using Internet Explorer.

You can use the SCRIPT element to add VBScript code to an HTML page.

20.3 The <SCRIPT> Tag

Any scripting language can be enclosed within a script tag in an ordinary HTML file. The script tag can be placed anywhere in the HTML page and it is better to place it in the head tag. We should mention the language as an attribute. It is a container tag.

```
<script language = VBScript>
```

```
</script>
```

20.4 Variables

If the value of a specified memory location changes at run time then it is defined to be a variable. The name with which we assign or manipulate the memory location at runtime is called a variable name.

Variable names follow the standard rules for naming anything in VBScript. A variable name:

- ◆ Must begin with an alphabetic character.

- ◆ Cannot have any special characters
- ◆ Must not exceed 255 characters.
- ◆ Must be unique in the scope in which it is declared.

Declaration of a Variable

The variables can be declared in VBScript using the following syntax.

```
DIM variablename
```

E.g.: dim x, empl, and sal

In VBScript we can use the variables implicitly without declaring it. But we may make some spelling mistakes, which may not result in an error, instead, it will generate a logically incorrect answer. To avoid this situation, we can include a statement option explicit in the first line of our program. This will make you explicitly to declare a variable that you use in any procedure inside the script.

```
<script language = vbscript> option explicit
```

```
dim x
```

```
.....
```

```
</script>
```

Scope and Lifetime of a Variable

A variable's scope is determined by where you declare it. When a variable is declared within a procedure, only code within that procedure can access or change the value of that variable. It has local scope and is called a procedure level variable. If a variable is declared outside a procedure, it is made available to all the procedures in the script. This is a script level variable, and it has script-level scope.

How long a variable exists is its lifetime. The lifetime of a script-level variable extends from the time it's declared until the time the script has finished running. At procedure level, a variable exists only as long as you are in the procedure. When the procedure exits, the variable is destroyed.

Declaration of the variables in the script explicitly can be done using the *Dim* statement, the *public* statement, and the *private* statement.

When a variable is declared as public that variable can be accessed by all the procedures in the script level. Private variables can be accessed only within the script level. When a variable is declared as Dim it can be accessed either at the procedure level or at the script level.

20.5 Short Summary

- ◆ VBScript acts as both a client-side and server-side programming language

- ◆ Any scripting language can be enclosed within a script tag in an ordinary HTML file
- ◆ If the value of a specified memory location changes at run time then it is defined to be a variable
- ◆ The variables can be declared in VBScript using the syntax - DIM variablename
- ◆ The scope of the variable may be either at the local level or at the script level
- ◆ A variable can be declared as either public or private

20.6 Brain Storm

1. What do you mean by client-side and server-side programming languages
2. How do you introduce scripts into a HTML document
3. Rules while defining a variable name
4. Discuss the scope of a variable and its declaration

Lab Unit (2 Real Time Hours)

Let us now refresh our HTML knowledge creating the web page as follows:

Design at least three HTML pages BookTitle.html, Chapters.html and Contents.HTML. Depending on number of chapters and chapters' content you have to design the rest of html pages. All pages should be saved in same folder for sake of easy reference and future enhancement.

Design a default html page with name default.html. It should be as shown in the figure by using frame.

Default.html

ABC Online Publication HTML BookTitle.htm	
Chapter 1 Introduction to HTML Chapter 2 Web Publishing Chapter 3 Links and Addressing Chapters.html	<i>Basic HTML Concepts</i> <i>Overview of HTML</i> Logical and Physical Elements What HTML is Not Contents.html

Use Design the Web Pages according to your requirement. Use all the possible TAG options.

Constants and Arrays

Objective

This lecture provides an insight into the following:

- ❖ Defining constants
- ❖ Variant data type of VBScript
- ❖ Representing VBScript data type
- ❖ Single and Multidimensional arrays
- ❖ Static and Dynamic arrays
- ❖ Adding comments to your script

Lecture - 21

- 21.1 *Snap Shot*
- 21.2 Constants
- 21.3 Data types in VBScript
- 21.4 Arrays
- 21.5 Commenting
- 21.6 Short Summary
- 21.7 Brain Storm

21.1 Snap Shot

Following our study on variables and their representation, we now study about user-defined constants. The representation of data is a very important feature of any language. VBScript represents data only as type Variant. We shall deal with the various subtypes of the variant type. We shall also study the array representations in VBScript. Here arrays may either be single arrays or multidimensional.

21.2 Constants

A constant is a named item that retains a constant value throughout the execution of a program. Constants can be used anywhere in your code in place of actual values. A constant can be a string or numeric literal, another constant, or any combination that includes arithmetic or logical operators except Is and exponentiation.

A constant is a meaningful name that takes the place of a number or string and never changes.

User defined constants can be declared using a keyword Const

For example:

```
Const A = "MyString"
```

21.3 Datatypes in VBScript

VBScript has only one data type called a Variant. A Variant is a special kind of data type that can contain different kinds of information, depending on how it's used. Because Variant is the only data type in VBScript, it's also the data type returned by all functions in VBScript.

At its simplest, a Variant can contain either numeric or string information. A Variant behaves as a number when you use it in a numeric context and as a string when you use it in a string context. That is, if you're working with data that looks like numbers, VBScript assumes that it is numbers and does the thing that is most appropriate for numbers. Similarly, if you're working with data that can only be string data, VBScript treats it as string data. The numbers can behave as strings by enclosing them in quotation marks

(" ").

Other than the simple numeric and string data, a variant data type can have other subtypes and the following table shows the subtypes of data that a Variant can contain.

Subtype	Description
Empty	Variant is uninitialized. Value is 0 for numeric variables or a zero-length string ("") for string variables.
Null	Variant intentionally contains no valid data.
Boolean	Contains either true or false.
Byte	Contains integer in the range 0 to 255.
Integer	Contains integer in the range -32,768 to 32,767.
Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807.
Long	Contains integer in the range -2,147,483,648 to 2,147,483,647.
Single	Contains a single-precision, floating-point number in the range -3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values.
Double	Contains a double-precision, floating-point number in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values.
Date (Time)	Contains a number that represents a date between January 1, 100 to December 31, 9999.
String	Contains a variable-length string that can be up to approximately 2 billion characters in length.
Object	Contains an object.
Error	Contains an error number.

To know what type a variable has been assigned, use TYPENAME (varname). This returns the data type the variable is currently assigned with.

21.4 Arrays:

Arrays are a group of similar elements stored in contiguous memory location.

Arrays are broadly classified as

1. Single dimensional array.
2. Multi dimensional array.

Single Dimensional Array

Single dimensional array can be further classified as

1. Static array
2. Dynamic array

Static Array:

Declaration of Static array can be given as

Dim variablename (value)

E.g.:

```
dim x(10)
```

The above declaration allocates a memory location under a variable name x and it provides space for entering 10 elements under the same variable. All the elements must be of same data type and it can be referred using the subscript of the variable. Arrays are Zero based means if we declare x(10) then 11 elements can be stored in that and all the values can be retrieved as x(0),x(1)x(10).

E.g.:

```
<script language = vbscript>
```

```
option explicit
```

```
dim x(3), i
```

```
for i = 0 to 2
```

```
x(i) = i
```

```
msgbox x(i)
```

```
next
```

```
</script>
```

Output :

X(0) = 0

X(1) = 1

X(2) = 2

X(3) = NULL

+6

Suppose it is necessary to erase the contents of the array i.e. when we consider the above example all the elements in the array must be made as NULL then we can use a Keyword ERASE

Syntax :

ERASE Arrayname

E.g.:

```

<script language = vbscript>
option explicit
dim x(1), i
for i = 0 to 1
x(i) = i
msgbox x(i)
next
Erase x
for i = 0 to 1
x(i) = i
msgbox x(i)
next
</script>

```

Output

X(0) = 0

X(1) = 1

X(0) = NULL

X(1) = NULL

Dynamic array

When a static array is declared, at the time of declaration itself the space to be allocated by the memory is specified. Even if the allocated memory is not used, it is left as null and memory space is wasted. Hence when allocation of an array variable can be done dynamically using Dynamic declaration where memory location is assigned at runtime.

We should follow three steps to declare a Dynamic array.

Dim x()

N = value

Redim x(N)

E.g.:

Dim x(), i, n

N = cint(Inputbox("Enter the size of the array"))

```
Redim x(n)
For i = 1 to n
X(i) = i
Msgbox x(i)
Next
```

Here in a Dynamic Array if we use ERASE x, it destroys the entire memory location itself where as in the case of Static array it initializes the array location.

In the following example,

```
<script language = vbscript>
dim x(), i
redim x(2)
for i = 0 to 2
x(i) = i
msgbox x(i)
next
redim x(3)
for i = 0 to 3
msgbox "after redim x(3) " & x(i)
next
</script>
```

Output

X(0)=0,x(1) = 1, x(2) = 2, x(0) = NULL, x(1) = NULL, x(2) = NULL, x(3) = NULL

Suppose we want the values in the array to be preserved then we can Use the Key word PRESERVE as shown in the example which will retain the values that were already initialized.

E.g.:

```
<script language = vbscript>
dim x(), i
redim x(2)
for i = 0 to 2
x(i) = i
msgbox x(i)
next
```

```

redim Preserve x(3)
for i = 0 to 3
msgbox "after redim x(3) " & x(i)
next
</script>

```

Output

X(0)=0,x(1) = 1, x(2) = 2, x(0) = 0, x(1) =1, x(2) = 2, x(3) = NULL

Incase of dynamic array we can use two functions LBOUND and UBOUND.

E.g.:

```

<script language = vbscript>
dim x(), i, n
n = cint(Inputbox("enter the value for size of array"))
redim x(n)
for i = lbound(x) TO ubound(x)
x(i) = i
next
</script>

```

Here LBOUND(x) is 0 and UBOUND (x) = n

Multidimensional array

Multidimensional Arrays are capable of storing multiple indexed values under the same variable name. Two-dimensional arrays are most commonly in use. Two-dimensional arrays store values in matrix form in the memory location.

Declaration of Two dimensional array:

```
Dim variablename (value, value)
```

E.g.:

```

<script language = vbscript>
Dim x(3,2), i, j
For i = 0 to 3
For j= 0 to 2
X(i,j) = i
Msgbox x(i,j)
Next
Next

```

</script>

Output

X(0,0) = 0,x(0,1) = 0, x(0,2) = 0

X(1,0) = 1,x(1,1) = 1, x(1,2) = 1

X(2,0) = 2,x(2,1) = 0, x(2,2) = 2

X(3,0) = 3,x(3,1) = 3, x(3,2) = 3

In case of two dimensional arrays if we use Ubound then ,

Dim x(3,5)

Ubound(x) = 3

Ubound(x,2)=5

21.5 Commenting

Comments are a vital aspect in any programming language. It aids clarity and readability to the program. VBScript can also be commented as follows:

```
<!-- I am an HTML Comment -->
<%
  REM Example for VBScript Comment.
  ` Another Example for VBScript Comment.
%>
```

21.6 Short Summary

- ◆ A constant is a meaningful name that takes the place of a number or string and never changes.
- ◆ User defined constants can be declared using a keyword Const
- ◆ VBScript has only one data type called a Variant
- ◆ Arrays are a group of similar elements stored in contiguous memory location
- ◆ Single dimensional array can be classified as Static and Dynamic Arrays
- ◆ Comments aids clarity and readability to the program

21.7 Brain Storm

1. How do you represent constants
2. Explain the various subtypes of the data type variant in VBScript
3. What are static and dynamic arrays
4. Explain the advantages and disadvantages in using single and multi dimensional arrays
5. What is the significance of adding Comments to your script.

Lecture 22

Date and Time Functions

Objective

This lecture provides an insight into the following:

- ❖ Adding Date and Time Functions to Scripts
- ❖ The DateAdd Function
- ❖ Determining the System Time

Lecture - 22

22.1 *Snap Shot*

22.2 Adding Date and Time Functions to Scripts

22.3 Short Summary

22.4 Brain Storm

22.1 Snap Snot

It is often necessary to include in our web pages, the current date and time. We might also require obtaining information about browsers who have visited our page in terms of the date they visited. VBScript includes a number of functions that allow you to retrieve and format dates and time. Let us now look in detail about the various date and time functions

22.2 Adding Date and Time Functions to Scripts

1. Date

Return Type : Date

Description

It returns the current system date and does not take any parameters.

2. DateAdd

Return Type : Date

Description

Returns a date to which a specified time interval has been added.

Syntax

`DateAdd(interval, number, date)`

The DateAdd function syntax has these parts:

Part	Description
Interval	Required. String expression that is the interval you want to add. See Settings section for values.
Number	Required. Numeric expression that is the number of interval you want to add. The numeric expression can either be positive, for dates in the future, or negative, for dates in the past.
Date	Required. Variant or literal representing the date to which interval is added.

Settings

The *interval* argument can have the following values:

Setting	Description
<i>Yyyy</i>	Year
<i>Q</i>	Quarter
<i>M</i>	Month
<i>Y</i>	Day of year
<i>D</i>	Day
<i>W</i>	Weekday
<i>Ww</i>	Week of year
<i>H</i>	Hour
<i>M</i>	Minute
<i>S</i>	Second

Remarks

You can use the DateAdd function to add or subtract a specified time interval from a date. For example, you can use DateAdd to calculate a date 30 days from today or a time 45 minutes from now. To add days to *date*, you can use Day of Year ("y"), Day ("d"), or Weekday ("w").

The DateAdd function won't return an invalid date. The following example adds one month to January 31:

```
NewDate = Dateadd("m",1,"31-Jan-95")
```

In this case, DateAdd returns 28-Feb-95, not 31-Feb-95. If *date* is 31-Jan-96, it returns 29-Feb-96 because 1996 is a leap year.

If the calculated date would precede the year 100, an error occurs.

3. DateDiff

Description

Returns the number of intervals between two dates.

Syntax

```
DateDiff(interval, date1, date2)
```

The interval specified for the previous function holds good here.

4. Day

Description

Returns a whole number between 1 and 31, inclusive, representing the day of the month.

Syntax

```
Day(date)
```

5. Date Serial

Description

Returns a Variant of subtype Date for a specified year, month, and day.

Syntax

DateSerial(*year, month, day*)

The DateSerial function syntax has these arguments:

Part	Description
<i>Year</i>	Number between 100 and 9999, inclusive, or Numeric Expression.
<i>Month</i>	Any numeric expression.
<i>Day</i>	Any numeric expression.

E.g.: DateSerial(99,2,26) = 26 - feb - 99

DateSerial(99,2,32) = 4 - mar - 99

6. Time

Returns the current system time

7. Now

Returns the current system time and date.

22.3 Short Summary

- ◆ Date returns the current system date and does not take any parameters.
- ◆ Day Returns a whole number between 1 and 31, inclusive, representing the day of the month
- ◆ Time returns the current system time
- ◆ Now returns the current system time and date

22.4 Brain Storm

1. Give the syntax for DateAdd
2. Describe the arguments used in the DateSerial function
3. What function do you use to manipulate a certain time interval from a given time

Lecture 23

Functions

Objective

This lecture provides an insight into the following:

- ❖ Numeric operators in VBScript
- ❖ System defined and User defined functions
- ❖ An insight into numeric, conversion and miscellaneous functions
- ❖ Getting user interaction through inputbox and msgbox functions

Lecture - 23

- 23.1 *Snap Shot*
- 23.2 Mathematical Operators
- 23.3 Functions
- 23.4 System Defined Functions
- 23.5 Inputbox and MSGbox Functions
- 23.6 Short Summary
- 23.7 Brain Storm

23.1 Snap Shot

Our web page often requires manipulating data, which may either be numeric or string. These are manipulated using mathematical operators. In the previous lecture we have seen the date/time functions. In addition to this there are a number of system defined functions namely numeric functions, conversion function and miscellaneous functions. Apart from these functions, we have Inputbox functions, which help us to get user input in our web pages and MSGbox function to give messages to the users.

23.2 Mathematical operators

VBScript includes all the mathematical operators that you would expect in a programming language. You can perform such operations as addition, subtraction, multiplication and division. The table lists the numeric operators available in VBScript and they are self-explanatory.

Arithmetic	
Description	Symbol
Exponentiation	^
Unary Negation	-
Multiplication	*
Division	/
Integer	\
Modulus Arithmetic	Mod
Addition	+
Subtraction	-
String Concatenation	&

Example:

```
<Script language = vbscript>
```

```
Dim x, y
```

```
X=10
```

```
Y=3
```

```
E = x ^ y ' find x to the power of y
```

```
I = x \ y ' performs integer division
```

```
M = x mod y ' returns remainder of the division performed by x/y
```

```
</script>
```

Output of the script

```
E=1000; I=3; M=1
```

23.3 Functions

A function can be defined as a method, which performs a action and returns a value with a specific return type whenever it is called for. Functions can be either

1. System defined functions
2. User defined functions

The subroutines and functions to be discussed in the next unit are nothing but the user defined functions.

23.4 System Defined Functions

System defined functions can be categorized broadly into five types.

1. String functions
2. Numeric functions
3. Date/Time functions
4. Conversion functions
5. Miscellaneous functions

Date/Time functions have been discussed earlier in the previous lecture and we shall see the details of String function later. Here, let us consider the other system defined functions.

Numeric Functions

The numeric functions are used to calculate and return some desired output value for some number.

ABS(number)

Return Type: Numeric

E.g.:

$$\text{ABS}(6) = 6$$

$$\text{ABS}(-6) = 6$$

SGN(number)

Return Type: Integer

E.g.:

$$\text{SGN}(34) = 1$$

$\text{SGN}(-34) = -1$

$\text{SGN}(0) = 0$

FIX(exp)

It returns a integer value towards a lesser value.

E.g.:

$\text{Fix}(6) = 6$

$\text{Fix}(-6) = -6$

$\text{Fix}(6.7) = 6$

$\text{Fix}(-6.7) = -7$

Other built in functions like COS, TAN, SIN can also be used.

Conversion functions

These functions are used to convert to specific datatypes that are passed as a parameter.

1. $\text{CINT}(\text{exp})$ - Converts the expression datatype to integer.
2. $\text{CLNG}(\text{exp})$ - Converts the expression datatype to long.
3. $\text{CSNG}(\text{exp})$ - Converts the expression datatype to single.
4. $\text{CDBL}(\text{exp})$ - Converts the expression datatype to double.
5. $\text{CSTR}(\text{exp})$ - Converts the expression datatype to string.
6. $\text{CDATE}(\text{exp})$ - Converts the expression datatype to Date.
7. $\text{CBOOL}(\text{exp})$ - Converts the expression datatype to Boolean.
8. $\text{CCUR}(\text{exp})$ - Converts the expression datatype to money.

Miscellaneous functions

$\text{ISNUMERIC}(\text{exp})$

Returns a boolean value to say whether the expression is returning a numeric value.

$\text{ISEMPTY}(\text{exp})$

Returns a boolean value if Variant datatype in the expression is empty.

$\text{ISNULL}(\text{exp})$

Returns a boolean value if the expression contains NULL value.

TypeName(Exp)

It gives the datatype depending on the value stored in the expression.

E.G.:

```
<script language = vbscript>
dim x,y
x=10
y="hello"
msgbox typename(x)
msgbox typename(y)
</script>
```

Output:

The output of the first message box will be Integer and the output of the second message box will be String. Though the outer classification of x and y is taken as variant, according to data stored in the variables the sub class data type is automatically assigned.

23.5 Inputbox and MSGbox Functions

In addition to the above mentioned System defined functions, VBScript also has two more system-defined functions, which enables the user to interact with the application.

INPUTBOX assigns a value to a variable in the application at run time by prompting a dialog box for entering a value. In contrast the application prompts a message or output to the user when a MSGBOX is used.

Msgbox

Syntax

Msgbox (prompt [, buttons] [, title])

The Msgbox syntax has these arguments:

Argument	Description
Prompt	String expression displayed as the message in the dialog box. The maximum length of prompt is approximately 1024 characters, depending on the width of the characters used. If prompt consists of more than one line, you can

	separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or carriage return–linefeed character combination (Chr(13) & Chr(10)) between each line.
Buttons	Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. See Settings section for values. If omitted, the default value for buttons is 0.
Title	String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.

Settings

The buttons argument settings are:

Constant	Value	Description
VbOKOnly	0	Display OK button only.
VbOKCancel	1	Display OK and Cancel buttons.
VbAbortRetryIgnore	2	Display Abort, Retry, and Ignore buttons.
VbYesNoCancel	3	Display Yes, No, and Cancel buttons.
VbYesNo	4	Display Yes and No buttons.
VbRetryCancel	5	Display Retry and Cancel buttons.
VbCritical	16	Display Critical Message icon.
VbQuestion	32	Display Warning Query icon.
VbExclamation	48	Display Warning Message icon.
VbInformation	64	Display Information Message icon.
VbDefaultButton1	0	First button is default.
VbDefaultButton2	256	Second button is default.
VbDefaultButton3	512	Third button is default.
VbDefaultButton4	768	Fourth button is default.
VbApplicationModal	0	Application modal; the user must respond to the message box before continuing work in the current application.
VbSystemModal	4096	System modal; all applications are suspended until the user responds to the message box.

The first group of values (0–5) describes the number and type of buttons displayed in the dialog box; the second group (16, 32, 48, 64) describes the icon style; the third group (0, 256, 512, 768) determines which button is the default; and the fourth group (0, 4096) determines the modality of the message box. When adding numbers to create a final value for the argument buttons, use only one number from each group.

Return Values

The MsgBox has the following return values:

Constant	Value	Button
VbOK	1	OK
VbCancel	2	Cancel
VbAbort	3	Abort
VbRetry	4	Retry
VbIgnore	5	Ignore
VbYes	6	Yes
VbNo	7	No

Remarks

If the dialog box displays a Cancel button, pressing the ESC key has the same effect as clicking Cancel.

Inputbox

Syntax

Inputbox (prompt [, title][, default][, xpos][, ypos])

The Inputbox syntax has these arguments:

Argument	Description
Prompt	String expression displayed as the message in the dialog box. The maximum length of prompt is approximately 1024 characters, depending on the width of the characters used. If prompt consists of more than one line, you can separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or carriage return-linefeed character combination (Chr(13) & Chr(10)) between each line.
Title	String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.
Default	String expression displayed in the text box as the default response if no other input is provided. If you omit default, the text box is displayed empty.
Xpos	Numeric expression that specifies, in twips, the horizontal distance of the left edge of the dialog box from the left edge of the screen. If xpos is omitted, the dialog box is horizontally centered.
Ypos	Numeric expression that specifies, in twips, the vertical distance of the upper edge of the dialog box from the top of the screen. If ypos is omitted, the dialog box is vertically positioned approximately one-third of the way down the screen.

Remarks

If the user clicks OK or presses ENTER, the Inputbox returns whatever is in the text box. If the user clicks Cancel, it returns a zero-length string ("").

E.g.:

```
Dim x
```

```
X = Inputbox("enter the value for x", "demo for Inputbox")
```

```
Msgbox "The value of x is " & x
```

23.4 Short Summary

- ◆ Mathematical or numeric operators are used to perform such operations as addition, subtraction, multiplication and division
- ◆ A function can be defined as a method, which performs an action and returns a value with a specific return type whenever it is called for
- ◆ The numeric functions are used to calculate and return some desired output value for some number
- ◆ Conversion functions are used to convert to specific datatypes that are passed as a parameter
- ◆ INPUTBOX assigns a value to a variable in the application at run time by prompting a dialog box for entering a value
- ◆ The application prompts a message or output to the user when a MSGBOX is used

23.5 Brain Storms

1. Give examples to explain the different types of operators
2. What do you mean by user defined and system defined function
3. Explain the significance of Conversion functions
4. Explain in detail the working of MSGbox and INPUTbox functions

Lecture 24

Conditional Statements

Objective

This lecture provides an insight into the following:

- ❖ Need for conditional statements
- ❖ Use of and syntax of IF statement
- ❖ Using Select Case statements in your script

Lecture - 24

24.1 *Snap Shot*

24.2 Using Conditional Statements

24.3 Short Summary

24.4 Brain Storm

24.1 Snap Shot

It is often necessary to control the flow of your script. This can be done with conditional and looping statements. Using conditional statements, you can write VBScript code that makes decisions and repeats action. Let us consider the Conditional Statements in detail

24.2 Using Conditional Statements

IF Statement:

Syntax:

```
If <condition> then
.....
.....
statements
.....
[elseif condition then]
.....
statements
.....
[else]
.....
statements
.....
end if
```

Example:

```
<script language = vbscript>
option explicit
dim m1,m2,m3,s,avg
m1=cint (Inputbox ("enter the first mark"))
m2=cint (Inputbox ("enter the second mark"))
m3=cint (Inputbox ("enter the Third mark"))
s=m1+m2+m3
msgbox "the total marks is " & s
avg = s/3.0
if avg >= 80 then
msgbox "Distinction"
elseif avg >= 60 and avg < 80 then
msgbox "First Class"
elseif avg >=40 and avg < 60 then
msgbox " Second Class"
else
msgbox "Fail Class"
```



```
end if
</script>
>
```

Select Case Statement

Select case (Expression/true/false)

```
Case value/condition
.....
.....
```

```
Case value/condition
.....
```

```
Case Else
.....
.....
```

```
End select
```

Ex1:

```
<script language = vbscript>
select case true
case a>b
msgbox a & " is greater"
case b>a
msgbox b & " is greater"
case else
msgbox "Both are equal"
end select
</script>
```

Ex2:

```
<script language = vbscript>
a=Inputbox("enter a value")
select case a
case 1
msgbox "the value entered is 1"
case 0
msgbox " the value entered is 0"
case else
msgbox "entered a value other than 0 / 1"
end if
</script>
```

24.3 Short Summary

- ◆ Conditional statements are used to alter the sequence of program flow
- ◆ IF statement is used when there is a need to check for a condition before proceeding with alternative sequence of operations
- ◆ Select Case Statement is used when there are a number of alternative paths to choose from for the program flow

24.4 Brain Storm

1. Compare and contrast IF statement with Select Case Statement
2. Give the syntax for IF statement
3. Explain with an example the Select Case statement

String Functions

Objective

This lecture provides an insight into the following:

- ❖ The string functions
- ❖ Case conversion of strings
- ❖ Concatenating strings
- ❖ Reversing a string
- ❖ Comparing two strings

Lecture - 25

- 25.1 *Snap Shot*
- 25.2 The String Functions
- 25.3 Short Summary
- 25.4 Brain Storm

Lab Unit

25.1 Snap Shot

In the previous lectures we have seen various system-defined functions. One among them is the string function, which we shall deal with in this lecture. VBScript includes a rich set of functions for working with strings. You will find these functions very useful when you need to manipulate data pulled from HTML forms or databases. You can use these functions to concatenate strings, extract strings, search strings and compare strings.

25.2 String Functions

1. LCASE(string)

Return Type: String

E.g.:

```
x= "RADIANT"  
X=LCASE(x)
```

Output:

"radiant"

2. UCASE(string)

Return Type: String

E.g.:

```
x= "RADIANT"  
X=UCASE(x)
```

Output :

"RADIANT"

3. LTRIM(string)

Return Type: String

E.g.:

```
x= "  RADIANT  "  
X=LTRIM(x)
```

Output :

"RADIANT "

4. RTRIM(string)

Return Type: String

E.g.:

```
x= "  RADIANT  "  
X=RTRIM(x)
```

Output:

" RADIANT "

5. TRIM(String)

Return Type: String

E.g.:

```
x= "  RADIANT  "  
X=TRIM(x)
```

Output :

```
"RADIANT "
```

TRIM () function must be used in all the places where we use the textboxes as the return type of the input type "text" is a string.

6. LEN(string)

Return Type: Integer

E.g.:

```
x= "Radiant"  
X = LEN(x)
```

Output :

```
7
```

7. LEFT(String, length)

Return Type: String

It returns number of characters specified in the length from the left.

E.g.:

```
x= "RADIANT "  
X=LEFT(x,3)
```

Output :

```
"RAD"
```

8. RIGHT(String, length)

Return Type: String

It returns number of characters specified in the length from the right.

E.g.:

```
x= "RADIANT "  
X=RIGHT(x,3)
```

Output :

```
"A NT"
```

9. MID(String, start, length)

Return Type: String

It returns characters specified in the length from the start position mentioned.

E.g.:

```
x= "RADIANT "  
X=MID(x,3,3)
```

Output :

```
"DIA"
```

Here the length of the string defined is 7. "R" takes the position 1 through "T" taking the position 7. When the start position is 3 it counts from the left and it happens to be "D" in this case. From that position, it starts to assign, till the length that has been specified as the third parameter. This function can be used as a substitute for LEFT and RIGHT functions.

10. STRREVERSE(string)

Return Type: String

It returns the reverse of a string specified.

E.g.:

X= "RADIANT"

X = STRREVERSE(x)

Output :

"TNAIDAR"

11. STRCOMP(string1, string2, options)

Return Type: Integer

It returns a integer value according to the length of the string1 and string2.

If string1 > string2 it returns 1

If string1 < string2 it returns -1

If string1 = string2 it returns 0

It checks alphabet wise and it also checks for the case.

"A" is different from "a" and interestingly "A" is considered less than "a" as the ASCII value of A is 65 and correspondingly for "a" is 97.

This function can hold a optional third parameter [options].

Either VBTEXTCOMPARE or VBBINARYCOMPARE can be used for comparison. By default it considers VBBINARYCOMPARE. That is, it checks for the case of the string and uppercase character is viewed different from lowercase character. But when we need not consider for the case that we are typing in, We can opt for VBTEXTCOMPARE, which checks only for the text discarding the case of the character.

12. INSTR (start, string1, string2, options)

Return Type: Integer

It checks the occurrence of the string2 in string1 from the start position mentioned and it returns the exact location of the next string2 present in the string1.

E.g.:

x = INSTR(1,"radiant", "a")

Output :

2

x = INSTR(3,"radiant", "a")

Output :

5

(The next occurrence of the character "a" from the position 3 is at position 5)

25.3 Short Summary

- ◆ Case conversion of a string is done using LCASE(string) and UCASE(string)
- ◆ Trimming a given string is done using LTRIM(string), RTRIM(string), TRIM(string)
- ◆ Determining the length of the string is done with the LEN(string) command
- ◆ We can reverse a given string using STRREVERSE(string)
- ◆ Comparison of two strings is done by STRCOMP(string1, string2, options)

25.4 Brain Storm

1. Write a script to convert a string in upper case to lower case and vice versa
2. Write a script to determine the length of a string. Accept the string from the user
3. How do you check the occurrence of one string in another

Lab Unit (2 Real Time Hours)

1. Design a Calculator to perform the basic Arithmetic Operations such as addition, Multiplication, Subtraction, Division and try to find log, sin, cos for the given values using VBScript.
2. Write a program to change your browser's window color randomly on a button press using VBScript.

Lecture 26

User – defined functions

Objective

This lecture provides an insight into the following:

- ❖ Need for subroutines
- ❖ Syntax definition for subroutines
- ❖ Parameters passing into a subroutine
- ❖ Using functions in your script

Lecture - 26

- 26.1 *Snap Shot*
- 26.2 Creating Subroutines
- 26.3 Creating Functions
- 26.4 Short Summary
- 26.5 Brain Storm

26.1 Snap Shot

Along our way we have seen that there are two types of functions that may be used in a VBScript namely system defined functions and user defined functions. String functions, date/time functions, conversion functions, miscellaneous functions and numeric functions are all system defined and we have already dealt in detail about each of these. In this lecture we focus our study on user-defined functions namely Subroutines and Functions

26.2 Creating Subroutines

In case you need to execute the same group of statements in more than one place in a script, we should consider using a Subroutine. A Subroutine can contain any collection of VBScript statements. The same subroutine may be called any number of times.

Syntax

```
Sub subroutinename([optional parameters])
```

```
.....
```

```
.....
```

```
[exit sub]
```

```
.....
```

```
.....
```

```
End sub
```

After writing a procedure it can be called in 2 ways. It can be called as

```
CALL subroutinename ([option parameter])
```

OR

```
Subroutinename ([option parameter])
```

Example:

```
<script language = vbscript>
option explicit
sub pname()
dim s,m1,m2,m3
s=m1+m2+m3
msgbox "The total marks for 3 subjects is " & s
end sub
call pname()
pname()
</script>
```

The above code is executed at the time of loading itself and the procedure is executed twice as it has been called two times.

Passing parameters to a subroutine

```

<script language = vbscript>
option explicit
sub pname(x)
dim s
s = x ^ 2
msgbox "The square of the number x is " & s
end sub
call pname(3)
pname(7)
</script>

```

The parameters are passed along while calling the procedure.

26.3 Creating Functions

Syntax

```
Function functionname ([parameters])
```

```
.....
```

```
.....
```

```
functionname=expression
```

```
.....
```

```
.....
```

```
exit function
```

```
.....
```

```
.....
```

```
End function
```

A function can be called by specifying functionname with parameters. It returns a value through the function name itself.

Example

The same code as given for subroutine can be executed through a function also and it can be made to return a value.

```

<script language = vbscript>
dim x,y
function fname(x)
fname = x ^ 2
end function

```

```
x=fname(3)
y=fname(7)
msgbox "The Function returns the values of x and y as " & x & y
</script>
```

26.4 Short Summary

- ◆ Subroutines and Functions are user defined functions
- ◆ A Subroutine can contain any collection of VBScript statements that may be called any number of times
- ◆ The parameters are passed along while calling the procedure or subroutine
- ◆ A function is similar to a subroutine and it returns a value

26.5 Brain Storm

1. Explain Subroutines with an example
2. How are functions useful in our Web page
3. Compare Subroutines and Functions

Lecture 27

Operators

Objective

This lecture provides an insight into the following:

- ❖ Need for logical connectives and operators
- ❖ Describing comparison operators
- ❖ A study on logical operators
- ❖ Truth table analysis for logical operators

Lecture - 27

- 28.1 Snap Shot
- 28.2 Using logical connectives and operators
- 28.3 Truth Table for logical operators
- 28.4 Short Summary
- 28.5 Brain Storm

27.1 Snap Shot

Creating web pages using VBScript is often used to create an interactive environment between the page and the user. Certain inputs may have to be obtained from the user, which are manipulated for one reason or the other. These may include comparing or combining values whether it is of numeric, string or Boolean type. These are taken care of by a set of logical connectives and operators, which we shall deal with in this lecture.

27.2 Using Logical Connectives and Operators

VBScript has a wide range of logical operators and connectives to perform necessary operation on the script. Let us consider the following table, which is self-explanatory.

Comparison	
Description	Symbol
Equality	=
Inequality	<>
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=
Object Equivalence	Is

Logical	
Description	Symbol
Logical Negation	Not
Logical Conjunction	And
Logical Disjunction	Or
Logical Exclusion	Xor
Logical Equivalence	Eqv
Logical Implication	Imp

27.3 Truth Table for Logical Operators

Logical Operators follows a set of rules, which can be used with comparison statements.

The rule followed by the operator is defined through truth table.

Truth Table for And Operator

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table for Or Operator

		Y
		0
		1
		1
		1

Truth Table for Not Operator

Where A, B refers input
 Y refers output
 0 considered False
 1 considered True

27.4 Short Summary

- ◆ VBScript has a wide range of logical operators and connectives to perform necessary operation on the script
- ◆ Logical Operators follows a set of rules, which can be used with comparison statements
- ◆ Truth tables may be used to represent the rules upon which logical operations are based

27.5 Brain Storm

1. Write a script to explain the use of logical operators
2. With an example give the significance of the comparison operators

Lecture 28

Iteration

Objective

This lecture provides an insight into the following:

- ❖ Repetition of code using loops
- ❖ The FOR loop
- ❖ Using the DO loop
- ❖ Looping using the WHILE loop

Lecture - 28

- 29.1 Snap Shot
- 29.2 Using loops to repeat code
- 29.3 Short Summary
- 29.4 Brain Storm

Lab Unit

29.1 Snap Shot

It often becomes necessary when scripting to repeat a set of code for a specific number of times. This situation arises when certain set of operations is to be repeated as long as a specified condition is true or until a specified condition is satisfied. Situation may also arise when we want to execute a statement a definite number of times rather than, until a condition becomes true or false. These situations are dealt with in this lecture

29.2 Using Loops to Repeat Code

Looping Structures are necessary to perform a set of instructions repeatedly.

FOR Loop

Syntax

For Variablename = initial value TO final value [step increment/decrement]

```

.....
statements
.....
[exit for]
.....
next

```

E.g.:

```

<script language = vbscript>
dim i
for i = 1 to 5
msgbox i
next
s1=cstr(Inputbox("enter the string"))
for i = 1 to len(s1)
msgbox mid(s1,i,1)
next
</script>

```

DO loop

Syntax:

Do while/until <condition>

```

.....
Statements

```

```
.....  
exit do  
.....  
loop
```

E.g.:

```
<script language = vbscript>  
dim a  
a = 10  
do while not a <=0  
msgbox a  
a = a - 1  
loop  
</script>
```

Do

Syntax:

```
Do  
.....  
statements  
.....  
statements  
.....  
exit do  
.....  
statements  
loop while/until <condition>
```

Try the same example with this looping construct and note the output.

While loop

Syntax:

```
While Condition  
.....  
statements  
.....  
wend
```

The example discussed above can be implemented using a while loop.

29.3 Short Summary

- ◆ Looping Structures are necessary to perform a set of instructions repeatedly.
- ◆ The for loop is used when we want to execute a statement a definite number of times rather than, until a condition becomes true or false
- ◆ Do loops are needed to perform iteration over a set of statements until a specified condition is satisfied
- ◆ The WHILE loop repeats a set of statements as long as a condition is satisfied

29.4 Brain Storm

1. Discuss the criteria upon which the selection of looping structure is done
2. Compare the DO loops with condition at the beginning of the loop and at the end of the loop
3. Write scripts to explain the use of the various looping structures

Lab Unit (2 Real Time Hours)

- a) Basic Salary for an employee is inputted through the keyboard. HRA is 20% of the Employee's basic salary and DA is 15% of Employee's Basic Salary. Write a program to calculate the employees Gross Salary and print the details.
- b) Find the Date difference between the current Date and Inputted Date.

Events and Forms

Objective

This lecture provides an insight into the following:

- ❖ Creating a simple page using VBScript
- ❖ Reaction of the Internet Explorer when <SCRIPT> tag is encountered
- ❖ Event Procedures
- ❖ Categories of Event Procedures
- ❖ Need for Forms in a VBScript
- ❖ How to represent forms in your web page
- ❖ InnerText and InnerHTML properties
- ❖ Visibility property

Lecture - 29

- 29.1 Snap Shot
- 29.2 A Simple Page
- 29.3 Events
- 29.4 Event Usage with Examples
- 29.5 VBScript and Forms
- 29.6 Short Summary
- 29.7 Brain Storm

Lab Unit

29.1 Snap Shot

We have upto now, on our study of VBScript dealt with the variables, array representation of variables, constants, operators and system defined and user-defined functions. We have also dealt with the various conditional and looping structures which all form an integral part of a Script. Using all what we have studied up to now on VBScript, we shall try to design a simple web page.

29.2 A Simple Page

With Microsoft Internet Explorer, you can view the page produced by the following HTML code. If you click the button on the page, you see VBScript in action. This is given in figure 29.1

```
<HTML>
<HEAD>
<TITLE> A Simple Page</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
    Sub Button1_OnClick
        MsgBox "Have a Good Day"
    End Sub
-->
</SCRIPT>
</HEAD>
<BODY>
<H2>A Simple Page</H2><hr>
<FORM><INPUT NAME="Button1" TYPE="Button" VALUE="Click Here"></FORM>
</BODY>
</HTML>
```

When Internet Explorer reads the page, it finds the <SCRIPT> tags, recognizes there is a piece of VBScript code, and saves the code. When you click the button, Internet Explorer makes a connection between the button and the code, and runs the procedure.

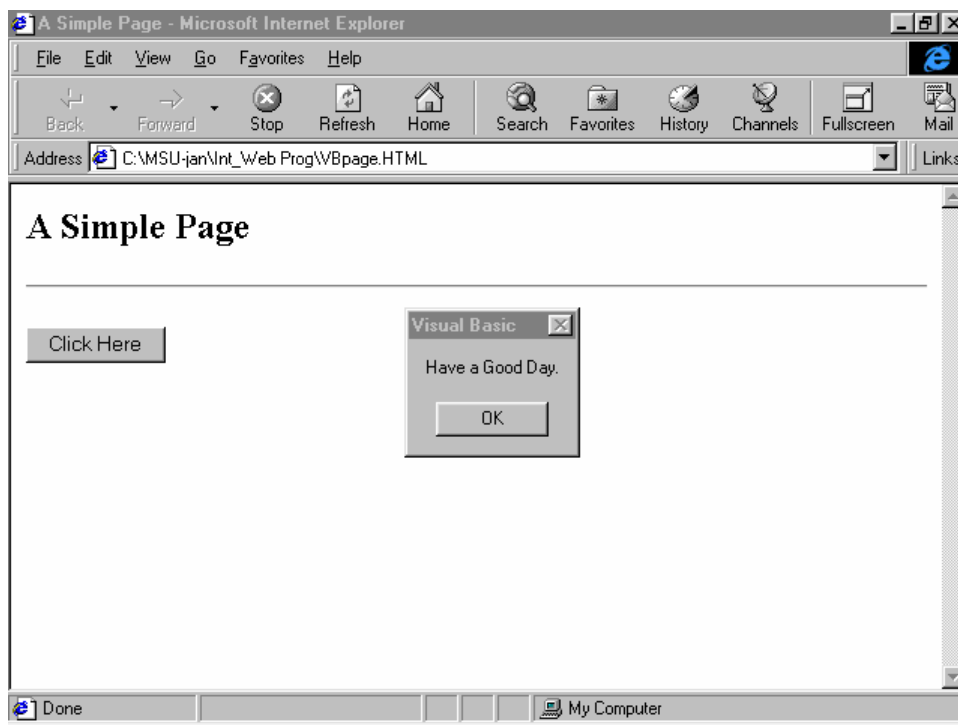


Figure 29.1 A Simple Page using VBScript

The Sub procedure in the <SCRIPT> is an event procedure. There are two parts to the procedure name: the name of the button, Button1 (from the NAME attribute in the <INPUT> tag), and an event name, OnClick. The two names are joined by an underscore(_). Any time the button is clicked, Internet Explorer looks for and runs the corresponding event procedure, Button1_OnClick.

Internet Explorer defines the events available for form controls in the Internet Explorer Scripting Object Model documentation

29.3 Events

Events are procedures, which will be executed whenever the user does some action.

We can categorize the events as

1. Mouse events
2. Keyboard events
3. Window events
4. Focus events

Mouse Events

We have different mouse events that are possible to be used in scripting.

1. Onmouseover

2. Onmouseout
3. Onmousemove
4. Onclick
5. Ondblclick
6. Onmousedown
7. Onmouseup

Keyboard Events

1. onkeydown
2. onkeypress
3. onkeyup

Onkeypress will not work with delete,ctrl, and other function keys(f1,...)

Onkeydown will work with all the keys in the keyboard.

Focus Events

1. onfocus - when an element is getting the focus in the web page.
2. Onblur - when an element loses the focus or when the cursor is moved out of the element.

Window Events

1. OnLoad - When the browser is opened and the window is loaded the event fires.
2. OnUnload - when the browser is closed the window gets unloaded or for every refresh button pressed or when a page is anchored the event fires.

29.4 Events Usage with Examples

The usage of events can be seen below through some examples.

```
<html>
<head>
<script language = vbscript>
sub p11_onmouseover
p11.style.color = "red"
p11.style.background="peachpuff"
```

```

end sub

sub p11_onmouseout
p11.style.color="green"
p11.style.background="white"
end sub
</script>
</head>
<body>
<p id = p11>This is a demo</p>
</body>
</html>

```

In the above example style is the method, which does some operation on the paragraph tag assigned with a id as p11. The procedure executes whenever the mouse is moved over the paragraph.

Suppose if the contents of the paragraph has to be changed on moving the mouse over the paragraph then we can use another property of paragraph.

```

<script language = vbscript>
sub p11_onmouseover
p11.innerText = "This line is changed dynamically"
p11.style.background="peachpuff"
end sub
sub p11_onmouseout
p11.innerHTML="<img src="" expbul1a.gif"">"
p11.style.color="green"
p11.style.background="white"
end sub
</script>

```

InnerText is the property used to change the content of the paragraph with some other text than existing currently. InnerHTML is the property used to change the content of the paragraph with some image that is located on the machine.

Here you can try the same example by substituting the "p" tag with "h1" through "h6". The style method can be used with any tag say anchor or table or image or div.

E.g.:

```

<html>
<head>
<script language = vbscript>
sub a11_onclick
d1.style.visibility = "visible"

```

```
end sub

sub d1_onmouseover
d1.style.visibility="hidden"
end sub

sub d1_onmouseout
d1.style.visibility="hidden"
end sub
</script>

</head>
<body>
<a id = a11 href = "demo.html">Click here to popout</a>
<div id = d1 style="visibility:hidden;position:absolute">
<table border=1 cellpadding=0 cellspacing = 0>
<tr><td>File
<tr><td>open
<tr><td>Save
<tr><td>Exit
</table>
</div>
</body>
</html>
```

Visibility is a property, which has two values hidden and visible and these must be given under quotes. The property "position : absolute" makes the contents under the div tag to pop out over the existing content after the div tag.

29.5 VBScript and Forms

Forms are constructed in the HTML documents by using the <FORM> element. This element contains several other elements, called controls that have a variety of methods for gathering information. When a form is completed and submitted, the information in its active controls is passed to a program that takes whatever action the form has been designed to perform.

Each element in the form has both a name and a value, thus the data that's passed for processing is in the form of name/value pairs. The processing of data is done by scripting languages such as VBScript or JavaScript or CGI program

All the tags in a HTML page can have a id and that id can be referred when using the events.

```
<tag id = "idname"> .....</tag>
<tag name = "name">.....</tag>
```

Tags, which can have name attribute, can be referred using the name and tags, which do not have the name attribute or tags having identical name can be differentiated through their IDNAME.

Syntax:

```
<script language = vbscript>
sub idname/Name_event()
.....
.....
end sub
</script>
```

29.6 Short Summary

- ◆ The Sub procedure in the <SCRIPT> is an event procedure
- ◆ Events are procedures, which will be executed whenever the user does some action
- ◆ Events may be classified as Mouse Events, Keyboard Events, Focus Events and Window Events
- ◆ Forms are constructed in the HTML documents by using the <FORM> element
- ◆ The processing of data is done by scripting languages such as VBScript or JavaScript or CGI program
- ◆ Tags are identified using IDNAME attributes
- ◆ InnerText is the property used to change the content of the paragraph with some other text than existing currently
- ◆ InnerHTML is the property used to change the content of the paragraph with some image that is located on the machine.
- ◆ Visibility is a property which has two values hidden and visible and these must be given under quotes

29.7 Brain Storm

1. Give the syntax for including a procedure to your VBScript
2. Explain with examples the action of the browser on the various Mouse Events and Keyboard Events

3. What do you mean by Focus Events and Window Events. Explain using examples
4. Discuss the need for forms in a Web page
5. What happens to the data input by the user
6. Discuss the various events properties using examples

Lab Unit (2 Real Time Hours)

Design a web page as given below

Text1 Should accept only numbers

The user should be allowed to enter only alphabets in text2

Text3 should accept only numbers and only 8 digits.

If the choice is female then the user should be allowed to enter values in Text4

If the allowances are applicable to the candidate it must be displayed in the respective labelboxes (HRA - 2500.00 DA - 3000.00)

If Ok button is clicked then a confirmation message should be displayed in the Label7

When CANCEL button is clicked, cancellation message should be displayed message box

On clicking the RESET button it should clear the Contents.

Use the various events like OnFocus, Onblur, etc.. for every control possible, (similar to events in VB Environment).

The form is titled "RADIANT SOFTWARE LIMITED EMPLOYEE INFORMATION". It contains the following elements:

- Empno :** A text box labeled "Text1".
- Name :** A text box labeled "Text2".
- Salary :** A text box labeled "Text3".
- Sex:** Radio buttons for "Male" and "Female". A text box labeled "Text 4" is positioned to the right of the radio buttons.
- ALLOWANCES:** A section containing two checkboxes: "Hra" and "Da".
 - Next to "Hra" is a text box labeled "Label5".
 - Next to "Da" is a text box labeled "Label6".
- Buttons:** Three buttons labeled "Ok", "Cancel", and "Reset" are stacked vertically.
- Label 7:** A large text box at the bottom of the form labeled "LABEL 7".

Lab Unit (2 Real Time Hours)

1. Design the form as below
2. Make Name & Password Textboxes and validate the input i.e.) Name textbox should get only alphabets and password should display only password character (*)
3. Write a Procedure to reverse the string and call it on clicking reverse string button click. Input should be through text1
4. Write a Procedure to display whether the number is prime or not. Check it with clicking of prime number button click
5. Write a procedure to get the input from the user in inputbox as salary and calculate the amount of loan eligible for that salary.
6. (for ex: salary=10,000 loan=18.5%(sal*12))
7. Display the loan amount in a message box.
8. Write a procedure to accept a word from the user and make the first letter alone as CAPS.

Try using all the possible inbuilt functions and develops this page for your practice.

RADIANT SOFTWARE LIMITED

Value :

Name :

Password :

Interactive Elements & Hiding Errors

Objective

This lecture provides an insight into the following:

- ❖ Using Interactive Elements in Scripting
- ❖ Radio and Checkbox buttons
- ❖ Examples for the above
- ❖ Methods to hide errors in your Web page

Lecture - 30

- 30.1 Snap Shot
- 30.2 Scripting using Interactive Elements
- 30.3 Scripting with Radio and Checkbox
- 30.4 Hiding Errors
- 30.5 Short Summary
- 30.6 Brain Storm

Lab unit

30.1 Snap Shot

In our final study on scripting using VBScript we shall now deal with scripting using interactive elements and scripting using radio and checkbox. The Web page that we create is to be viewed by millions of people and it is important on our part to make it free of errors. Of course human are prone to make mistakes. So here we discuss the way these errors can be hidden from the user.

30.2 Scripting Using Interactive Elements

We can use the values of the interactive elements in HTML using the following syntax

Tagname.value

The ASCII values of each and every key that is being typed can be trapped using a window event function “window.event.keycode”.

Example:

```
<html>
<body>
<script language = vbscript>
sub t1_onkeypress
msgbox window.event.keycode
end sub
</script>
<Input type = text name = t1>
</body>
</html>
```

The onkeypress event fires whenever we press any key by having the focus on the text box named t1. The ascii value of the key pressed is returned by the message box.

If the interactive elements are placed under a form then they must be referred inside the script through their form name.

E.g.:

```
<form name = f1>
<Input type = text name = t1>
</form>
```

The value of t1 must be retrieved as f1.t1.value.

30.3 Scripting With RADIO and CHECKBOX

E.g.1:

```

<html>
<head>
<script language = vbscript>
sub bt1_onclick
for i = 0 to 2
if fm1.r1(i).checked then
msgbox fm1.r1(i).value
end if
next
end sub
</script>
<head>
<body>
<form name = fm1>
<input type = radio name = r1 value = bread>Bread
<input type = radio name = r1 value = jam>Jam
<input type = radio name = r1 value = butter>Butter
<input type = button name = bt1 value = find>
</form>
</body>
</html>

```

In the above example the radio button has the same name and when any one option is clicked the value of that is retrieved on the click event of the button. The same example can be tried by changing the type as checkbox and observe the output.

E.g. 2:

```

<HTML>
<head>
<script language = vbscript>
<!--
sub s1_onchange
t2.value=s1.value
end sub

sub r11_onclick
t3.value=r11.value
end sub

sub r12_onclick
t3.value=r12.value
end sub

Sub bt1_onclick
Select Case true
Case c11.checked and c12.checked
    MsgBox c11.value & c12.value
Case c11.checked

```

```

        MsgBox c11.value
    Case c12.checked
        MsgBox c12.value
    Case else
        MsgBox "you have not selected your hobbies"
    End select

    If t1.value = "radiant" and pwd.value = "pepsi" then
        MsgBox " you are an authorized user"
    Else
        MsgBox "you are unauthorized"
    End if
End sub
-->
</script>
</head>
<body>

```

```

Enter the name <Input type = text name = t1><br>
Enter Password <Input type = password name = pwd><br>
Select city <br>
<Select name="s1" id="s1">
<Option value = chennai>Chennai
<Option value = Mumbai>Mumbai
<Option value = Calcutta>Calcutta
<Option value = delhi>Delhi
</Select>
<Input type = text name =t2>
<br>

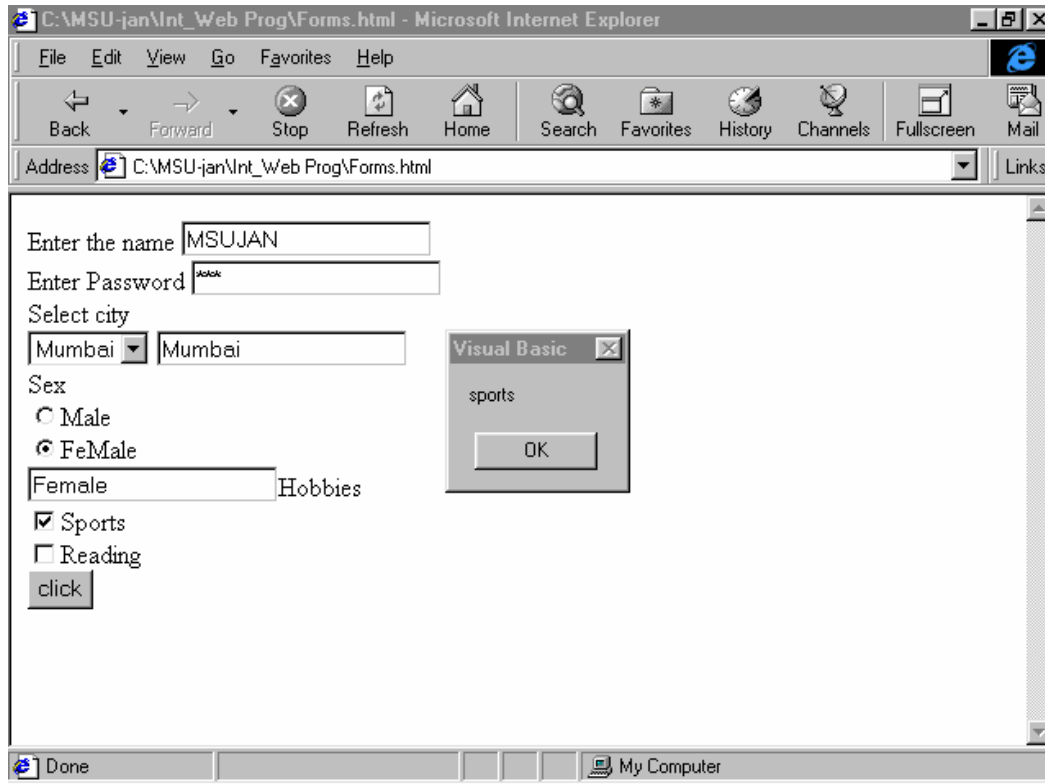
```

```

Sex<br>
<Input type = radio name= r1 id = r11 value = male>Male<br>
<Input type = radio name = r1 id = r12 value = Female>FeMale<br>
<Input type = text name = t3>Hobbies<br>
<Input type = checkbox name = ck1 id = c11 value=sports>Sports<br>
<Input type = checkbox name = ck1 id = c12 value=reading>Reading<br>
<Input type = button name = bt1 value = click to know the hobbies>

</body>
</html>

```



30.4 Hiding Errors

Bugs in a program are always bad. However, they are particularly bad on a Web site, where potentially thousands of people get first hand knowledge of your programming mistakes. However hard we try to avoid them, we are prone to bugs. Let us now consider ways to hide these bugs.

Consider this script:

```
<%
  mystring = "God's delays are not denials"
  mystring = UPPERCASE(mystring)
%>
<%=mystring%>
```

It is to be noted that there is no function UPPERCASE() in VBScript. We only do have UCASE function, which implies that the above script is erroneous. The Error would be appear in a Web page for all the world to see.

Now, consider the following script:

```
<%  
ON ERROR RESUME NEXT  
mystring = "God's delays are not denials"  
mystring = UPPERCASE(mystring)  
>  
<%=mystring%>
```

This script is similar to the previous one except for the single statement ON ERROR RESUME NEXT. On execution, the script executes without an error being reported. This is because, the ON ERROR RESUME NEXT statement forces the script to continue with the next statement when it encounter an error.

If you need to check whether an error has occurred within a script, you can add the following bit of code:

```
<%  
ON ERROR RESUME NEXT  
mystring = "God's delays are not denials"  
mystring = UPPERCASE(mystring)  
IF ERR.NUMBER>0 THEN  
>  
ERR.CLEAR  
END IF  
>  
<%=mystring%>
```

In this case, whenever an error occurs, the error number is recorded in the ERR object. If ERR.NUMBER is greater than zero, you know an error has occurred. Once an error occurs, you should clear it by using the statement ERR.CLEAR. You need to clear the error so you can record a new one if it happens.

30.5 Short Summary

- ◆ We can use the values of the interactive elements in HTML using the syntax Tagname.value
- ◆ If the interactive elements are placed under a form then they must be referred inside the script through their form name
- ◆ To hide errors in our script we use the single statement ON ERROR RESUME NEXT

30.6 Brain Storm

1. How do you trap the ASCII values of the key pressed
2. Explain the referencing of the interactive elements within the script
3. How do you check the existence of errors in your script

Lab Unit (2 Real Time Hours)

1. Design the form as below.
2. If Then Else
3. Find the greatest of three Numbers using the above statement.
4. Select Case
5. Give the Grade in text3 for the mark you have entered in the text1 and text2.
6. Do While Loop
7. Input a number and display all the multiples of 5 till that number.
8. For Next
9. Calculate the factorial of a given number.
10. Do Until
11. Create a code segment to input and find the sum and average of n numbers

Try using all the controls (listbox, option, check etc.. with their Events and perform some operations using those controls.

RADIANT SOFTWARE LIMITED

Text1 Text2 Text3

option2

Option1

CheckBox1 CheckBox2