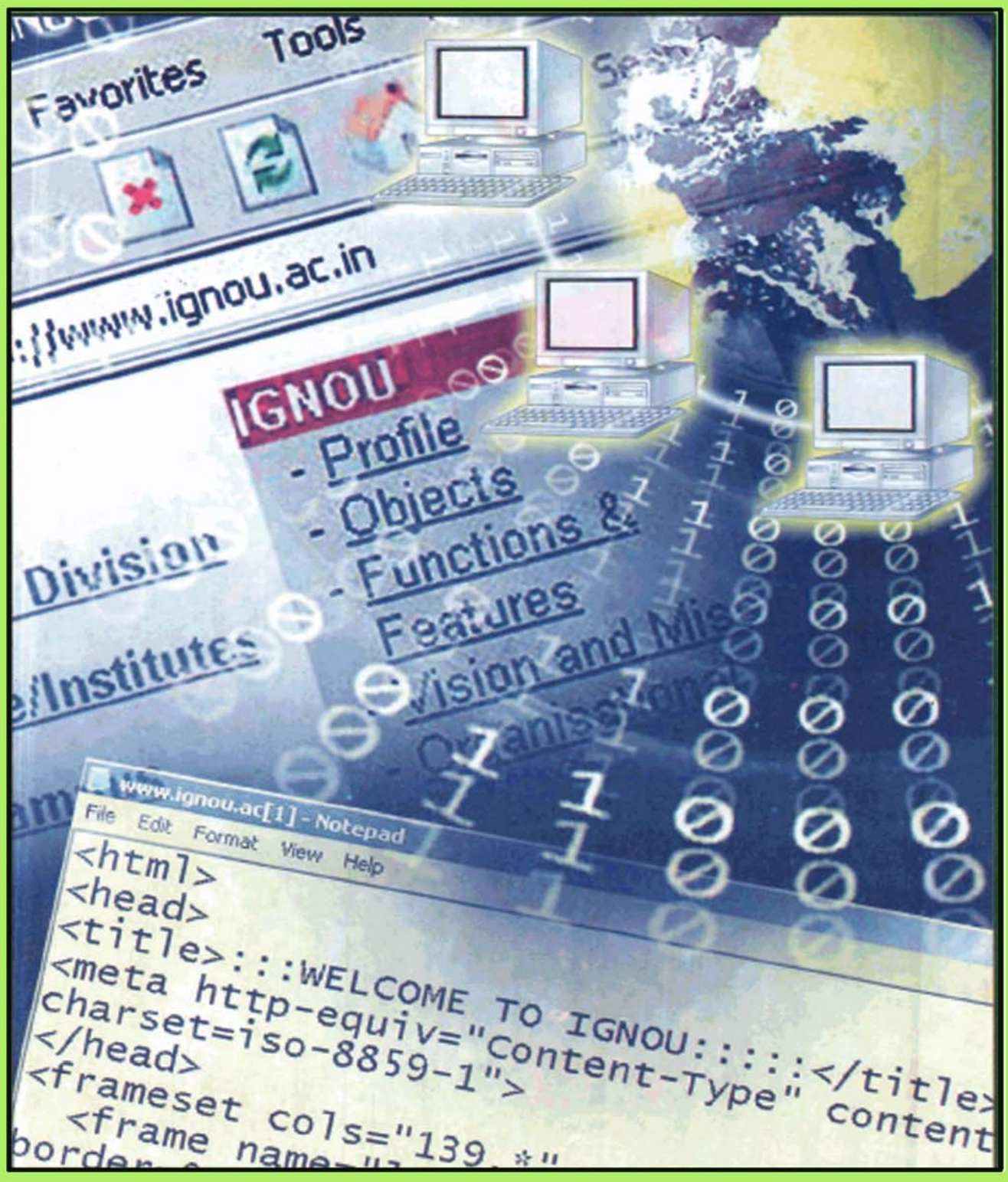
Indira Gandhi National Open University
School of Computer and Information Sciences

MCSL-016
INTERNET CONCEPTS
WEB DESIGN



Scripting Language

1

Indira Gandhi
National Open University
School of Computer and
Information Sciences

MCSL - 016
INTERNET CONCEPTS AND WEB DESIGN

Block

1

SCRIPTING LANGUAGES

Programme / Course Design Committee

Prof. Sanjeev K. Aggarwal, IIT, Kanpur
Prof. M. Balakrishnan, IIT , Delhi
Prof Harish Karnick, IIT, Kanpur
Prof. C. Pandurangan, IIT, Madras
Dr. Om Vikas, Sr. Director, MIT
Prof P. S. Grover, Sr. Consultant,
SOCIS, IGNOU

Faculty of School of Computer and Information Sciences
Shri Shashi Bhushan
Shri Akshay Kumar
Prof Manohar Lal
Shri V.V. Subrahmanyam
Shri P.Venkata Suresh

Print Preparation Team

Block Editor: Milind Mahajani,
Software Consultant, Delhi

Shri. Kamal Uppal
Software Engineer
CRIS, Chanakyapuri
Govt. of India Enterprises

Shri. Shashi Bhushan
SOCIS, IGNOU

Prof. A. K Verma
C-4/1, SFS Flats, Saket, } Language Editor
New Delhi

Course Coordinator: Shashi Bhushan

Block Production Team

Shri H.K Som, SOCIS

Acknowledgements

To all the faculty of SOCIS, IGNOU for their comments on the course material; to Shri Vikas Kumar for help in finalizing the CRC.

May, 2004

ISBN – 81-266-1253-3

Printed and published on behalf on the Indira Gandhi National Open University, New Delhi by The Director, SOCIS.

COURSE INTRODUCTION

Internet is a hot topic today. It has revolutionalised the way we live, work, learn and communicate with each other. It has opened up new gateways of opportunities for education and helped increase productivity, as well efficiency in areas like healthcare, banking, governance, and manufacturing all over the world.

As IT Professionals we have to deal with the whole range of Web technologies starting from the Internet, i.e. to HTML, Advanced HTML, Java Script, VB Script, XML, COM, JSP, ASP, Dream weaver, etc. The purpose of the course is to introduce some of these technologies. In the first Block of course, we study how to design specific page and dynamic web pages. We will also examine forms and frames, two of the most important features of the web pages. The second part of the course, we focus on the practical issues. Each topic is covered in the 10 practical sessions of three hours duration each.

BLOCK INTRODUCTION

This block introduces three major topics namely, Internet, Scripting Languages, and HTML Editor. Internet has been discussed in the first unit and remaining four units focuses on Scripting Language and the last unit is about Dreamweaver, which is a HTML Editor. The block is organized as follows:

Unit 1: In this unit, we describe about the classification of computer network and then describes how does the Internet work? We also take up practical issues like how to configure a browser, how to connect to Internet etc. Internet is based on two standard protocols TCP and IP, which has also been discussed in brief. At the end of this unit we describe different types of services provided by the Internet.

Unit 2: This unit is about HTML, which is a Scripting Language. When we connect to a Web site, the Web server on the remote computer presents our browser with a file in a special format. The contents of the file are stored in a special format-using HTML. The unit introduces a large number of tags.

Unit 3: Describes how to make interactive Web sites for which we require features like, frames, tables, and forms for accepting user input.

Unit 4: In this unit we are going to learn about another Scripting Language, JavaScript. This is mainly used for validating forms. The Scripting Language is based on an object model. We will also discuss how to write JavaScript code and insert name into HTML documents and how to make the Web page more dynamic and interactive.

Unit 5: This unit is about VBScript, a Microsoft Scripting Language. This is the last unit on Scripting Language, which is quite common. VBScript is a Microsoft Scripting Language. It enables us to write programs that enhance the power of Web page by allowing to control their behaviour. It also discusses about objects in VBScript and talk about dictionary object as an example.

Unit 6: The last unit of this block is about HTML Editor, which is a tool for managing and creating a Web site. At the end of this unit we discuss how to choose option, use inspectors and panels, and set preferences that best fit our work styles.

References:

1. *The Internet* by Douglas E. Comer
2. *Web Technologies* by Achyut S Godbole and Atul Kahate
3. *Data Communication & Networking* by Bchrouz A. Forouzan
4. *Computer Networking* by James F. Kurose, Keith, W. Ross, Pearson
5. *Computer Networking* by A. S. Janenbaun, 4th Edition, PHI

UNIT 1 THE INTERNET

Structure **Page No.**

1.0 INTRODUCTION

The Internet is worldwide computer network that interconnects, million of computing devices throughout the world. Most of these devices are PC's, and servers that store and transmit information such as web pages and e-mail messages. Internet is revolutionizing and enhancing the way we as humans communicate, both locally and around the globe. Everyone wants to be a part of it because the Internet literally puts a world of information and a potential worldwide audience at your fingertips.

The Internet evolved from the ARPANET (Advanced Research Projects Agency) to which other networks were added to form an inter network. The present Internet is a collection of several hundred thousand of networks rather than a single network. From there evolved a high-speed backbone of Internet access for sharing these of networks. The end of the decade saw the emergence of the World Wide Web, which heralded a platform-independent means of communication enhanced with a pleasant and relatively easy-to-use graphical interface.

World Wide Web is an example of an **information protocol/service** that can be used to send and receive information over the Internet. It supports:

- **Multimedia Information** (text, movies, pictures, sound, programs . . .).
- **HyperText Information** (information that contains links to other information resources).
- **Graphic User Interface** (so users can point and click to request information instead of typing in text commands).

The **World Wide Web** model follows **Cient/Server** software design. A service that uses client/server design requires two pieces of software to work: **Client Software**, which you use to request information, and **Server Software**, which is an **Information Provider.**

The server software for the World Wide Web is called an **HTTP server** (or informally a Web server). Examples are **Mac HTTP, CERN HTTP, and NCSA HTTP.** The client software for World Wide Web is called a Web browser. Examples are: **Netscape, and Internet Explorer.**

1.1 OBJECTIVES

After going through this unit students should be able to:

- Make classification of networks;
- understand two types of networking models;
- understand the concept of packet switching;
- understand how to access to the internet;
- list the services available on Internet; and
- understand how does the Internet works.

1.2 CLASSIFICATION OF NETWORKS

There are different approaches to the classification of compute Networks. One such classification is based on the distance approach. In this section we will discuss such networks.

The networks can be classified into LAN, MAN and WAN networks. Here, we describe them into brief to understand the difference between the types of network.

Local Area Network (LAN)

LAN is a privately - owned computer networks confined to small geographical area, such as an office or a factory widely used to connect office PCs to share information and resources. In a Local area network two or more computers are connected by same physical medium, such as a transmission cable. An important characteristic of Local Area networks is speed. i.e. they deliver the data very fast compared to other types of networks with typical data transmission speed are 10-100 Mbps.

A wide variety of LANs have been built and installed, but a few types have more recently become dominant. The most widely used LAN system is the Ethernet system. Intermediate nodes (i.e. repeaters, bridges and switches) allow LANs to be connected together to form larger LANs. A LAN may also be connected to another LAN or to WANs and MANs using a “router”.

In summary, a LAN is a communications network, which is:

- local (i.e. one building or group of buildings)
- controlled by one administrative authority
- usually high speed and is always shared

LAN allows users to share resources on computers within an organization.

Metropolitan Area Network (MAN)

A MAN, basically a bigger versions of a LAN is designed to extend over an entire city. It may be single network such as a cable television network, or it may be a means of connecting a number of LANs into a large network so that resources may be

shared for example, a company can use a MAN to connect the LANs in all of its offices throughout a city.

A MAN typically covers an area of between 5 and 50 km diameter. Many MANs cover an area the size of a city, although in some cases MANs may be as small as a group of buildings

The MAN, its communications links and equipment are generally owned by either a consortium of users or by a single network provider who sells the service to the users. This level of service provided to each user must therefore be negotiated with the MAN operator, and some performance guarantees are normally specified.

A MAN often acts as a high-speed network to allow sharing of regional resources (similar to a large LAN). It is also frequently used to provide a shared connection to other networks using a link to a WAN.

Wide Area Network (WAN)

The term Wide Area Network (WAN) usually refers to a network, which covers a large geographical area, and use communications subnets (circuits) to connect the intermediate nodes. A major factor impacting WAN design and performance is a requirement that they lease communication subsets from telephone companies or other communications carriers. Transmission rates are typically 2 Mbps, 34 Mbps, 155 Mbps, 625 Mbps (or sometimes considerably more). The basic purpose of the subnet is to transmit message from one end to another end through intermediate nodes.

In most WAN a subnet consists of two types of elements: (i) Transmission lines (ii) Switching element.

Transmission lines also called channels move about from one machine to another machine. The basic purpose of the switching element is to select the outgoing path for forwarding the message.

Numerous WANs have been constructed, including public switched networks, large corporate networks, military networks, banking networks, stock brokerage networks, and airline reservation networks. A WAN is wholly owned and used by a single company is often referred to as an enterprises network.

1.3 NETWORKING MODELS

There are two types of networking models available: OSI reference Model and the TCP/IP Network Model for the design of computer network system. In this section we shall look at these models.

OSI (Open System Interconnection) Networking Model

An open system is a model that allows any two different systems to communicate regardless of their underlying architecture. The purpose of the OSI model is to open communication between different devices without requiring changes to the logic of the underlying hardware and software.

The OSI model is not a protocol, it is a model for understanding and designing a network architecture that is inter- operable, flexible and robust.
The OSI model has a seven-layered architecture. These are:

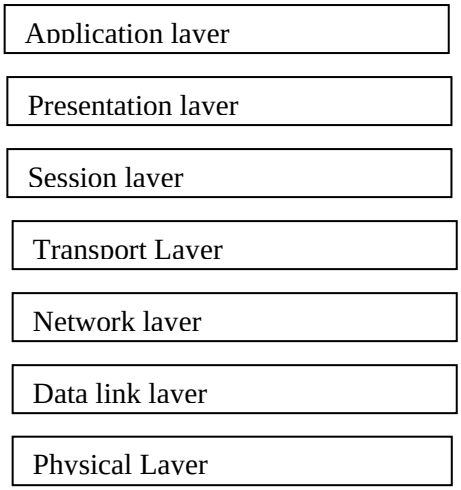


Figure 1: OSI Model

Physical layer: the physical layer is concerned with sending raw bits between the source and destination nodes over a physical medium. The source and destination nodes have to agree on a number of factors.

Signal encoding: how are the bits 0 and 1 to be represented?

Medium: what is the medium used and its properties?

Bit synchronization: is the transmission synchronous or asynchronous?

Transmission type: whether the transmission is serial or parallel?

Transmission mode: is the transmission simplex, half-duplex or full duplex?

Topology: what is the network topology i.e. star, mesh, ring or bus?

Data link layer: the data link layer is responsible for transmitting a group of bits between the adjacent nodes. The group of bits is known as frame. The network layer passes a data unit to the data link layer and data link layer adds the header information to this data unit. The data link layer performs the following functions:

- **Addressing:** Headers and trailers are added containing the physical addresses of the adjacent nodes and removed on a successful delivery.
- **Framing:** Grouping of/bits received from the network layer into manageable units called frame
- **Flow control:** to regulate the amount of data that can be sent to the receiver.
- **Media access control (MAC):** who decide who can send data, when and how much.
- **Synchronization:** this layer also contains bits to synchronize the timing to know the bit interval to recognize the bit correctly.
- **Error control:** it incorpoprates the CRC to ensure the correctness of the frame.
- **Node to node delivery:** it's also responsible for error-free delivery of the entire frame/packet to the next adjacent node.

Network layer: The network layer is responsible for routing a packet within the subnet that is, from source to destination nodes across multiple nodes in the same network or across multiple networks. This layer also ensures the successful delivery of a packet to the destination node. The network layer performs the following functions:

- Routing: To find the optimal route

- Congestion control: which is based on two approaches (i) Increase on the resources (ii) Decrease the word
- Accounting and billing

Transport layer: this layer is the first end-to-end layer. Header of the transport layer contains information that helps send the message to the corresponding layer at the destination node. The message is broken into packets and may travel through a number of intermediate nodes. This layer takes care of error control and flow control both at the source and destination for the entire message. The responsibilities of the transport layer are:

- Host-to-host message delivery
- Flow Control
- Segmentation and reassembly

Session layer: the main functions of this layer are to establish, maintain and synchronize the interaction between two communication hosts. It makes sure that once a session is established it must be closed gracefully. It also checks and establishes connections between the hosts of two different users. The session layer also decides whether both users can send as well as receive data at the same time or whether only one host can send and the other can receive. The responsibilities of session layer are:

- **Sessions and sub sessions:** this layer divides a session into sub session for avoiding retransmission of entire message by adding the checkpoint feature.
- **Synchronization:** this layer decides the order in which data needs to be passed to the transport layer.
- **Dialog control:** this layer also decides which user application sends data and at what point of time and whether the communication is simplex, half duplex or full duplex.
- **Session closure:** this layer ensures that the session between the hosts is closed gracefully.

Presentation layer: when two hosts are communicating with each other they might use different coding standards and character sets for representing data internally. This layer is responsible for taking care of such differences. This layer is responsible for:

- Data encryption and decryption for security
- Compression
- Translation

Application layer: it's the topmost layer in the OSI model, which enables the user to access the network. This layer provides user interface for network applications such as remote login, World Wide Web and FTP. The responsibilities of the application layer are:

- File access and transfer
- Mail services
- Remote login
- World Wide Web

TCP/IP Networking Model

TCP/IP is an acronym for Transmission Control Protocol / Internet Protocol. TCP/ IP is a collection of protocols, applications and services. TCP/IP protocol were developed prior to the OSI model therefore its layers do not match with the OSI model.

The TCP/IP protocol suit is made of the five layers: Physical, data link, network, transport & application. The first four layers provide physical standards network

interface, internetworking and transport mechanism whereas the last layer comprises of the functionalities of the three topmost layers in the OSI model.

1.4 WHAT IS A PACKET SWITCHING?

End systems are connected together by communication links. There are many types of communication links, which are made of different types of physical media, including fiber optics, twisted pair, coaxial cable and radio links. Different links can transmit data at different rates. The link transmission rate is often called the bandwith of the link, which is typically measured in bits/second. The highest the bandwith, the more is the capacity of the channel. End systems are not usually directly attached to each other via a single communication link. Instead, they are indirectly connected to each other through intermediate switching devices known as routers. A router takes a chunk of information arriving on one of its incoming communication links and forwards that chunk of information on one of its outgoing communication links. In the jargon of computer networking, the chunk of information is called a packet. The path that the packet takes from the sending end system, through a series of communication links and routers, to the receiving end system is known as a route or path through the network. Rather than providing a dedicated path between communicating end systems, the Internet uses a technique known as packet switching that allows multiple communicating end systems to share a path, or parts of a path, at the same time. Similar to a router, there is another special machine called gateways used in the network that allows different networks to talk to the Internet, which uses TCP/IP.

Packet switching is used to avoid long delays in transmitting data over the network. Packet switching is a technique, which limits the amount of data that a computer can transfer on each turn. Packet switching allows many communications to proceed simultaneously. Each packet contains a header that specifies the computer to which the packet should be delivered and the destination is specified using computer's address. Computers that share access to a network take turns in sending packets. On each turn, a given computer sends one packet. IP uses this packet switching concept to deliver messages on the Internet If the destination address does not exist on the local network, it is the responsibility of that network's router to route the message one step closer to its destination. This process continues until the destination machine claims the message packet.

1.5 ACCESSING THE INTERNET

Before we can use the Internet, we have to gain access to it. This access is achieved in one of several ways, which we will discuss in this section. Above all, the Internet is a collection of networks that are connected together through various protocols and hardware.

Dial-up Connection: one of the commonest ways of connection to Internet is through dial up connection using a modem and a telephone line. Using these you can connect to a host machine on the Internet. Once connected the telecommunications software allows you to communicate with the Internet host. When the software runs it uses the modem to place a telephone call to a modem that connects to a computer attached to the Internet.

The SLIP (Serial Line Internet Protocol) or PPP (Point to Point Protocol): two protocols; **serial line interface protocol (SLIP)** and the **point-to-point protocol (PPP),** allow a user to dial into the Internet. They convert the normal telephone data stream into TCP/IP packets and send them to the network. With these, the user becomes a peer station on the Internet and has access to all of the Internet's facilities.

Internet Service Providers

As mentioned earlier, nobody truly owns the Internet, but it is maintained by a group of volunteers interested in supporting this mode of information interchange. Central to this control is the Internet service provider (ISP) which is an important component in the Internet system. Each ISP is a network of routers and communication links. The different ISPs provide a variety of different types of network access to the end systems, including 56 Kbps dial-up modem access, residential broadband access such as cable modem or DSL, high-speed LAN access, and wireless access. ISPs also provide Internet access to content providers, connecting Web sites directly to the Internet. To allow communication among Internet users and to allow users to access worldwide Internet content, these lower-tier ISPs are interconnected through national and international upper-tier ISPs, such as Sprint. An upper-tier ISP consists of high-speed routers interconnected with high-speed fiber-optic links. Each ISP network, whether upper-tier or lower-tier, is managed independently, runs the IP protocol (see below), and conforms to certain naming and address conventions.

ISDN (Integrated Services Digital Network) Service

The whole idea of ISDN is to digitize the telephone network to permit the transmission of audio, video and text over existing telephone lines. The purpose of the ISDN is to provide fully integrated digital services to users.

The use of ISDN for accessing the Internet has breathed new life into the ISDN service. ISDN's slow acceptance was due mostly to a lack of a need for its capabilities. Being a digital interface, ISDN has provided a means for accessing web sites quickly and efficiently. In response to this new demand, telephone companies are rapidly adding ISDN services.

The ISDN standard defines three channels types, each with the different transmission rate: bearer channel (B), data channel (D) and hybrid channel (H) (see the following table)

Channel	Data Rate (Kbps)
B	64
D	16, 64
H	384, 1536, 1920

The B channel is defined at a rate of 64 Kbps. It is the basic user channel and can carry any type of digital information in full duplex mode as long as the required transmission does not exceed 64 kbps. A data channel can be either 16 or 64 kbps depending on the needs of the user used to carry control signals for B channels.

Of the two basic rate B channels, one is used to upload data to the Internet and one to download from the Internet. The D Channel assists in setting up connection and maintaining flow control. There are three ways ISDN can be used to interface to the Internet, by using a modem, adaptor, or bridge/router. ISDN modems and adaptors limit access to a single user. Both terminate the line into an ISDN service. The difference between them is that the ISDN modem takes the Internet traffic and pushes it through the computer serial port, while, the faster ISDN adaptor connects directly to the computer's buses.

ISDN bridge/routers allow for local network connections to be made through ISDN to the Internet. The ISDN termination is made into an Ethernet-type LAN so that multiple users can achieve access to the Net through a single access address. Transfer rates between user and the Internet are between 56 and 128 Kbps.

Direct ISP Service through Leased Line

The most costly method of accessing the Internet is to use leased lines that connect directly to the ISP. This will increase access rate to anywhere between 64 K and 1.5 Mbps, depending on the system in use. Equipment called data service units (DSU) and channel service units (CSU) are set up in pairs, one pair at the customer site and the other at the ISP site. There is no phone dialing required since the connection is direct. Also the only protocol needed to complete the access is TCP/IP, for much the same reason. Depending on the transfer rate required and the distance between the sites, cabling between them can be made with fiber optic cables or unshielded twisted-pair (UTP) copper wire.

Cable Modem

One more way of accessing the Internet currently being developed is the use of cable modems. These require that you subscribe to a cable service and allow you two-way communication with the Internet at rates between 100K and 30 Mbps. The cable modem performs modulation and demodulation like any other modem, but it also has a tuner and filters to isolate the Internet signal from other cable signals. Part of the concern for use of the cable modem is to formulate LAN adapters to allow multiple users to access the Internet. A medium access control (MAC) standard for sending data over cable is being formulated by the IEEE 802.14 committee.

1.6 INTERNET PROTOCOLS

A communication protocol is an agreement that specifies a common language two computers use to exchange messages. For example, a protocol specifies the exact format and meaning of each message that a computer can send. It also specifies the conditions under which a computer should send a given message and how a computer should respond when a message arrives. Different types of protocols are used in Internet such as IP and TCP. A computer connected to the Internet needs both TCP and IP software. IP provides a way of transfering a packet from its source to destination and TCP handles the lost datagrams and delivery of datagrams. Together, they provide a reliable way to send data across the Internet. We discuss about these protocols in brief in the following section.

1.6.1 Internet Protocol (IP)

The Internet protocol specifies the rules that define the details of how computers communicate. It specifies exactly how a packet must be formed and how a router must forward each packet on toward its destination. Internet Protocol (IP) is the protocol by which data is sent from one computer to another on the Internet. Each computer (known as a host) on the Internet has at least one IP address that uniquely identifies it from all other computers on the Internet. When sending or receiving data, the message gets divided into little chunks called packet. Each of these packets contains both the senders Internet address and the receiver’s address. The packet that follows the IP specification is called an IP datagram. The Internet sends an IP datagram across a single network by placing it inside a network packet. For network the entire IP datagram is data. When the network packet arrives at the next computer, the computer opens the packet and extracts the datagram. The receiver examines the destination address on the datagram to determine how to process it. When a router, determines that the datagram must be sent across another network, the router creates a new network packet, encloses the datagram inside the packet and sends the packet across another network toward its destination. When a packet carrying a datagram arrives at its final destination, local software on the machine opens the packet and processes the datagram. Because a message is divided into a number of packets a different route can send each packet across the Internet. Packets can arrive in a different order than the order they were sent in. The Internet Protocol just delivers them. It's up to another

protocol, the Transmission Control Protocol to put them back in the right order. IP is a connectionless protocol, which means that there is no established connection between the end points that are communicating. Each packet that travels through the Internet is treated as an independent unit of data without any relation to any other unit of data. In the Open Systems Interconnection (OSI) communication model, IP is in layer 3, the Networking Layer.

1.6.2 Transmission Control Protocol (TCP)

TCP makes the Internet reliable. TCP solves many problems that can occur in a packet switching system. TCP provide the following facilities:

- TCP eliminates duplicate data.
- TCP ensures that the data is reassembled in exactly the order it was sent
- TCP resends data when a datagram is lost.
- TCP uses acknowledgements and timeouts to handle problem of loss.

The main features of TCP are:

Reliability: TCP ensures that any data sent by a sender arrives at the destination as it was sent. There cannot be any data loss or change in the order of the data. Reliability at the TCP has four important aspects:

- Error Control
- Loss control
- Sequence control
- Duplication control

Connection-oriented: TCP is connection-oriented. Connection-oriented means a connection is established between the source and destination machines before any data is sent i.e. a connection is established and maintained until such time as the message or messages to be exchanged by the application programs at each end have been exchanged. The connections provided by TCP are called Virtual Connections. It means that there is no physical direct connection between the computers.

TCP is used along with the Internet Protocol to send data in the form of message units between computers over the Internet. While IP takes care of handling the actual delivery of the data, TCP takes care of keeping track of the individual units of data (called Packet) that a message is divided into for efficient routing through the Internet. TCP provides for a reliable, connection-oriented data transmission channel between two programs. Reliable means that data sent is guaranteed to reach its destination in the order sent or an error will be returned to the sender.

For example, when an HTML file is sent to someone from a Web server, the Transmission Control Protocol (TCP) program layer in that server divides the file into one or more packets, numbers the packets, and then forwards them individually. Although each packet has the same destination IP address, it may get routed differently through the network. At the other end (the client program in our computer), TCP reassembles the individual packets and waits until they have arrived to forward them as a single file.

TCP is responsible for ensuring that a message is divided into the packets that IP manages and for reassembling the packets back into the complete message at the other end. In the Open Systems Interconnection (OSI) communication model, TCP is in layer 4, the Transport Layer.

Check Your Progress 1

1. State whether True or False:

   a) Internet is a Global network and is managed by a profit-oriented organization.
   b) TCP does not control duplication of packets.
   c) For logging in to Internet you must have an account on its host machine.
   d) A router is used at the network layer.
   e) Internet provides only one-way communication.

2. What are the different layers in the TCP/IP networking Model?
3. Describe the facilities provided by the TCP?

1.7 INTERNET ADDRESS

Addresses are essential for virtually everything we do on the Internet. The IP in TCP/IP is a mechanism for providing addresses for computers on the Internet. Internet addresses have two forms:

- Person understandable which is expressed as words
- Machine understandable which is expressed as numbers

The following can be a typical person understandable address on Internet:

VVS @ ignou.ac.in

VVS is an username which in general is the name of the Internet account. This name is same as the one, which you may use when logging into the computer on which you have your Internet account. Logging in is the process of gaining access to your account on a computer, which is shared by several users. Your Internet account is created on it.

@ Connect "who" with where:

.ignou is a subdomain (could be several in each could be separated by (dot). Last one is referred to a domain).

.edu is a domain top or what part in – It refers to "where" part which is a country code.

1.7.1 Structure of Internet Servers Address

The structure of an Internet server's address keyed into a client's software is as follows:

http://www.microsoft.com

Where:

http is the communication protocol to be used
www is the notation for World Wide Web
.Microsoft is the registered domain Name associated with the IP address of an Internet Server.
.com the server provides commercial services to clients who connect to it.

To help to speed up access, its IP address can be directly represented in form of numbers. 127.57.13.1 instead of the domain name, m*icrosoft.com.* In this case no name resolution needs to take place.

An Internet address is a unique 32-bit number that is typically expressed as four 8-bit octets, with each octet separated by a period. Each of the octets can take on any number from 0 through 255.

Hosts, Domains and Subdomains

Hosts are in general, individual machines at a particular location. Resources of a host machine is normally shared and can be utilized by any user on Internet. Hosts and local networks are grouped together into domains, which then are grouped together into one more larger domains. For an analogy a host computer is considered as an apartment building in a housing complex and your account is just an apartment in it.

Domain may be an apartment complex, a town or even a country. Sub-domains may correspond to organizations such as IGNOU. India comes under a large domain. “IN”. Computers termed as name servers contain database of Internets host addresses. They translate word addresses or persons understandable into numeric equivalents. Let us see another example of Internet address:

http://www.ignou.ac.in

What does it all mean? Actually to the ISP server, very little. The server wants to see something quite different. It wants to see a 32-bit number as an Internet address. Something like this equivalent decimal grouping:

198.168.45.249

The Internet addresses, known as universal resources locators (URL), are translated from one form to the other using an address resolution protocol. The first address is in the form we are most used to and that user use to access an Internet site. In this example, the address is for a website, identified by the hypertext transfer protocol (http), which controls access to web pages. Following http is a delimiter sequence, ://, and identification for the world wide web (www).

The domain name, ignou.ac follows www and identifies the general site for the web.(dot) edu is one example of a domain top, which is a broad classification of web users. Other common domain tops are:

.com for commerce and businesses
.gov for government agencies
.mil for military sites
.org for all kinds of organizations.

Lastly, in this example is a country code, again preceded by a dot. Here we are using in for the India, which is the default country.

Addresses may be followed by subdomains separated by dots or slashes (/) as needed. These addresses are translated into a 32-bit (4 decimal numeric groups) address shown as for http:// www.ignou.ac.in we will further discuss this topic in the next section.

1.7.2 Address Space

Internet addresses are divided into five different types of classes. The classes were designated A through E. class A address space allows a small number of networks but a large number of machines, while class C allows for a large number of networks but a relatively small number of machines per network. The following figure lists five address classes used in classical network addresses.

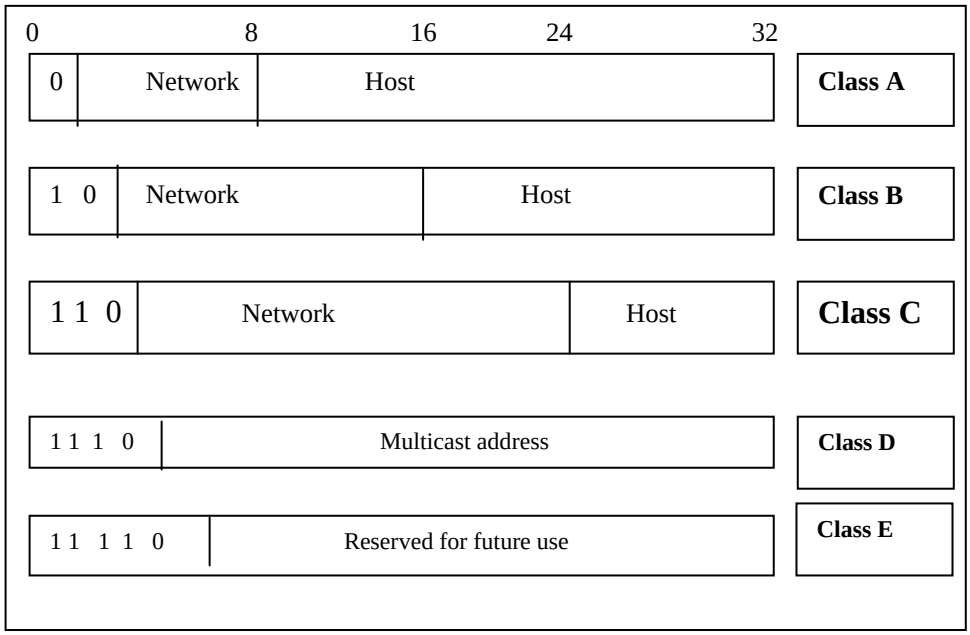


Figure 2: The IP Address Structure

Regardless of the class of address space assigned, organizations assigned a particular class of address will not utilize the entire address space provided. This is especially in the case of class A and Class B address allocation schemes.

Ports

A port is an additional 16-bit number that uniquely identifies the particular service on any given machine on the Internet. Port numbers are 16 bit wide, therefore each computer on the Internet has a maximum number of 2^{16} or 65,536 ports. The particular application is identified by its unique port number in the same way that a specific television station has a unique channel number.

Port numbers are divided into three ranges:

- Well-known ports are those from 0 through 1,023.
- Registered ports are those from 1,024 through 49,151.
- Dynamic and private ports are those from 49,152 through 65,535.

Well-known ports, those ranging from 0 through 1,023 are where most common services on the Internet are residing. These ports are controlled and assigned by the Internet Assigned Number Authority (IANA) and on most systems can be used only by system (root) processes or by programs executed by privileged users.

1.8 HOW DOES THE INTERNET WORK?

As discussed in the previous section every computer connected to the Internet has a unique address. Let's say your IP address is 1.2.3.4 and you want to send a message to the computer with the IP address 5.6.7.8. The message you want to send is "Hello computer 5.6.7.8!" Let's say you've dialed into your ISP from home and the message must be transmitted over the phone line. Therefore the message must be translated from alphabetic text into electronic signals, transmitted over the Internet, and then translated back into alphabetic text. How is this accomplished? Through the use of a **protocol stack**. Every computer needs one to communicate on the Internet and it is usually built into the computer's operating system (i.e. Windows, Unix, etc.). The protocol stack used on the Internet is refered to as the TCP/IP protocol stack, which was discussed in section 1.3.

If we were to follow the path that the message "Hello computer 5.6.7.8!" took from our computer to the computer with IP address 5.6.7.8, it would happen something like this:

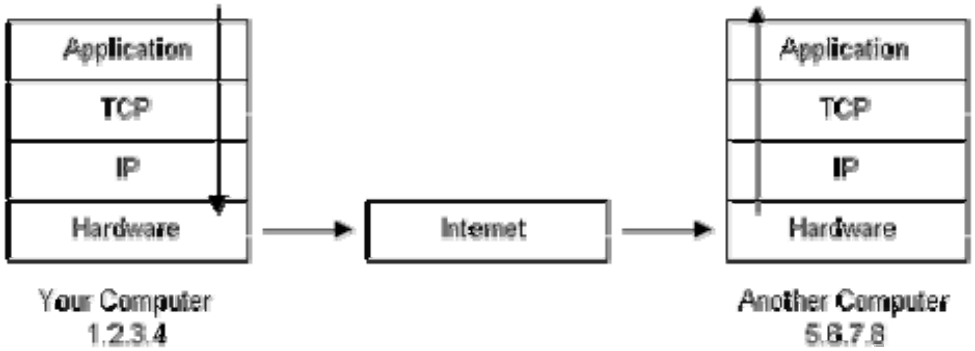


Figure 3: Environment of the Packet Flow

1. The message would start at the top of the protocol stack on your computer and work it's way downward.
2. If the message to be sent is long, each stack layer that the message passes through may break the message up into smaller chunks of data. This is because data sent over the Internet (and most computer networks) are sent in manageable chunks. On the Internet, these chunks of data are known as **packets**.
3. The packets would go through the Application Layer and continue to the TCP layer. Each packet is assigned a **port number**, which is used by program on the destination computer to receive the message because it will be listening on a specific port.
4. After going through the TCP layer, the packets proceed to the IP layer. This is where each packet receives it's destination address, 5.6.7.8.
5. Now that our message packets have a port number and an IP address, they are ready to be sent over the Internet. The hardware layer takes care of turning our packets containing the alphabetic text of our message into electronic signals and transmitting them over the phone line.
6. On the other end of the phone line your ISP has a direct connection to the Internet. The ISPs **router** examines the destination address in each packet and determines where to send it. Often, the packet's next stop is another router. More on routers and Internet infrastructure later.
7. Eventually, the packets reach computer 5.6.7.8. Here, the packets start at the bottom of the destination computer's TCP/IP stack and work upwards.
8. As the packets go upwards through the stack, all routing data that the sending computer's stack added (such as IP address and port number) is stripped from the packets.
9. When the data reaches the top of the stack, the packets have been re-assembled into their original form, "Hello computer 5.6.7.8!"

1.9 INTRANET AND EXTRANET

Intranets are basically “small” Internets. They use the same network facilities that the Internet does, but access is restricted to a limited sphere. For instance, a company can set up an intranet within the confines of the company itself. Access can be tightly controlled and limited to authorized employees and staff. There is no connection to the Internet or any other outside network. Functions like web sites, file uploads and downloads, and e-mail is available on intranets within the confines of the network. Since frivolous sites are no longer available, there is no employee time lost due to accessing them. There is, of course, the limitation of the networking area. The very benefit of restricting access to all of the facilities available on the Internet also restricts communication to other desirable locations. This is where the extranet steps in.

An extranet is network that connects a number of intranets into a truly mini-Internet Access is extended to all the intranets connected through the extranet, but, again, not to the Internet. Extranets requires a constant Internet connection and a hypertext transfer protocol (http) server.

Extranets can also be used to connect an intranet to the Internet so that remote offsite access can be made into a company's intranet by an authorized individual. This can facilitate through an extranet.

Basically, it uses passwords and smart cards to log in to a gateway server that checks the requester's security credentials. If the user checks out, he or she is allowed access into the company's intranet structure.

A number of URL address are set aside for intranet and extranet use. Essentially because intranets are self-contained networks, the same set of addresses can be used by all intranets without conflict. Extranet addresses are designed to recognize the intranets they connect and correctly preface each intranet address with an identifier. This allows two interconnected intranets to retain the same set of address values and keep them from being mistaken. One class A address, ranging from 10.0.0.0 to 10.255.255.255 is reserved for intranet usage. Again, since an intranet is a self-contained system, it only needs one class A network to designate the main network. Subnetworks use reserved class B and class C addresses. There are 16 class B addresses, from 172.16.0.0 ti 172.31.255.255 and 256 class C addresses, which range from 192.168.0.0 to 192.168.255.255.

1.10 INTERNET INFRASTRUCTURE

So now you know how packets travel from one computer to another over the Internet. But what's in-between? What actually makes up the Internet infrastructure or backbone?

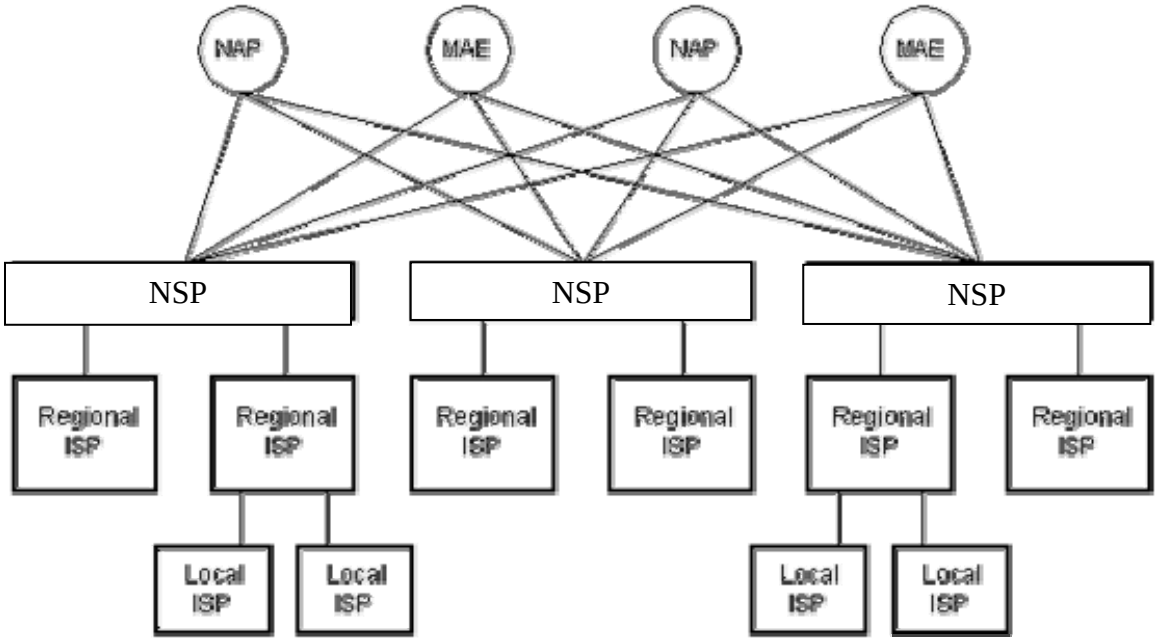


Figure 4: Internet Backbone

The Internet backbone is made up of many large networks, which interconnect with each other. These large networks are known as **Network Service Providers** or **NSP**s. These networks **peer** with each other to exchange packet traffic. Each NSP is required to connect to **Network Access Points** or **NAP**s. At the NAPs, packet traffic may jump from one NSP's backbone to another NSP's backbone. NSPs also interconnect at **Metropolitan Area Exchanges** or **MAE**s. MAEs serve the same purpose as the

NAPs but are privately owned. NAPs were the original Internet interconnects points. Both NAPs and MAEs are referred to as Internet Exchange Points or **IX**s. NSPs also sell bandwidth to smaller networks, such as ISPs and smaller bandwidth providers. Below is a picture showing this hierarchical infrastructure.

This is not a true representation of an actual piece of the Internet. The above figure is only meant to demonstrate how the NSPs could interconnect with each other and smaller ISPs. None of the physical network components are shown in this figure. This is because a single NSP's backbone infrastructure is a complex drawing by itself. Most NSPs publish maps of their network infrastructure on their web sites and can be found easily. To draw an actual map of the Internet would be nearly impossible due to it's size, complexity, and ever changing structure.

The Internet Routing Hierarchy

So how do packets find their way across the Internet? Does every computer connected to the Internet know where the other computers are? Do packets simply get 'broadcast' to every computer on the Internet? The answer to both the preceeding questions is 'no'. No computer knows where any of the other computers are, and packets do not get sent to every computer. The information used to get packets to their destinations is contained in routing tables kept by each router connected to the Internet.

Routers are packet switches. A router is usually connected between networks to route packets between them. Each router knows about it's sub-networks and which IP addresses they use. The router usually doesn't know what IP addresses are 'above' it. Examine the figure below. The black boxes connecting the backbones are routers. The larger NSP backbones at the top are connected at a NAP. Under them are several sub-networks, and under them, more sub-networks. At the bottom are two local area networks with computers attached.

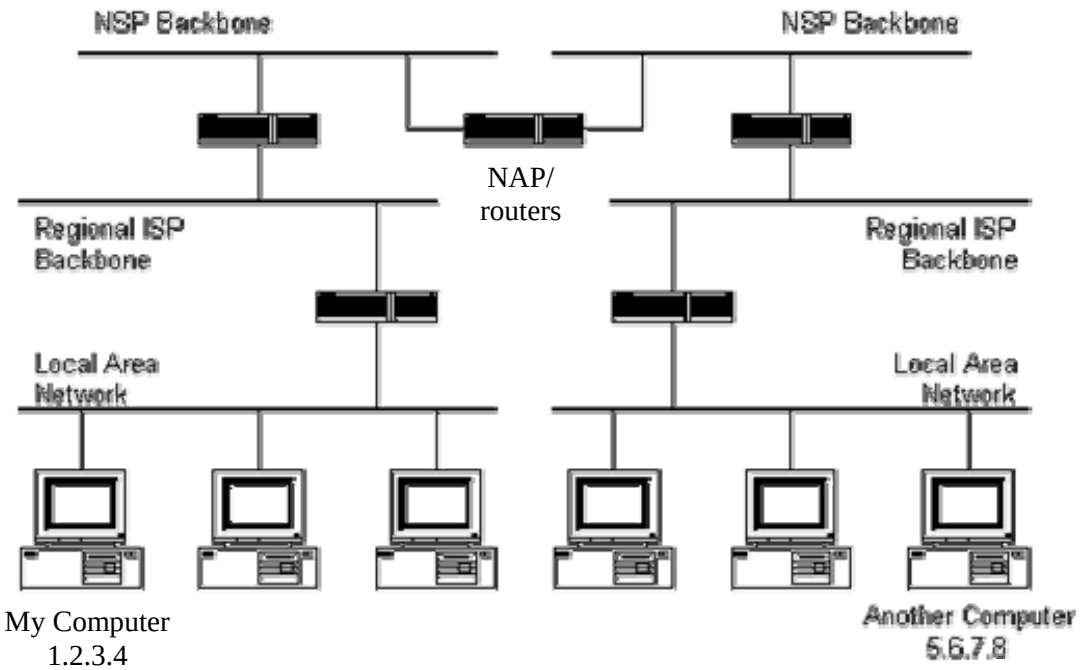


Figure 5: Routes Connecting in Network

When a packet arrives at a router, the router examines the IP address put there by the IP protocol layer on the originating computer. The router checks it's routing table. If the network containing the IP address is found, the packet is sent to that network. If the network containing the IP address is not found, then the router sends the packet on a default route, usually up the backbone hierarchy to the next router. Hopefully the

next router will know where to send the packet. If it does not, again the packet is routed upwards until it reaches a NSP backbone. The routers connected to the NSP backbones hold the largest routing tables and here the packet will be routed to the correct backbone, where it will begin its journey 'downward' through smaller and smaller networks until it finds it's destination.

Domain Names and Address Resolution

But what if you don't know the IP address of the computer you want to connect to? What if you need to access a web server referred to as *www.anothercomputer.com*? How does your web browser know where on the Internet this computer lives? The answer to all these questions is the **Domain Name Service** or **DNS**. The DNS is a distributed database, which keeps track of computer's names and their corresponding IP addresses on the Internet.

Many computers connected to the Internet host part of the DNS database and the software that allows others to access it. These computers are known as DNS servers. No DNS server contains the entire database; they only contain a subset of it. If a DNS server does not contain the domain name requested by another computer, the DNS server re-directs the requesting computer to another DNS server.

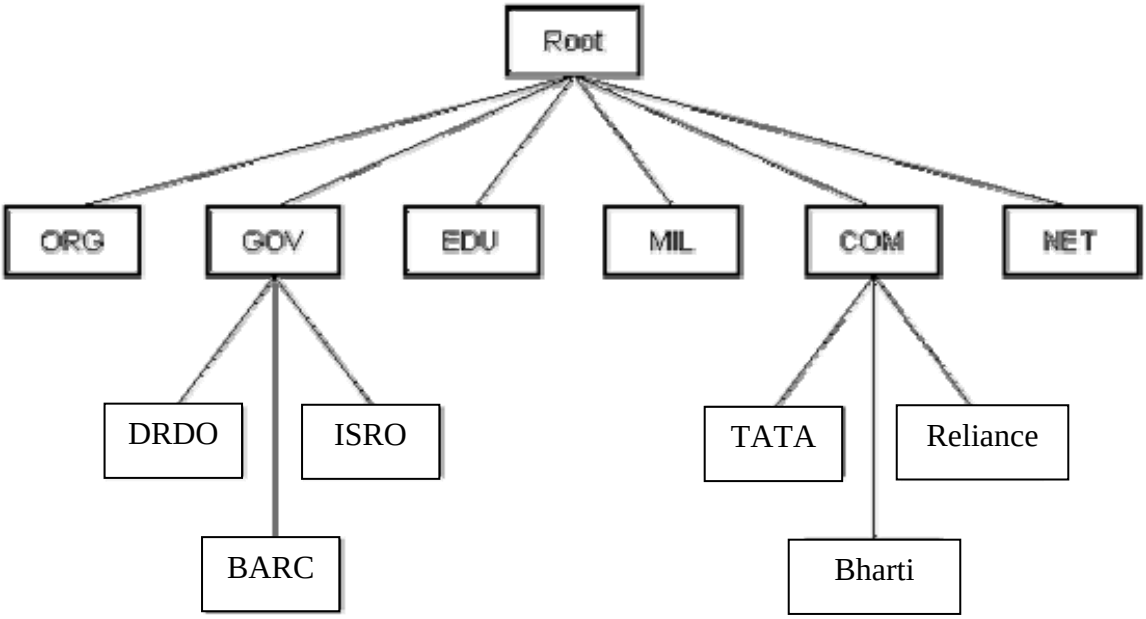


Figure 6: DNS Hierarchy

The Domain Name Service is structured as a hierarchy similar to the IP routing hierarchy. The computer requesting a name resolution will be re-directed 'up' the hierarchy until a DNS server is found that can resolve the domain name in the request. Figure 6 illustrates a portion of the hierarchy. At the top of the tree are the domain roots. Some of the older, more common domains are seen near the top. What is not shown are the multitude of DNS servers around the world which form the rest of the hierarchy.

When an Internet connection is setup (e.g. for a LAN or Dial-Up Networking in Windows), one primary and one or more secondary DNS servers are usually specified as part of the installation. This way, any Internet applications that need domain name resolution will be able to function correctly. For example, when you enter a web address into your web browser, the browser first connects to your primary DNS server. After obtaining the IP address for the domain name you entered, the browser then connects to the target computer and requests the web page you wanted.

1.11 PROTOCOLS AND SERVICES ON INTERNET

To work with Internet and to utilize its facilities we use certain tools. For example, Telnet is a tool, which is utilized for logging on remote computers on the Internet. Let us briefly discuss about some of the important tools and services.

1.11.1 Domain Name System

Domain name is a name given to a network for ease of reference. Domain refers to a group of computers that are known by a single common name. Somebody has to transfer these domain names into IP addresses. It is decided on the physical location of the web server as well as where the domain name is registered. Some generic domain names are:

Domain name	Description
Com	Commercial organization
Edu	Educational organization
Gov	Government organization
Mil	Military group
Org	Non-profit organization

Thus, humans use domain names when referring to computers on the Internet, whereas computers work only with IP addresses, which are numeric. DNS was developed as a distributed database. The database contains the mappings between the domain names and IP addresses scattered across different computers. This DNS was consulted whenever any message is to be sent to any computer on the Internet. DNS is based on the creation of the hierarchical domain based naming architecture, which is implemented as a distributed database. It is used for mapping host names and email addresses to IP addresses. Each organization operates a domain name server that contains the list of all computers in that organization along with their IP addresses. When an application program needs to translate a computer's name into the computer's IP address, the application becomes a client of the DNS. It contacts a domain name server and sends the server an alphabetic computer name then the server returns the correct IP address. The domain name system works like a directory. A given server does not store the names and addresses of all possible computers in the Internet. Each server stores the name of the computers at only one company or enterprise.

1.11.2 SMTP and Electronic Mail

One of the very useful things about Internet is that it allows you almost instantly exchange of electronic message (e-mail) across the worlds. E-mail is a popular way of communication on the electronic frontier. You can E-mail to your friend or a researcher or anybody for getting a copy of a selected paper. Electronic mail system provides services that allowed complex communication and interaction. E-mail provide the following facilities:

- Composing and sending/receiving a message.
- Storing/forwarding/deleting/replying to a message.
- Sending a single message to more than one person.
- Sending text, voice, graphics and video.
- Sending a message that interacts with other computer programs.

Another commonly used Internet service is electronic mail. E-mail uses an application level protocol called **Simple Mail Transfer Protocol** or **SMTP**. SMTP is also a text-based protocol, but unlike HTTP, SMTP is connection oriented. SMTP is also more complicated than HTTP.

When you open your mail client to read your e-mail, this is what typically happens:

1. The mail client (Netscape Mail, Lotus Notes, Microsoft Outlook, etc.) opens a connection to it's default mail server. The mail server's IP address or domain name is typically setup when the mail client is installed.
2. The mail server will always transmit the first message to identify itself.
3. The client will send an SMTP HELO command to which the server will respond with a 250 OK message.
4. Depending on whether the client is checking mail, sending mail, etc. the appropriate SMTP commands will be sent to the server, which will respond accordingly.
5. This request/response transaction will continue until the client sends an SMTP QUIT command. The server will then say goodbye and the connection will be closed.

Similarly, when you send an e-mail message your computer sends it to an SMTP server. The server forwards it to the recipients mail server depending on the email address. The received message is stored at the destination mail server until the addressee retrieves it. To receive E-mail a user Internet account includes an electronic mailbox. A message sent for you is received at your Internet host computer, where it is stored in your electronic mailbox. As soon as you login into your Internet account, one of the first things you should do is to check your mailbox.

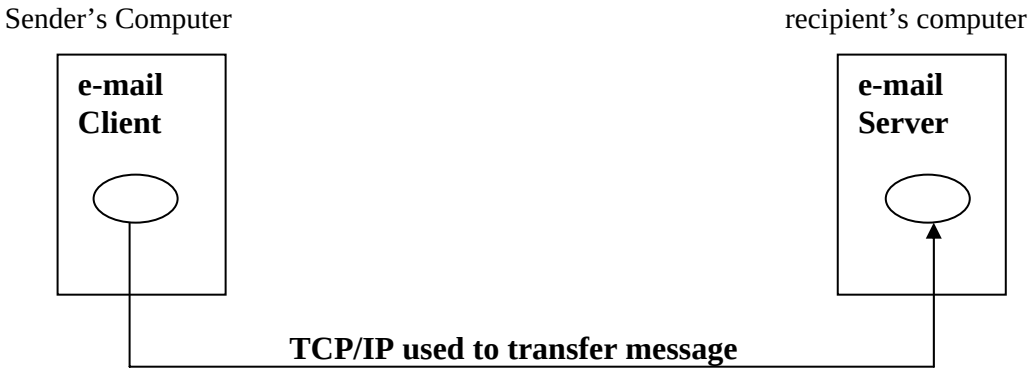


Figure 7: An e-mail transfer across the Internet uses two programs: client and server

E-mail system follows the client-server approach to transfer messages across the Internet. When a user sends an E-mail message a program on the sender's computer becomes a client. It contacts an e-mail server program on the recipient's computer and transfers a copy of the message. Some of the mail programs those exist on Internet are UCB mail, Elm, Pine etc. However, one thing, which you must emphasize while selecting a mail program, is the user friendliness of that program. Through E-mail on Internet you can be in direct touch of your friend and colleagues.

Mailing lists on Internet

Another exciting aspect about the E-mail is that you can find groups of people who share your interests-whether you are inclined toward research, games or astronomy. E-mail provides a mechanism for groups of people who have shared interests to establish and maintain contact. Such interest groups are referred to as *mailing lists* (lists for short). After all they are mailing lists of the members e-mail addresses. You can subscribe to any of such lists. You will receive copies of all the mail sent to the list. You can also send mail to al subscribers of the list.

1.11.3 Http and World Wide Web

One of the most commonly used services on the Internet is the World Wide Web (WWW). The application protocol that makes the web work is **Hypertext Transfer**

Protocol or **HTTP**. Do not confuse this with the Hypertext Markup Language (HTML). HTML is the language used to write web pages. HTTP is the protocol that web browsers and web servers use to communicate with each other over the Internet. It is an application level protocol because it sits on top of the TCP layer in the protocol stack and is used by specific applications to talk to one another. In this case the applications are web browsers and web servers.

HTTP is a connectionless text based protocol. Clients (web browsers) send requests to web servers for web elements such as web pages and images. After the request is serviced by a server, the connection between client and server across the Internet is disconnected. A new connection must be made for each request. Most protocols are connection oriented. This means that the two computers communicating with each other keep the connection open over the Internet. HTTP does not however. Before an HTTP request can be made by a client, a new connection must be made to the server.

When you type a URL into a web browser, this is what happens:

1. If the URL contains a domain name, the browser first connects to a domain name server and retrieves the corresponding IP address for the web server.
2. The web browser connects to the web server and sends an HTTP request (via the protocol stack) for the desired web page.
3. The web server receives the request and checks for the desired page. If the page exists, the web server sends it. If the server cannot find the requested page, it will send an HTTP 404 error message. (404 means 'Page Not Found' as anyone who has surfed the web probably knows.)
4. The web browser receives the page back and the connection is closed.
5. The browser then parses through the page and looks for other page elements it needs to complete the web page. These usually include images, applets, etc.
6. For each element needed, the browser makes additional connections and HTTP requests to the server for each element.
7. When the browser has finished loading all images, applets, etc. the page will be completely loaded in the browser window.

Most Internet protocols are specified by Internet documents known as a **Request For Comments** or **RFC**s. RFCs may be found at several locations on the Internet.

WWW is an Internet navigation tool that helps you to find and retrieve information links to other WWW pages. The WWW is a distributed hypermedia environment consisting of documents from around the world. The documents are linked using a system known as hypertext, where elements of one document may be linked to specific elements of another document. The documents may be located on any computer connected to the Internet. The word “document” is not limited to text but may include video, graphics, databases and a host of other tools.

The World Wide Web is described as a “wide area hypermedia information initiative among to give universal access to large universe of documents”. World Wide Web provides users on computer networks with a consistent means to access a variety of media in a simplified fashion. A popular software program to search the Web is called **Mosaic,** the Web project has modified the way people view and create information. It has created the first global hypermedia network.

Once again the WWW provides an integrated view of the Internet using clients and servers. As discussed earlier, clients are programs that help you seek out information while servers are the programs that find information to the clients. WWW servers are placed all around the Internet.

The operations of the Web mainly rely on **hypertext** as its means of interacting with users. But what is hypertext? Hypertext as such is the same as regular text that is it

can be written, read, searched or edited; however, hypertext contains connections within the text to other documents. The hypertext links are called **hyperlinks.** These hyperlinks can create a complex virtual web of connections.

Hypermedia is an advanced version of hypertext documents as it contains links not only to other pieces of text but also to other forms of media such as sounds, images and movies. Hypermedia combines hypertext and multimedia.

1.11.4 Usenet and Newsgroups

In Internet there exists another way to meet people and share information. One such way is through Usenet *newsgroups*. These are special groups set up by people who want to share common interests ranging from current topics to cultural heritages. These are currently thousands of Usenet newsgroups.

The Usenet can be considered as another global network of computers and people, which is interwined with the Internet. However, Usenet does not operate interactively like the Internet, instead Usenet machines store the messages sent by users. Unlike mail from mailing lists, the news articles do not automatically fill your electronic mailbox. For accessing the information on newsnet, one needs a special type of program called a newsreader. This program help in retrieving only the news you want from Usenet storage site and display it on your terminal. Usenet is like living thing, New newsgroups gets added, the groups which have too much traffic get broken up into smaller specialized groups, the groups even can dissolve themselves. However, all of this occurs based on some commonly accepted rules and by voting. For Usenet, there is no enforcement body; it entirely depends on the cooperation of its computers owners and users.

The newsgroups are really meant fro interaction of people who share your interests. You can post your own questions as well as your answers to the questions of others, on the Usenet. One thing, which is worth mentioning here, is that when one is interacting wit people on Internet certain mannerism should be adopted. These rules are sometimes called “netiquette”. In a face-to-face conversation you can always see a person’s facial gestures and hand movements and can ascertain whether he is teasing or is being sarcastic or sometimes even lying. However, in on-line interaction one cannot see the person one is interacting with. The rules of netiquette may help to compensate some of these limitations of this on-line environment.

1.11.5 FTP

FTP (File Transfer Protocol), a standard Internet protocol, is the simplest way to exchange files between computers on the Internet. Like the Hypertext Transfer Protocol (Hypertext Transfer Protocol), which transfers displayable Web pages and related files, FTP is an application protocol that uses the Internet's TCP/IP protocols. FTP is commonly used to transfer Web page files from their creator to the computer that acts as their server for everyone on the Internet. It's also commonly used to downloading programs and other files to the computer from other servers. However, for such transfer you need an account name on a host and the password. The FTP program will make connection with the remote host, which will help you to browse its directories and mark files for transfer. However, you cannot look at the contents of a file while you are connected via FTP. You have to transfer the copy and then look at it once it is on your own account.

FTP includes many commands but only few are used to retrieve a file. A user needs to understand the three basic commands to connect to remote computer, retrieve a copy of file and exit the FTP program. The commands with their meanings are:

Command	Purpose
Open	connect to a remote computer
get	retrieve a file from the computer
bye	terminate the connection and leave the FTP program

Transferring a file via FTP requires two participants: an FTP client program and FTP server program. The FTP client is the program that we run on our computers. The FTP server is the program that runs on the huge mainframe somewhere and stores tens thousands of files. It is similar to an online library of files. The FTP client can download (receive) or upload (send) files to the FTP server. Using Web browser you can download the files but you can not upload the files. FTP applications will help you to upload the files to the web sites, which you are maintaining.

FTP only understands two basic file formats. It classifies each file either as a text file or a binary file. A text file contains a sequence of characters collected into lines. Although computers used ASCII encoding for text files, FTP includes commands to translate between ASCII and other character encoding. FTP uses the classification binary file for all nontext files. The user must specify binary for any file that contains:

- A computer program
- Audio data
- A graphic or video image
- A spreadsheet
- A document from a word processor
- A compressed file

FTP service compress files to reduce the total amount of disk space the files require. Before transferring a file user must tell FTP that the file contains ASCII text or nontext file. FTP assumes to perform ASCII transfers unless the user enters the binary command.

There are many FTP programs that you can download from the Internet. Windows has its own command line based FTP program. To execute it, select Run from Windows taskbar and type FTP and press enter. By typing open command you can connect to any ftp server. To connect to FTP server you must have a login name and the password. Most of the FTP servers allow anonymous connections. In this case username is anonymous and password is your e-mail address.

Another important FTP program, which is available as a shareware, is WSFTP. Using this window based program it is easier to maintain your web site.

1.11.6 Telnet

TELNET stands for TErminal NETwork. Telnet is both a TCP/IP application and a protocol for connecting a local computer to a remote computer. Telnet is a program that allows an Internet host computer to become a terminal of another host on the Internet. Telnet is the Internet remote login service. Telnet protocol specifies exactly how a remote login interacts. The standard specifies how to client contacts the server and how the server encodes output for transmission to the client. To use the Telnet service, one must invoke the local application program and specify a remote machine. The local program becomes a client, which forms a connection to a server on the remote computer. The client passes keystrokes and mouse movements to the remote machine and displays output from the remote machine on the user’s display screen. Telnet provides direct access to various services on Internet. Some of these services are available on your host, but Telnet is especially useful when these services are not available on your host. For example, if you want to use graphical interfaces designed by other users then Telnet, allows you to access their hosts and use their new interfaces. Similarly, whenever someone creates a useful service on his host, Telnet

allows you to access this valuable information resource. This tools Is especially useful for accessing public services such as library card catalogues, the kind of databases available on the machine etc. You can also log into any catalogue service of a library and use it.

The working of TELNET

1. The commands and characters are sent to the operating system on the common server computer.
2. The local operating system sends these commands and characters to a TELNET client program, which is located on the same local computer.
3. The TELNET client transforms the characters entered by the user to an agreed format known as Network Virtual Terminal (NVT) characters and sends them to the TCP/IP protocol stack of the server computer. NVT is the common device between the client and server.
4. The commands and text are first broken into TCP and then IP packets and are sent across the physical medium from the local client computer to the server.
5. At the server computer's end, the TCP/IP software collects all the IP packets, verifies their correctness and reconstructs the original command and handover the commands or text to that computer operating system.
6. The operating system of the server computer hands over these commands or text to the TELNET server program, which is executing on that remote computer.
7. The TELNET server program on the remote server computer then transforms the commands or text from the NVT format to the format understood by the remote computer. The TELNET cannot directly handover the commands or text to the operating system so TELNET hands over the commands/text to the Psuedo-terminal driver.
8. The Pseudo-terminal driver program then hands over the commands or text to the operating system of the remote computer, which then invokes the application program on the remote server.

The working of the TELNET is extremely simple. Suppose you are working as a faculty member of Indira Gandhi National Open University. You have a typical account FACULTY-1 on the IGNOU computer, which is one of the hosts of the Internet. You are selected for academic exchange scholarship to USA. You will get a user account in U.S.A. However, all your colleagues know only your IGNOU account. Thus, using Telnet you can always log on to your account in India for mail your papers for using programs etc.

There are many databases available on the Internet. You can explore these databases using Telnet. There are going to be many Internet services yet to be created. Every year and better means of accessing the treasures of the Internet is appearing in which Telnet is the key for accessing.

Check Your Progress 2

1. State whether True or False:

    a) E-mail can be used to send text, pictures and movies.
    b) Usenet facility is same as that of mailing list facility.
    c) Anonymous FTP allows viewing and retrieving a file from the archieve of a host without having an account on that machine.
    d) Telnet is used for remote login.
    e) An Internet address is a 16-bit number.
    f) Each computer on the Internet has a maximum number of 2^{16} or 65,536 ports.
2. What is an Internet address?
3. What is FTP?
4. What is Telnet?

1.12 INTERNET TOOLS

In this section we shall look at two software tools available on the Internet.

1.12.1 Search Engines

Search Engines are programs that search the web. Web is a big graph with the pages being the nodes and hyperlinks being the arcs. Search engines collect all the hyperlinks on each page they read, remove all the ones that have already been processed and save the rest. The Web is then searched breadth-first, i.e. each link on page is followed and all the hyperlinks on all the pages pointed to are collected but they are not traced in the order obtained. Automated search is the service that is provided by Search engines. An automated search service allows an individual to find information that resides on remote computers. Automated search systems use computer programs to find web pages that contain information related to a given topic. It allows to locate:

- Web pages associated with a particular company or individual
- Web pages that contain information about a particular product.
- Web pages that contain information about a particular topic.

The results of an automated search can be used immediately or stored in a file on disk to use it later. The results of a search are returned in the form of a web page that has a link to each of the items that was found. Automated search is helpful when a user wants to explore a new topic. The automated search produces a list of candidate pages that may contain information. The user reviews each page in the list to see whether the contents are related to topic or not. If so, the user records the location or if not user moves on to the page in the list. Search mechanisms uses a similar method of search as in the telephone book i.e. before any user invoke the search mechanism a computer program contacts computers on the Internet, gathers a list of available information, sorts the list and then stores the result on a local disk on the computer that runs a search server. When a user invokes a search, the user client program that contacts the server. The client sends a request that contains the name the user entered. When the request arrives at the server, it consults the list of file names on its local disk and provides the result.

1.12.2 Web Browser

A Web browser is software program that allows you to easily display Web pages and navigate the Web. The first graphical browser, Mosaic, was developed in Illinois at the National Center for Supercomputing Applications (NCSA). Each browser displays Web-formatted documents a little differently. As with all commercial software, every program has its own special features.

The two basic categories of Web browser are:

- **Text-only browsers**: A text-only browser such as Lynx allows you to view Web pages without showing art or page structure. Essentially, you look at ASCII text on a screen. The advantage of a text-only browser is that it displays Web pages very fast. There's no waiting for multimedia downloads.

- **Graphical browsers**: To enjoy the multimedia aspect of the Web, you must use a graphical browser such as Netscape Navigator or NCSA Mosaic. Graphical browsers can show pictures, play sounds, and even run video clips. The drawback is that multimedia files, particularly graphics, often take a long time to download. Graphical browsers tend to be significantly slower than their text-only counterparts. And this waiting time can be stretched even further with slow connections or heavy online traffic.

Many different browsers are available for exploring the Internet. The two most popular browsers are Netscape Navigator and Microsoft Internet Explorer. Both of these are graphical browsers, which means that they can display graphics as well as text.

Check Your Progress 3

1. State whether True or False

    a) Surfing means that you are sending for specific information on Internet.
    b) HTML is used for creating home page for World Wide Web.
    c) Hypermedia is same as Hypertext.
    d) Netscape these days is one of the widely used browser.
    e) Windows 95 provides software for browsing Internet.
    f) Hypertext documents contain links to other documents.

1.13 SUMMARY

This unit describes the basic concepts about an Internet. Internet is a network of networks where lot of information is available and is meant to be utilized by you. No one owns the Internet. It consists of a large number of Interconnected autonomous networks that connect millions of computers across the world. The unit describes the various tools available on the Internet and the various services provided by the Internet to users. In this unit we have talked about the Electronic mail Usenet and newsgroups, FTP, Telnet and search engines. We also describe the use of frequently asked questions. The unit also describes the importance of Internet addresses. Addresses are essential for virtually everything we do on the Internet. There are many services available on the Internet for document retrieval. For browsing the Internet there are many browsers available such as Gopher and World Wide Web. Both of these browsers are easy to use and most popular browsing mechanisms on the Internet.

1.14 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. True or false

    a) False
    b) False
    c) True
    d) True
    e) False

2. The TCP/IP networking model has five layers, which are:

    - The Physical Layer
    - The Data link Layer
    - The Network Layer
    - The Transport Layer
    - The Application layer

3. Transmission Control protocol provide various facilities which include:

    - TCP eliminates duplicate data.
    - TCP ensures that the data is reassembled in exactly the order it was sent
    - TCP resends data when a datagram is lost.
    - TCP uses acknowledgements and timeouts to handle problem of loss.

Check Your Progress 2

1. True or false

    a) True
    b) True
    c) True
    d) True
    e) False
    f) True

2. Addresses are essential for virtually everything we do on the Internet. The IP in TCP/IP is a mechanism for providing addresses for computers on the Internet. Internet addresses have two forms:

    - Person understandable which expressed as words
    - Machine understandable which reexpressed as numbers

    Internet addresses are divided into five different types of classes. The classes were designated A through E. class A address space allows a small number of networks but a large number of machines, while class C allows for a large number of networks but a relatively small number of machines per network.

3. FTP (File Transfer Protocol), a standard Internet protocol. It is the simplest way to exchange files between computers on the Internet. FTP is an application protocol that uses the Internet's TCP/IP protocols. FTP is commonly used to transfer Web page files. Transferring a file via FTP requires two participants: an FTP client program and FTP server program. The FTP client is the program that we run on our computers. The FTP server is the program that runs on the huge mainframe somewhere and stores tens thousands of files.

4. Telnet is both a TCP/IP application and a protocol for connecting a local computer to a remote computer. Telnet is the Internet remote login service. Telnet protocol specifies exactly how a remote login interacts. The standard specifies how to client contacts the server and how the server encodes output for transmission to the client. To use the Telnet service, one must invoke the local application program and specify a remote machine.

Check Your Progress 3

1. True or false

    a) False
    b) True
    c) False
    d) False
    e) True
    f) True

UNIT 2 INTRODUCTION TO HTML

Structure **Page No.**

2 INTRODUCTION

You would by now have been introduced to the Internet and the World Wide Web (often just called the Web) and how it has changed our lives. Today we have access to a wide variety of information through Web sites on the Internet. We can access a Web site if we have a connection to the Internet and a browser on our computer. Popular browsers are Microsoft Internet Explorer, Netscape Navigator and Opera. When you connect to a Web site, your browser is presented with a file in a special format by the Web server on the remote computer. The contents of the file are stored in a special format using Hyper Text Markup Language, often called HTML This format is rendered, or interpreted, by the browser and you then see the page of the web site from your computer.

HTML is one language in a class of markup languages, the most general form of which is Standard Generalized Markup Language, or SGML. Since SGML is complex, HTML was invented as a simple way of creating web pages that could be easily accessed by browsers. HTML is a special case of SGML.

HTML consists of tags and data. The tags serve to define what kind of data follows them, thereby enabling the browser to render the data in the appropriate form for the user to see. There are many tags in HTML, of which the few most important ones are introduced in this unit. HTML files usually have the extension ".htm" or ".html".

If you want to create Web pages, you need a tool to write the HTML code for the page. This can be a simple text editor if you are hand-coding HTML. You also have sophisticated HTML editors available that automate many (though not all) of the tasks of coding HTML. You also need a browser to be able to render your code so that you can see the results.

2.1 OBJECTIVES

After going through this unit you should be able to learn:

- basic concepts of HTML;
- basic tags of HTML;
- how to control text attributes such as the font;

- how to work with images in HTML; and
- significance of Meta Tag

The unit covers only the simpler concepts of HTML and does not by any means deal with the subject comprehensively.

2.2 WHAT IS HTML?

As indicated earlier, HTML stands for HyperText Markup Language. HTML provides a way of displaying Web pages with text and images or multimedia content. HTML is not a programming language, but a markup language. An HTML file is a text file containing small markup tags. The markup tags tell the Web browser, such as Internet Explorer or Netscape Navigator, how to display the page. An HTML file must have an htm or html file extension. These files are stored on the web server. So if you want to see the web page of a company, you should enter the URL (Uniform Resource Locator), which is the web site address of the company in the address bar of the browser. This sends a request to the web server, which in turn responds by returning the desired web page. The browser then renders the web page and you see it on your computer.

HTML allows Web page publishers to create complex pages of text and images that can be viewed by anyone on the Web, regardless of what kind of computer or browser is being used. Despite what you might have heard, you don't need any special software to create an HTML page; all you need is a word processor (such as Microsoft Word) and a working knowledge of HTML. Fortunately, the basics of HTML are easy to master. However, you can greatly relieve tedium and improve your productivity by using a good tool. A simple tool is Microsoft FrontPage that reduces the need to remember and type in HTML tags. Still, there can always be situations where you are forced to handcode certain parts of the web page.

HTML is just a series of tags that are integrated into a document that can have text, images or multimedia content. HTML tags are usually English words (such as blockquote) or abbreviations (such as p for paragraph), but they are distinguished from the regular text because they are placed in small angle brackets. So the paragraph tag is <p>, and the blockquote tag is <blockquote>. Some tags dictate how the page will be formatted (for instance, <p> begins a new paragraph), and others dictate how the words appear (<b> makes text bold). Still others provide information - such as the title - that doesn't appear on the page itself. The first thing to remember about tags is that they travel in pairs. Most of the time that you use a tag - say <blockquote> - you must also close it with another tag - in this case, </blockquote>. Note the slash - / - before the word "blockquote"; that is what distinguishes a closing tag from an opening tag.

The basic HTML page begins with the tag <html> and ends with </html>. In between, the file has two sections - the header and the body.

The header - enclosed by the <head> and </head> tags - contains information about a page that will not appear on the page itself, such as the title. The body - enclosed by <body> and </body> - is where the action is. Everything that appears on the page is contained within these tags.

HTML pages are of two types:

- Static
- Dynamic

Static Pages

Static pages, as the name indicates, comprise static content (text or images). So you can only see the contents of a web page without being able to have any interaction with it.

Dynamic Pages

Dynamic pages are those where the content of the web page depend on user input. So interaction with the user is required in order to display the web page. For example, consider a web page which requires a number to be entered from the user in order to find out if it is even or odd. When the user enters the number and clicks on the appropriate button, the number is sent to the web server, which in turn returns the result to the user in an HTML page.

2.3 BASIC TAGS OF HTML

Let us now look at tags in more detail. A <TAG> tells the browser to do something. An ATTRIBUTE goes inside the <TAG> and tells the browser *how* to do it. A tag can have several attributes. Tags can also have default attributes. The *default value* is a value that the browser assumes if you have not told it otherwise. A good example is the font size. The default font size is 3. If you say nothing the size attribute of the font tag will be taken to have the value 3.

Consider the example shown in Fig. 2.1. Type the code specified in the figure in a text editor such as notepad and save it as "fig1.html". To render the file and see your page you can choose one of two ways: 1) Find the icon of the html file you just made (fig1.htm) and double click on it. Or- 2) In Internet Explorer, click on File/Open File and point to the file (fig1.htm).

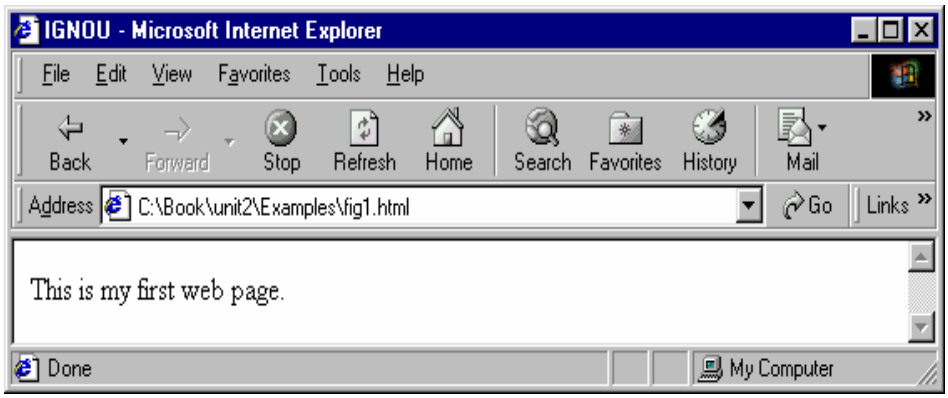


Figure 2.1: A Simple Web Page

```
<HTML>
 <!- - This is a comment - - >
   <HEAD>
     <TITLE> IGNOU      </TITLE>
  </HEAD>
  <BODY>
  This is my first web page.
 </BODY>
</HTML>
```

2.3.1 HTML Tag

As shown in Figure.2.1, <HTML> is a starting tag. To delimit the text inside, add a closing tag by adding a "/" to the starting tag. Most but not all tags have a closing tag. It is necessary to write the code for an HTML page between <HTML> and </HTML>. Think of tags as talking to the browser or, better still, giving it instructions. What you have just told the browser is 'this is the start of an HTML document' (<HTML>) and 'this is the end of an HTML document' (</HTML>). Now you need to put some matter in between these two markers.

Every HTML document is segregated into a HEAD and BODY. The information about the document is kept within <HEAD> tag. The BODY contains the page content.

2.3.2 TITLE Tag

The only thing you have to concern yourselves with in the HEAD tag (for now) is the TITLE tag.

The bulk of the page will be within the BODY tag, as shown in Figure.2.1.

```
<HEAD>
    <TITLE> IGNOU </TITLE>
 </HEAD>
```

Here the document has been given the title IGNOU. It is a good practice to give a title to the document created.

What you have made here is a skeleton HTML document. This is the minimum required information for a web document and all web documents should contain these basic components. Secondly, the document title is what appears at the very top of the browser window.

2.3.3 BODY Tag

If you have a head, you need a body. All the content to be displayed on the web page has to be written within the body tag. So whether text, headlines, textbox, checkbox or any other possible content, everything to be displayed must be kept within the BODY tag as shown in Figure 2.1.

Whenever you make a change to your document, just save it and hit the Reload/Refresh button on your browser. In some instances just hitting the Reload/Refresh button doesn't quite work. In that case hit Reload/Refresh while holding down the SHIFT key.

The BODY tag has following attributes:

a. **BGCOLOUR:** It can be used for changing the background colour of the page. By default the background colour is white.
b. **BACKGROUND:** It is used for specifying the image to be displayed in the background of the page.
c. **LINK:** It indicates the colour of the hyperlinks, which have not been visited or clicked on.
d. **ALINK:** It indicates the colour of the active hyperlink. An active link is the one on which the mouse button is pressed.
e. **VLINK:** It indicates the colour of the hyperlinks after the mouse is clicked on it.
f. **TEXT:** It is used for specifying the colour of the text displayed on the page.

Consider the following example:

```
<HTML>
 <TITLE> IGNOU</TITLE>
  <BODY        BGCOLOUR="#1234567"       TEXT = "#FF0000">
    Welcome to IGNOU
   </BODY>
```

</HTML>

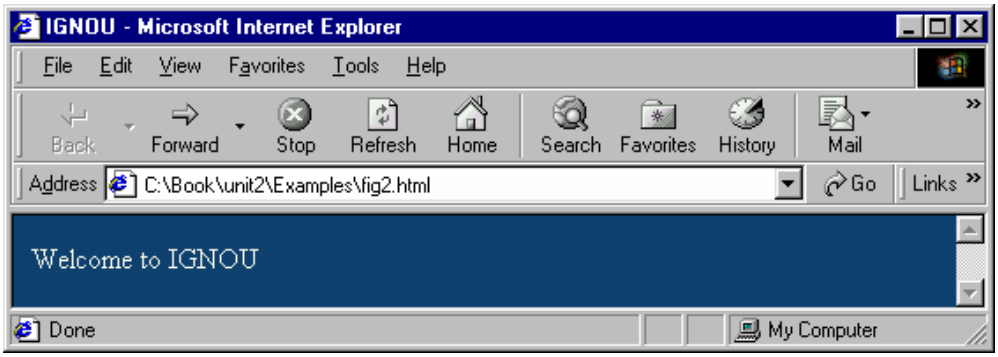


Figure 2.2: A Web Page with a Background Colour

The values specified for BGCOLOUR and TEXT tags indicate the colour of the background of the page and that of the text respectively. These are specified in hexadecimal format. The range of allowable values in this format is from "#000000" to "#FFFFFF". The"#" symbol has to precede the value of the colour so as to indicate to the browser that has to be interpreted as a hexadecimal value. In this six digit value, the first two digits specify the concentration of the colour red, the next two digits specify the concentration of the colour green and the last two digits specify the concentration of the colour blue. So the value is a combination of the primary colours red, green and blue and that is why it is called RGB colour. If we specify the value "#FF0000", the colour appears to be red."#000000" gives black and "#FFFFFF" gives the colour white. You also have the option of specifying the colour by giving its name, like:
<BODY TEXT = "WHITE">.

You can also specify a background image instead. (Note that the image should be in the same folder as your HTML file. More on this below).

```
<HTML>
 <BODY BACKGROUND="swirlies.gif">
   Welcome to INDIA
 </BODY>
</HTML>
```

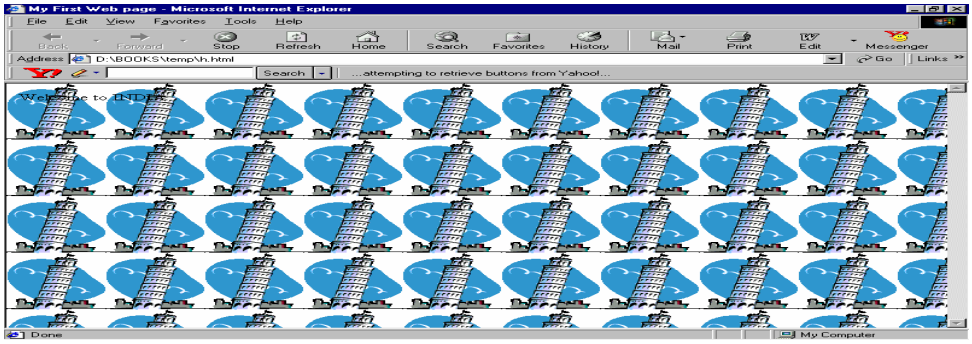


Figure 2.3: A Web Page with an Image in the Background

2.4 FORMATTING OF TEXT

Text formatting, in other words presenting the text on an HTML page in a desired manner, is an important part of creating a web page. Let us understand how we can lay out of text controls its appearance on a page.

2.4.1 Headers

Headers are used to specify the headings of sections or sub-sections in a document. Depending on the desired size of the text, any of six available levels (<H1> to <H6>) of headers can be used. Figure 2.4 shows the usage and varying size of the rendered text depending upon the tag used.

```
<HTML>
 <HEAD>
  <TITLE> IGNOU</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
   <H1> Header Level 1</H1>
   <H2> Header Level 2</H2>
   <H3> Header Level 3</H3>
   <H4> Header Level 4</H4>
   <H5> Header Level 5</H5>
   <H6> Header Level 6</H6>
  </CENTER>
 </BODY>
</HTML>
```

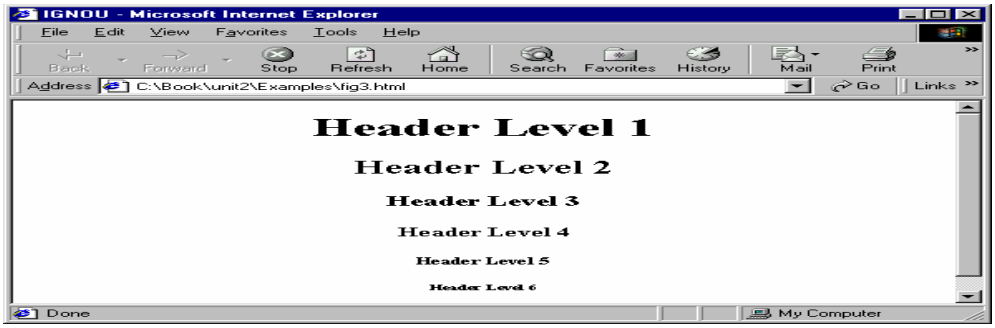


Figure 2.4: Rendering of the Six Header Levels

There is no predefined sequence for using the different levels of the header tags nor any restrictions on which one can be used. So the user has the option of using any level of header tag anywhere in the document. If you want to center text on a page, use the CENTER tag. The text written between <CENTER> and </CENTER> tag gets aligned in the center of the HTML page. As seen in Figure 2.4, the maximum size of the text is displayed using the <H1> tag. So the size goes in decreasing order with the increasing order of the level (i.e. From <H1> to <H6>).

2.4.2 Formatting Tags

Let us now look at some more tags that can be used to format text. These are all given in the example shown in Figure 2.5.

```
<HTML>
 <HEAD>
  <TITLE> IGNOU</TITLE>
 </HEAD>
 <BODY>
   <B> IGNOU </B>
   provides several <I> programmes </I>
   in the <B><I>Computer </I></B> stream.
   <P>
    One of the <I> programmes </I> is <B><U> MCA</U></B>
   </P>
   <B>MCA </B> stands for <TT> Master Of Computer Applications </TT>
   <BR>
```

```
    <S>For MCA</S> <B> IGNOU </B> is considered to be one of the premier
    universities.
    <BR>
    <STRONG>IGNOU</STRONG> believes in <STRONG><EM>
    Quality</EM></STRONG> education
    <BR>
    <P>
According to <CITE> IGNOU, </CITE> <B> MCA<B> is one of its best
programmes offering convenient timings to the student so that s/he can pursue the
course while working at a job.
    </P>
    <BLOCKQUOTE>
For convenience all the courses offered by IGNOU can be seen on its website. A
student has also been provided the flexibility of seeing all the information regarding
admission to the next semester, examination result etc. on its website.
    </BLOCKQUOTE>
    <HR NOSHADE>
    <B> IGNOU contact details are :
    <ADDRESS>
      IGNOU, <BR>
      MAIDAN GARHI, <BR>
      NEW DELHI – 110 068 <BR>
      Website : www.ignou.ac.in
    </ADDRESS>
  </BODY>
</HTML>
```

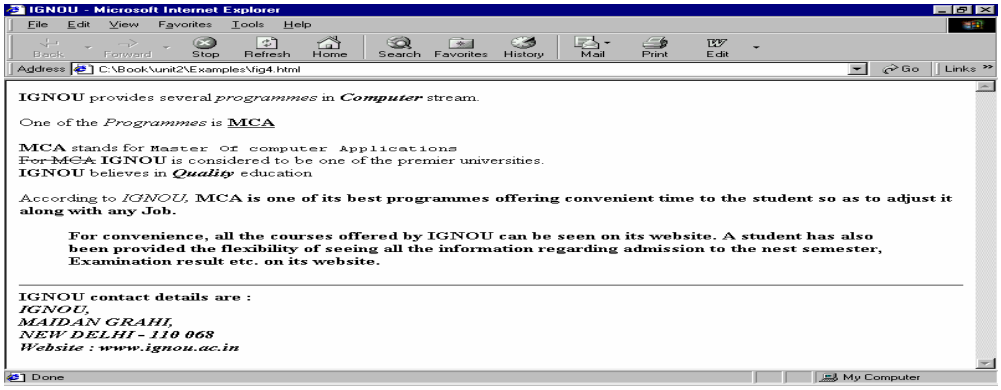


Figure 2.5: An Example showing Various Formatting Tags.

a. **BOLD:** The text can be displayed in boldface by writing it in between the <B> and </B> tags.

b. **ITALICS:** It starts with <I> and ends with </I> tag. It is used to display the text in italics.

c. **UNDERLINE:** It is used for underlining the text to be displayed. The <U> tag is used for this purpose. These tags can be nested. So in order to see the text in boldface, in italics and underlined, it should be placed between the <B><I><U> and </U></I></B> tags. Note that the closing tags are written in reverse order, because any tag used within some other tag should be closed first.

d. **PARAGRAPH:** If you want to display the text in the form of paragraphs, then the <P> tag should be used.

e. **TT:** The <TT> tag is used for displaying text in a fixed width font similar to that of a typewriter.

f. **STRIKE:** If you want the text to be marked with a strikethrough character, place it within the <S> and </S> tags.

g. **STRONG:** There are certain text-based browsers that do not render the text as boldfaced. So you can use the <STRONG> tag instead of the <B> tag. This displays the text to stand out in the most appropriate manner for the browser.

h. **EM:** Just as the <STRONG> tag corresponds to the <B> tag, the <EM> can be used instead of the <I> tag. The reason for using it is the same as for the <STRONG> tag. The <EM> tag is used for emphasizing the text in the manner most appropriate to the browser.

i. **BR:** This tag is used for inserting a line break. The text written after the
 tag is displayed from the beginning of the next line onwards. This tag does not need a corresponding terminating tag.

j. **HR:** This tag puts a horizontal line on the page. It has the following attributes:

- ALIGN: It is used for aligning the line on the page. The possible values of this attribute are LEFT, RIGHT, and CENTER. It is useful when the width of the line is less than the width of the page.
- NOSHADE: This attribute is used to prevent any shading effect.
- SIZE: It is used for specifying the thickness of the line.
- WIDTH: You can set the width of a line using this attribute. The value can be specified either in pixels or as a percentage of the width of the page, e.g., <HR WIDTH = "30%">.

k. **BLOCKQUOTE:** This tag indents the left margin of the text. It is used for displaying the text as quoted text as shown in Figure 2.5.

l. **ADDRESS:** This tag, as shown in Figure 2.5, displays the text in italics.

m. **CITE:** The text placed in between the <CITE> and </CITE> tags is rendered in italics by the browser.

2.4.3 PRE Tag

This tag is used to present the text exactly as written in the code, including whitespace characters. It is terminated by a </PRE> tag. Consider the example shown in Figure 2.6 to understand how this tag works.

```
<HTML>
  <HEAD>
    <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY>
    <PRE>

IGNOU also offers a virtual campus. Studying through the virtual campus is a new
concept in the field of education and this is the first such experiment in India.

While studying through the virtual campus mode, students have access to the
following learning resources and experiences:

• Satellite based interactive tele-conferencing sessions.
• Viewing recorded video sessions.
• Computer based tutorials on CD-ROM.

Printed booklets for specific
```

```
    </PRE>
  </BODY>
</HTML>
```

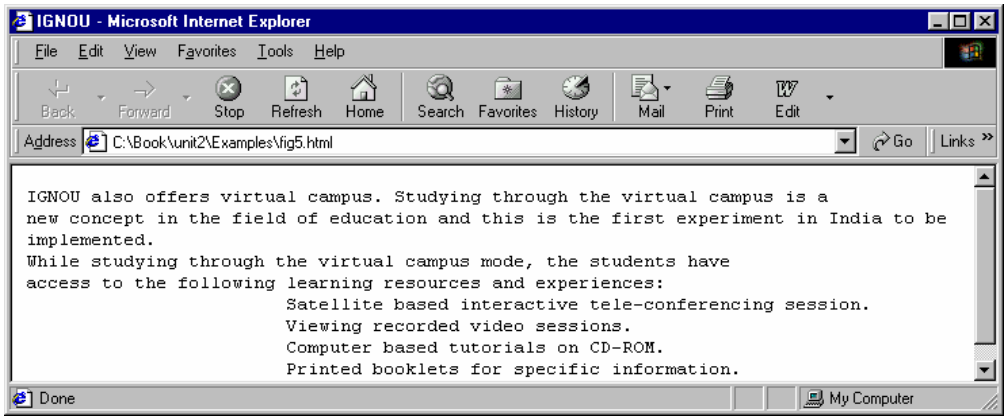


Figure 2.6: Presenting Preformatted Text

As shown in Figure 2.6, the format of the text presented in the browser remains the same as written in the code.

If we do not use the <PRE> tag, the browser condenses the white space when presenting the text on the web page.

2.4.4 FONT Tag

HTML provides the flexibility of changing the characteristics of the font such as size, colour etc. Every browser has a default font setting that governs the default font name, size and colour. Unless you have changed it, the default is Times New Roman 12pt with the colour being black. Now with IE 6.0 (Internet Explorer) you can specify font names other than the default, such as ARIAL and COMIC SANS. Consider the example shown in Figure 2.7.

```
<HTML>
  <HEAD>
    <TITLE> IGNOU </TITLE>
  </HEAD>
  <BODY BGCOLOUR="#FFFFFF">
   <CENTER>

     Welcome to <FONT SIZE=6>INDIA </FONT><BR>
     Welcome to <FONT FACE = “ARIAL” SIZE=6>INDIA </FONT><BR>
     Welcome to <FONT FACE = “ARIAL” SIZE=6 COLOUR = “BLUE”>INDIA
     </FONT><BR>

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

Figure 2.7: Using the FONT Tag

Let us look at the syntax of the <FONT> tag with its different attributes.
<FONT FACE = "name" SIZE = n colour = #RGB >

The attributes are:

a. FACE: This attribute is used to change the font style. Its value should be given as the name of the desired font. But the font name specified must be present in the system, otherwise the default-name is used by the browser. The font will only display if the user has that font installed on the computer. Arial and Comic Sans MS are pretty widely distributed with Microsoft Windows. So if you are using a Microsoft supplied operating system, these are likely to be available on your computer. In Figure 2.7, you see an example of the Arial font being used.

b. SIZE: Font can be displayed in any of the 7 sizes:

Tiny	Small	Regular	Extra Medium	Large	Real Big	Largest
1	2	3	4	5	6	7

c. COLOUR: With this attribute you can change the desired font colour. The values can be specified either by using the hexadecimal format as already described, i.e., #RGB, or by the name of the colour. The hex code of the colour has been explained earlier in this unit. As shown in Figure 2.7, the value of the colour attribute in the third line has been specified as "Blue". So the text present in the code between the <FONT> and </FONT> tags appears in blue. By default the colour of the text is black. If we specify the text colour in the <FONT> tag then this value overrides the colour value, if any, specified in the <BODY> tag.

2.4.5 Special Characters

You have seen that there are certain characters that have special meaning in HTML code. For example, the "< " and " > " characters delimit tags. If you want to display such characters on the web page, you have to take care that the characters are not interpreted and are displayed literally. To display the "< " character, it can be specified as "&lt". The "&" interprets the "lt" as the "< " symbol and displays it. But now what if you want to display the & symbol itself? Simply writing "&" in the code will not display it. But first, let us see how to display some special characters. Consider the example shown in Figure 2.8 and also look at Table 2.1.

```
<HTML>
  <BODY BGCOLOUR="#FFFFFF">
    This is         
    Used for         blank space.
    <BR>
    &lt;  is the Less Than symbol <BR>
    &gt; is the Greater Than symbol <BR>
```

```
    &amp; is the ampersand symbol <BR>
    &quot; is the quotation mark symbol <BR>
    &agrave; is small a, grave accent symbol <BR>
    &Agrave; is capital a, grave accent symbol <BR>
    &ntilde; is small n, tilde symbol <BR>
    &Ntilde; is capital n, tilde symbol  <BR>
    &uuml; is the umlaut small u symbol <BR>
    &Uuml; is the umlaut <BR>
    &#144; is the symbol Delta<BR>
    &#188; is the quarter symbol <BR>
    &#189;is the hay symbol <BR>
</BODY>
</HTML>
```

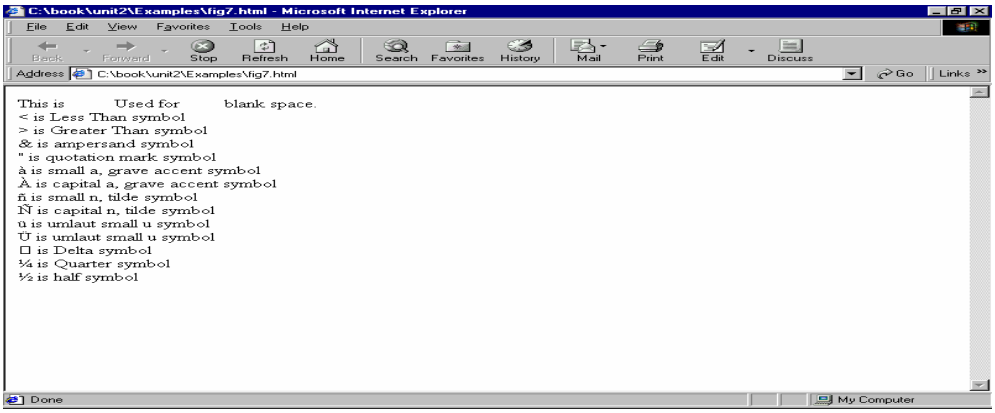


Figure 2.8: Entering Special Characters

The special characters shown in Figure 2.8 are some of the most frequently used characters displayed on web pages. Each of the special characters can be displayed by using its character sequence after the "&". These can be seen in the following Table 2.1.

Table 2.1: Displaying Special Characters

Special Character	Character Symbol	Description
<	<	Less-than symbol
>	>	Greater-than symbol
&	&	Ampersand
"	"	Quotation Mark
		Blank space
à	à	small a, grave accent
À	À	capital A, grave accent
ñ	ñ	small n, tilde
Ñ	Ñ	capital N, tilde
ü	ü	umlaut small u
Ü	Ü	umlaut capital U
□		delta
¼	¼	One Fourth
½	½	Half

The browser will display your text in a steady stream unless you tell it to do so otherwise with line breaks. It will reduce any amount of white space to a single space. If you want more spaces, you must use the space character ( ). If you hit Return (or Enter) while you are typing, the browser will interpret that as a space unless there is already a space there.

Consider another example, which shows how to display multiple blank lines. Code a space character with a line break for each blank line you want.

```
<HTML>
  <HEAD>
    <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY BGCOLOUR="#FFFFFF">
    Welcome to <BR>
     <BR>
     <BR>
     <BR>
     <BR>
     <BR>INDIA
  </BODY>
</HTML>
```

Check Your Progress 1

Describe yourself on a web page and experiment with colours in BGCOLOUR, TEXT, LINK, VLINK, and ALINK. Try out different fonts and sizes and also the other tags you have studied so far, such as the <PRE> tag, as well.

Check Your Progress 2

Add the following information to the web page that you created above.

"Job: Software Engineer
The requirement for the job is that the person should be B.E./M.E/M.C.A. having an aggregate score of 70% or above. The job is project based, so it would be for ½ year only initially. ¼ of the salary would be deducted towards income tax, PF and other statutory deductions."

2.5 WORKING WITH IMAGES

Let us now make our web pages more exciting by putting in images.

You specify an image with the <IMG> (image) tag. Earlier in this unit, displaying the images on a page was explained using the BACKGROUND attribute of the <BODY> tag, which displays the image as the background image. Background images make the page heavy and hence the page takes a considerable amount of time to load. But the <IMG>tag can be used for displaying an image with the desired height and width. Let us look at an example (Figure 2.9).

```
<HTML>
  <HEAD>
    <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY BGCOLOUR="#FFFFFF">
    <IMG SRC="image.gif" WIDTH=130 HEIGHT=101 ALT = "IMAGE IS
    TURNED OFF" ALIGN = "BOTTOM" BORDER = 2> This text is placed at the
    middle of the image.
```

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

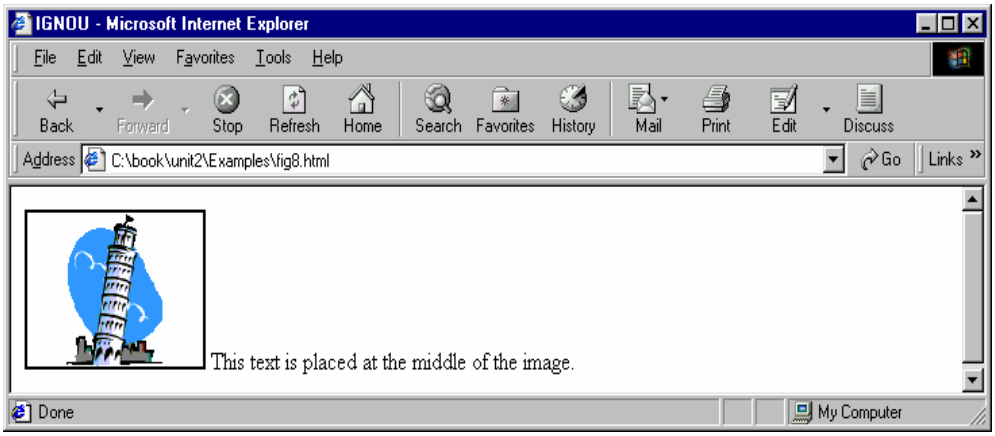


Figure 2.9: Displaying Images on a Web Page

Let us take a look at the syntax of the <IMG> tag:

<IMG SRC = "FILENAME.GIF" WIDTH = "*value*" HEIGHT = "*value*" ALT = "*alternate text*" BORDER = "*value*" ALIGN = "*value*">

a. SRC: This attribute specifies the pathname to the source file that contains the image. The value in the above example, "image.gif", means that the browser will look for the image named image.gif in the same folder (or directory) as the html document itself.

b. WIDTH: This is used for specifying the desired width of the image.

c. HEIGHT: This is used for specifying the desired height of the image.

d. BORDER: An important attribute of the IMG tag is BORDER. This attribute specifies the width of the border of the image. By default it is 0, i.e. there is no border. As shown in Figure 2.9 the image "image.gif" has been given a border 2 pixel wide. The following code gives a wider border to the above image.

```
<BODY BGCOLOUR="#FFFFFF">
    <IMG SRC="image.gif" WIDTH=130 HEIGHT=101 BORDER=10>
</BODY>
```

e. ALT: Another IMG attribute that is important is ALT. ALT is sort of a substitute for the image that is displayed or used when the user is using a browser that does not display images. Someone may be using a text only browser, he may have image loading turned off for speed or he may be using a voice browser (a browser where the web page is read out). In those cases, that ALT attribute could be very important to your visitor as it could be used (given the proper text) to describe the image that is not on the screen.

Check Your Progress 3

Create your own background with a paint program using the following steps:

- Create a small graphic with the paint program.
- Save it as a .jpg or .gif file in the same subdirectory (or folder) that you are keeping the html page that you have been creating.
- Create a simple HTML file with a background, and put the name of your .jpg or

.gif file after the BACKGROUND attribute. (Note: You can easily create the simple html file by copying the html tags above from this web page and pasting them into your new html file. Be sure to substitute the name of your .jpg or .gif file for “image.gif”).

- Save the simple html file that you have just created and open it with your web browser. What do you see? When you are finished, return to this page and continue.

2.6 META TAG

You might be aware of, and perhaps may have used, search engines such as Google to look for web pages on a topic of interest. The META Tag comes in useful if you want your web page to be easily locatable by search engines. When you enter a search string, the search engine shows web pages containing that string, provided the web page has used those in META tag appropriately. The search engine interacts with the META tag of the HTML page in order to find the required string. Information inside a Meta element should be such as to describe the document. Consider the following example (Figure 2.10).

```
<HTML>
  <HEAD>
    <TITLE>IGNOU</TITLE>
    <META NAME="author" CONTENT="IGNOU">
    <META NAME="description " CONTENT="This website shows you the
different courses offered by
     IGNOU">
    <META NAME="keywords " CONTENT=" Website, different courses offered,
IGNOU,mca,bca">
  </HEAD>
  <BODY>
    <P>
    The meta attributes of this document identify the author and courses offered.
    </P>
  </BODY>
</HTML>
```

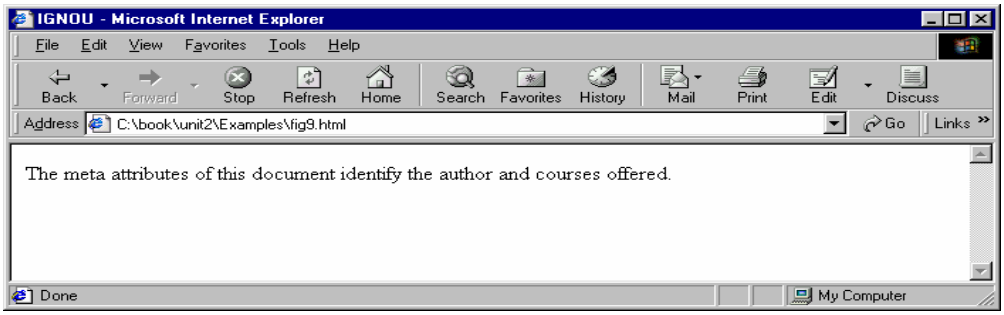


Figure 2.10: Using the META Tag

Meta tags have two attributes:

a. NAME: This attribute is used for identifying the type of META tag you are including.
b. CONTENT: This attribute is used to specify the keywords that the search engine catalogs.

Consider the following code of the example shown in Figure 2.10.

<META NAME= “description” CONTENT="This website shows you the different courses offered by IGNOU">

The CONTENT attribute provides the list of words in the form of a sentence to the search engine. So if someone searches for one of the keywords listed by you in the META tag, then your web site would also appear in the result of the search. It is useful to include META tags that include as many keywords as possible. This makes the web page more likely to show up in a search.

You can also specify keywords by separating them by commas as shown in the following code fragment of Figure 2.10.

<META NAME= “keywords” CONTENT= “Website, different courses offered, IGNOU,mca,bca”>
You can use either of the methods of specifying the META tag as convenient.

Consider another example shown in Figure 2.11. This example demonstrates how to redirect a user if your site address has changed.

```
<HTML>
  <HEAD>
    <TITLE>IGNOU</TITLE>
    <META HTTP-EQUIV="Refresh"
CONTENT="5;URL=http://www.ignou.ac.in">
  </HEAD>
  <BODY>
    <P>
    Sorry! We have moved! The new URL is: www.ignou.ac.in
    </p>
    <p>
    You will be redirected to the new address in five seconds.
    </p>
  </BODY>
</HTML>
```

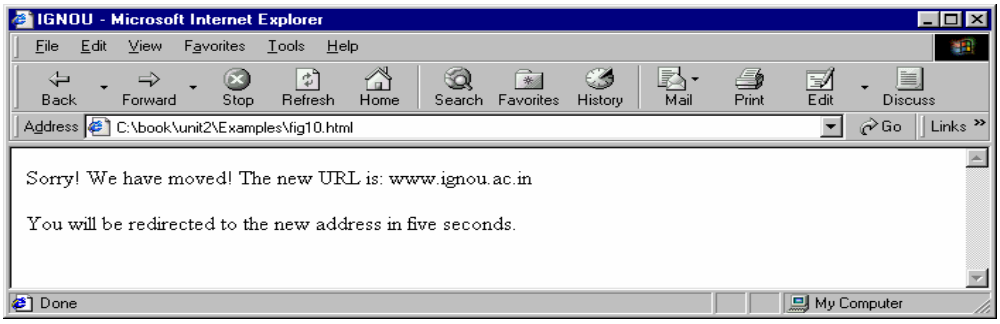


Figure 2.11: Redirecting a User if the Site has Moved

Consider the following code of Figure 2.11

<META HTTP-EQUIV= “Refresh” CONTENT=“5;URL=http://www.ignou.ac.in”>

It indicates to the browser that the page has to be refreshed in 5 seconds with the new URL “http://www.ignou.ac.in”. So when the user sees this page by specifying its original URL,the browser is redirected to the webpage “www.ignou.ac.in” after five seconds. This type of redirection is useful when you want that a user accessing your old website should automatically be redirected to the new website address.

Now let us consider an example that makes use of almost all the tags explained so far.

Case Study: Design a single page web site for a store listing the products and services offered. The store sells computers and related products. The site should contain images explaining the products graphically.

```
<HTML>
<HEAD>
<TITLE> SOLVED CASE STUDY FOR HTML </TITLE>
</HEAD>
<BODY LINK="#0000ff" VLINK="#800080">

<P ALIGN="CENTER"> </P>
<B><I><U><FONT SIZE=5><P ALIGN="CENTER">ABC Products</P> </FONT>
</B></O></I><P>ABC store sells the latest  in computers and computer products.
Besides, we also stock stationery.</P>
<P ALIGN="CENTER"><HR></P>
<B><U><P>Product 1.</P>
</U></B><P><IMG SRC="Image1.gif" WIDTH=127 HEIGHT=102></P>
<P>This is a notebook. It has 200 pages. Each page has three columns with a heading
for date, name and address. Its cost is Rs. 100 only.</P>
<P ALIGN="CENTER"><HR></P>
<B><U><P>Product 2.</P>
</U></B><P><IMG SRC="Image2.gif" WIDTH=127 HEIGHT=102></P>
<P>This is a computer. It has 512 MB RAM with a 2.3 GHz processor and an 80 GB
HDD. Its cost is Rs. 30,000 only. It is pre-loaded with Windows 2003.  You can buy
Microsoft Office software too from us.</P></BODY>
</HTML>
```

Check Your Progress 4

Design a single page web site for a university containing a description of the courses offered. It should also contain some general information about the university such as its history, the campus, its unique features and so on. The site should be coloured and each section should have a different colour.

2.7 SUMMARY

In this unit we have learnt how to create simple HTML pages. The contents of the page have to be written within the BODY tag. The HEAD tag includes the title of the document. An important part of displaying a page is the proper formatting of the text. There exist many tags to do this job. The headers of the sections and sub-sections of the document can be displayed using the header tags (<H1> to <H6>). The <P> tag is used to demarcate a paragraph. The <B>, <I> and <U> tags are used to mark the text as bold, italics and underlined respectively. The <STRONG> and <EM> tags are used to emphasize the text in bold and italics. The <BLOCKQUOTE> tag indents the left margin of the text. The <ADDRESS> tag displays the text in italics. Any text placed between the <CITE> and </CITE> tags, is rendered in italics by the browser. You can display the text exactly as written in the code using the <PRE> tag. The size, colour and the type of the font can be specified using the <FONT> tag. The <IMG> tag is used for inserting images in the document. We have also looked at the very useful <META> tag. This tag is used to redirect the users to other pages, and to provide information about the page.

References: INTERNET & WORLD WIDE WEB BY DEITEL, DEITEL & NIETO
HANDS ON HTML BY GREG ROBERTSON

2.8 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. <HTML>
<HEAD>
</HEAD>
<BODY LINK="#0000ff" VLINK="#ffff99" BGCOLOUR="#FFFFFF">
<B><FONT SIZE=4><FONT FACE="Arial, sans-serif">Ram</FONT></FONT></B></P>
<B><U><FONT SIZE=3><FONT FACE="Arial, sans-serif">Qualification</FONT></FONT></U></B></P>

<FONT FACE="Arial, sans-serif"><B>Title</B></FONT>

<FONT FACE="Arial, sans-serif"><B>Authority</B></FONT>

<FONT FACE="Arial, sans-serif"><B>Division %</B></FONT>

<FONT FACE="Arial, sans-serif"><B>Year </B></FONT>

<FONT FACE="Arial, sans-serif">10</FONT>

<FONT FACE="Arial, sans-serif">CBSE Board</FONT>

<FONT FACE="Arial, sans-serif">60.00%</FONT>

<FONT FACE="Arial, sans-serif">1994</FONT>

<FONT FACE="Arial, sans-serif">12 </FONT>

<FONT FACE="Arial, sans-serif">CBSE Board</FONT>

<FONT FACE="Arial, sans-serif">67.00%</FONT>


```
<FONT FACE="Arial, sans-serif">1996</FONT>
        
<BR><FONT FACE="Arial, sans-serif">B.E.</FONT>
           &
nbsp;   

<FONT FACE="Arial, sans-serif">Delhi University</FONT>
           &
nbsp;   

<FONT FACE="Courier, monospace">I</FONT><FONT FACE="Arial, sans-
serif">
Division</FONT>
           &
nbsp;         &n
bsp;      
<FONT FACE="Arial, sans-serif">1999</FONT>
<BR><BR><B><U><FONT SIZE=3><FONT FACE="Arial, sans-
serif">Personal
Details</FONT></FONT></U></B>
<PRE>

<FONT FACE="Arial, sans-serif">
Father's Name: Mr. Shyam mehta
Date of Birth: 23 Jan, 1976. </FONT>
<FONT FACE="Arial, sans-serif">
Address: A2-81b, East Of kailash,New Delhi.
Tel: 29090909,9220101010</FONT>

<FONT FACE="Arial, sans-serif">
Email: ram@hotmail.com
Sex: Male
Marital Status: Single
Interests and activities: Troubleshooting hardware and software problems.
</FONT>
</PRE>
</BODY>
</HTML>
```

Check Your Progress 2

2.
```
<HTML>
<HEAD>
</HEAD>
<BODY LINK="#ffff00" VLINK="#ffff99" BGCOLOUR="#ffffff">
<PRE>
<U><B><FONT FACE="Times New Roman, serif">Job: Software Engineer
</FONT></B></U>
<FONT FACE="Times New Roman, serif">

The requirement for the job is that the person should be B.E./M.E/M.C.A having
an aggregate score of 70% or above. The job is project based, so it would be for
½ year only initially. ¼ of the salary would be deducted towards income tax, PF
and other statutory deductions.

RE>
<FONT SIZE=4><FONT FACE="Arial, sans-
```

Comment: Please correct the solution to reflect the pre-formatted text as amended in the question.

```
serif">Ram</FONT></FONT></B></P>
<B><U><FONT SIZE=3><FONT FACE="Arial,
sans-serif">Qualification</FONT></FONT></U></B></P>

<FONT FACE="Arial, sans-serif"><B>Title</B></FONT>
           
           
  

<FONT FACE="Arial, sans-serif"><B>Authority</B></FONT>
            
     

<FONT FACE="Arial, sans-serif"><B>Division
%</B></FONT>
            
            
  

<FONT FACE="Arial, sans-serif"><B>Year </B></FONT>
            
     
<BR>

<FONT FACE="Arial, sans-serif">10</FONT>
            
       

<FONT FACE="Arial, sans-serif">CBSE Board</FONT>
            
           

<FONT FACE="Arial, sans-serif">60.00%</FONT>
            
            
        

<FONT FACE="Arial, sans-serif">1994</FONT>
         
<BR><FONT FACE="Arial, sans-serif">12 </FONT>
            
       

<FONT FACE="Arial, sans-serif">CBSE Board</FONT>
            
            

<FONT FACE="Arial, sans-serif">67.00%</FONT>
            
            
        

<FONT FACE="Arial, sans-serif">1996</FONT>
         
<BR><FONT FACE="Arial, sans-serif">B.E.</FONT>
           &
nbsp;   

<FONT FACE="Arial, sans-serif">Delhi University</FONT>
```

```
            &
nbsp;   

<FONT FACE="Courier, monospace">I</FONT><FONT FACE="Arial, sans-
serif">
Division</FONT>
            &
nbsp;           &n
bsp;      

<FONT FACE="Arial, sans-serif">1999</FONT>
<BR><BR><B><U><FONT SIZE=3><FONT FACE="Arial, sans-
serif">Personal
Details</FONT></FONT></U></B>
<PRE>
<FONT FACE="Arial, sans-serif">
Father's Name   : Mr. Shyam mehta
Date of Birth   : 23 Jan,1976. </FONT>

<FONT FACE="Arial, sans-serif">
Address: A2-81b, East Of kailash,New Delhi.
Tel.: 29090909,9220101010</FONT>

<FONT FACE="Arial, sans-serif">
Email: ram@hotmail.com
Sex   : Male
Marital Status   : Single
Interests and activities   : Troubleshooting hardware and software problems.
</FONT>
</PRE>
</FONT>
    </BODY>
</HTML>
```

Check Your Progress 3

1.
```
<HTML>
<HEAD>
<TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOUR="#FFFFFF">
<IMG SRC="abc.jpg" WIDTH=130 HEIGHT=101 ALT = " IMAGE IS
TURNED OFF" ALIGN = "MIDDLE" BORDER = 2> This text is placed at the
middle of the image.
</BODY>
</HTML>
```

Comment: This solution does not reflect an answer to the question. There the image was to form the page background, for instance. Please amend the solution.

Check Your Progress 4

1.
```
<HTML>
<HEAD>
</HEAD>
<BODY LINK="#0000ff" VLINK="#800080" BGCOLOUR="#ff6600">

<P ALIGN="CENTER"> </P>
<B><I><U><FONT SIZE=4><P ALIGN="CENTER">IGNOU</P>
</I></U></FONT><FONT SIZE=2><P>I</B>ndira <B>G</B>andhi
<B>N</B>ational <B>O</B>pen University is a very old and reputed
```

University. IGNOU offers various types of courses that are both academic and technical. </P>
<P> </P>

</FONT><FONT SIZE=4 COLOUR="#0000ff"><P>Master in Computer Applications</P>

</FONT><FONT COLOUR="#0000ff"><P>This course has 6 semesters and the total duration is 3 years. The maximum duration allowed for completing the course is 7 years. The fee per semester is Rs 5000 </FONT> </P>

<FONT SIZE=4 COLOUR="#00ff00"><P>Bachelor in Computer Applications</P>

</FONT><FONT COLOUR="#00ff00"><P>This course has 6 semesters and the total duration is 3 years. The maximum duration allowed for completing the course is 6 years. The fee per semester is Rs 3000</P>

</FONT><FONT SIZE=4 COLOUR="#ff0000"><P>Bachelor in Information Technology</P>

</FONT><FONT COLOUR="#ff0000"><P>This course has 6 semesters and the total duration is 3 years. The maximum duration allowed for completing the course is 5 years. The fee per semester is Rs 10,000.</P></FONT></BODY>
</HTML>

Page 40: [1] Comment **milind**

Please correct this to show a small “a” with a grave accent.

Page 40: [2] Comment **milind**

Please correct this to show a small n with a tilde.

Page 40: [3] Comment **milind**

Please omit the delta symbol if it is not displaying correctly.

UNIT 3 ADVANCED HTML

Structure **Page No.**

3.0 INTRODUCTION

In the previous unit you have learned the basics of HTML. After learning about how to make static web pages, let us now learn how to develop Interactive Web sites. A good web site should be interactive and easy to use and understand. Of course, very simple web sites that merely need to present some static information may not provide for user input. Interactive web sites are those that are capable of taking input from the user and presenting the output on the basis of the inputs given. To be able to do so, you will need more HTML features than have been covered so far. Features like Links, Lists, Tables, Input controls will allow you to create sophisticated web pages that respond dynamically to user input. HTML provides all these features using different tags such as A, UL, OL, INPUT, FRAMESET and others, that you will study in this unit. Besides the tags themselves you will also learn about their common attributes.

3.1 OBJECTIVES

This unit will enable you to create sophisticated, interactive web pages. After going through this unit you will be able to learn:

- links using ANCHOR tag;
- ordered, unordered and definition Lists;
- tables;
- frames to divide a web page into different parts; and
- forms for accepting user input.

3.2 LINKS

Hyperlinks, or links are one of the most important characteristics of web pages. A link moves us from the current page to a destination that is specified in the HTML page.

URL Stands for Universal Resource Locator. A URL is just an address that tells the browser precisely where on the Internet the resource is to be found. The process of parsing the URL and actually connecting to the resource can be somewhat complex and does not concern us here.

3.2.1 Anchor Tag

The Anchor tag is used to create links between different objects like HTML pages, files, web sites etc. It is introduced by the characters <A> and terminated by </A>. HREF is the most common attribute of the ANCHOR tag. It defines the destination of the link.

```
<HTML>
  <HEAD>
     <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF">
     Go to <A HREF="http://www.ignou.ac.in/">IGNOU!</A>
  </BODY>
</HTML>
```

As shown in Figure 3.1, the text “IGNOU” present between the <A> and </A> tags becomes the hyperlink. On clicking anywhere on this hyperlink, the browser would attempt to connect to the given URL and the website http://www.ignou.ac.in would open, if possible. An email link can be specified in the same way. We just have to specify the email address instead of a page address as the value of HREF as shown in the following code. On clicking on the link, the default mail program on the user’s computer opens up with a “To:” address as specified in the hyperlink. You can then type your message and send e-mail to that address.

```
<BODY BGCOLOR="#FFFFFF">
Send me <A HREF="mailto:forrest@bubbagump.com">mail</A>
</BODY>
```

It is also possible to make an image a link if desired. This is done using the <IMG> tag. Consider the following example.

```
<HTML>
  <HEAD>
     <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF">
```

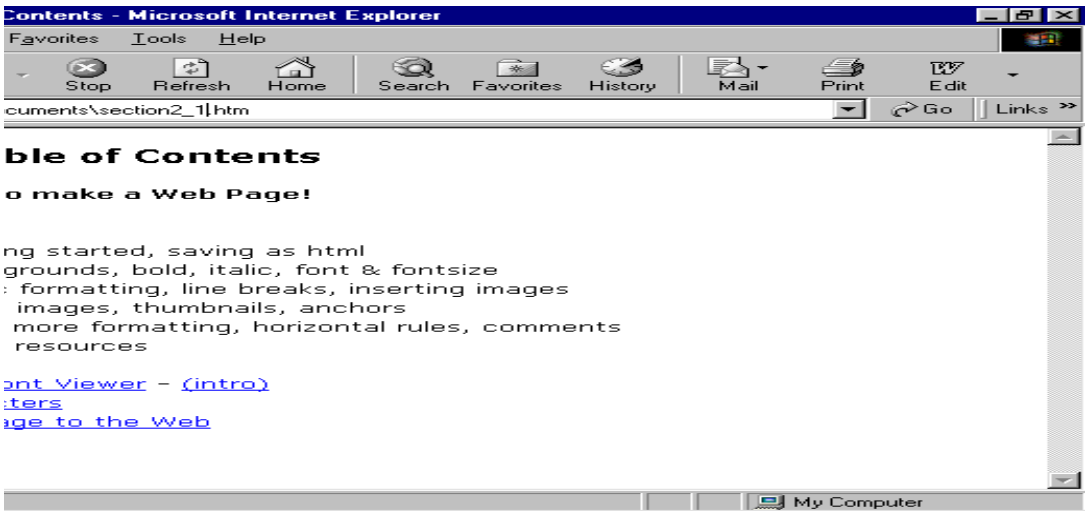
```
    Go to <A HREF="http://ignou.ac.in/"><IMG SRC="image.gif" WIDTH=130
HEIGHT=101></A>
  </BODY>
</HTML>
```

So in the example shown in Figure 3.2, the image becomes the link. When the user clicks on the image, the web site http://www.ignou.ac.in opens up, if possible.

Check Your Progress 1

1. Create a web page that provides links to five different web pages or to entirely different web sites.

2. Write HTML code for the following page (the underlined text is linked to another file called link_text)



3.3 LISTS

Lists are used when the data are to be mentioned in the form of points like: causes of a particular issue, list of items etc. Lists break up the monotony of a long paragraph and direct the reader's attention to the essential parts.

Lists are segregated into three types, namely Ordered lists, Unordered lists and Definition lists.

3.3.1 Unordered Lists

First, we will build an unordered list. Sometimes, these lists are also called bulleted lists. These lists are characterized by list items that do not have numbers. They are used when the points in the list have no particular order. They are delimited by the <UL> and </UL> tags. Each point in the list is delimited by the <LI> and </LI> tags.

```
<HTML>
  <HEAD>
    <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF">
    What I want for Id
    <UL>
      <LI>a big red truck</LI>
```

Comment: Better give an example that an Indian reader will relate to.

```
        <LI>an aeroplane that flies</LI>
        <LI>  a nice soft toy</LI>
        <LI>a drum set</LI>
        <LI>a  Walkman</LI>
        <LI>  extra pocket money</LI>
      </UL>
  </BODY>
</HTML>
```

The syntax of the <UL> tag is:
<UL TYPE=""> where TYPE= "DISC", "SQUARE" or "CIRCLE".
<LI> </LI>
</UL>

The bullet appearance can be changed to be round (a dark circle), a disc or a circle depending on the value of the TYPE attribute. As shown in the Figure 3.3, the list of items are included within the <UL> and </UL> tags. Each list item must be preceded with a <LI> tag.

3.3.2 Ordered Lists

Lists having numbered items are known as ordered lists. They are used when the items in the list have a natural order. They can also be used when the number of items in the list needs to be known at a glance. To make an **ordered** list, simply change the <UL> tag to <OL>.

```
<HTML>
  <HEAD>
     <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF">
     What I want for my birthday
     <OL>
       <LI>a big red truck</LI>
       <LI>a toy train</LI>
       <LI>a Barbie doll</LI>
       <LI>a drum set</LI>
       <LI>binoculars</LI>
       <LI>a month's extra pocket money</LI>
     </OL>
  </BODY>
</HTML>
```

The syntax of the <OL> tag is:
<OL TYPE="" START=""> where TYPE= "1", "a", "A", "i", "I" and START is the first item number.
<LI> </LI>
</OL>

The type of the numbering format desired should be specified as the value of the TYPE attribute. The possible types are shown above in the syntax. The numbering starts from the START attribute's value.

3.3.3 Definition Lists

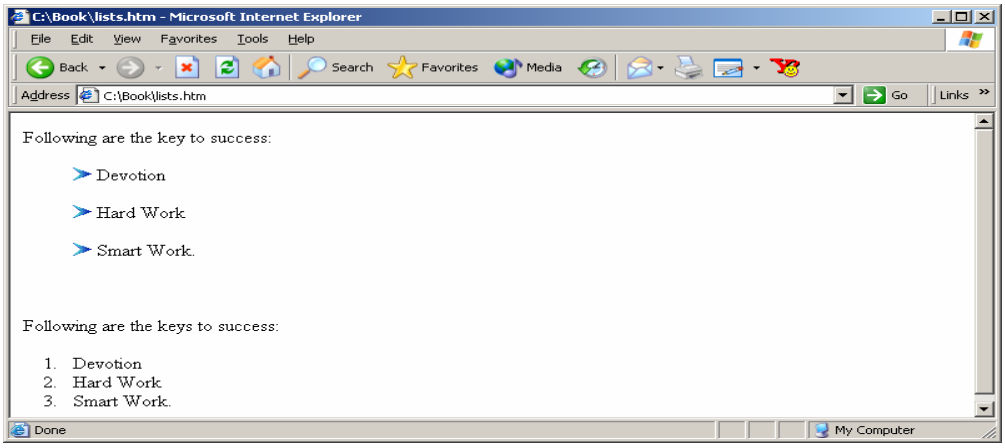
Another type of list is a **definition** list. Definition lists have a heading and the text appears below that.

```
<HTML>
  <HEAD>
     <TITLE>IGNOU</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF">
    <DL>
      <DT>10th Amendment </DT>
        <DD>The powers not delegated to the United States by the Constitution, nor
        prohibited by it to the States, are reserved to the States respectively, or to the
        people.
        </DD>
    </DL>
  </BODY>
</HTML>
```

A definition list is introduced by the <DL> tag and terminated by </DL>. The Definition heading should be specified between the <DT> and </DT> tags. The Definition should be specified between <DD> and </DD> tags.

Check Your Progress 2

1. Design a web page that looks similar to following page:



3.4 TABLES

In this section you will see how to put tables in your web documents. It is not that a table is simply a combination of rows and columns. If you have ever seen any table in an attractive web page you might be interested to learn how they make good use of the **<TABLE>** and related tags.

3.4.1 Table, TR and TD Tags

Three tags form the essential ingredients for creating a table.

TABLE: This is the main tag. It tells the browser that a table follows. It has attributes like size and border width.

TR: A TableRow defines a horizontal row that consists of **TableData** *cells*.

TD: This tag specifies an individual block or *cell* in a table row.

Thus a *table* is made up of *rows*, which in turn are made up of *cells*.

<--This---	**----is----**	**----a----**	**---Table---**	**---Row-->**
			Cell	

You are now ready to create some tables! We must stress that if you want to learn how to make quality HTML documents, then you would be spending your time well if you teach yourself the tags. In our opinion the best HTML editors to use for beginners are text-based editors. These editors will force you to code HTML yourself. They don't attempt to do it for you. Once you are an expert, you can use other tools to improve your productivity so that you do not have to handcode your pages.

Now let us start creating tables. The following example shows some tables with different attributes.

```
<HTML>
  <HEAD>
    <TITLE> IGNOU</TITLE>
  </HEAD>
  <BODY>
    <!- - Table with border = 5 - - >
    First: Table with border = 5
    <TABLE BORDER=5>
      <TR>
        <TD>Ajay</TD>
      </TR>
    </TABLE>
    <!- - Table with width = 50%- - >
    Second: Table with width = 50%
    <TABLE BORDER=3 WIDTH=50%>
      <TR>
        <TD>Ajay</TD>
      </TR>
    </TABLE>
    <!- - Table with width = 50 - - >
    Third: Table with width = 50
    <TABLE BORDER=1 WIDTH=50>
      <TR>
        <TD>Ajay</TD>
      </TR>
    </TABLE>
    <!- - Table with height = 75 and align = center and valign=top - - >
    Fourth: Table with height = 75 and align = center and valign=top -
    <TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
        <TD ALIGN=CENTER>Ajay</TD>
      </TR>
    </TABLE>
    <!- - Table with an image- - >
    Fifth: Table with an image
    <TABLE BORDER=3>
      <TR>
        <TD ALIGN=LEFT VALIGN=MIDDLE><IMG SRC="image1.gif"
WIDTH=32
```

```
        HEIGHT=32></TD>
      </TR>
    </TABLE>
    <!- - A complete Table with different sized columns- - >
    Sixth: A complete Table with different sized columns
    <TABLE BORDER=3 WIDTH=300 HEIGHT=75>
      <TR>
        <TD WIDTH=60%>Ajay</TD>
        <TD WIDTH=20%>Ramesh</TD>
        <TD WIDTH=20%>Vijay</TD>
      </TR>
      <TR>
        <TD>Pankaj</TD>
        <TD>Vikas</TD>
        <TD>Rohan</TD>
      </TR>
    </TABLE>
</BODY>
</HTML>
```

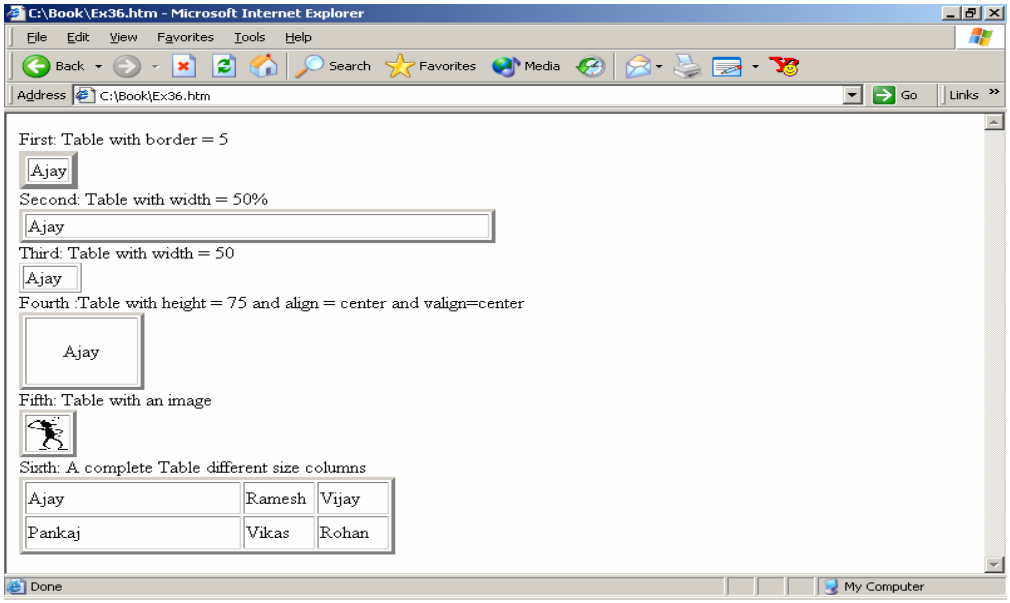


Figure 3.1: Tables with Different Attributes.

Comment: This output cannot come from the code given earlier. From where are "First:", "Second:" etc coming?

In the first table the BORDER attribute is given a value of 5. The default is no border i.e. border = 0.

When no sizes are specified, the table is only as big as it needs to be, as shown in the first and third table in Figure 3.6. Specifying a table size is easy. Let us reduce the table size to 50% of the browser window as has been done in the second example. See the output in the second table in Figure 3.6. As you can see in the example there are two ways to specify the table width. Each style has its uses.
You can also change the *height* of a table. The fourth table in Figure 3.6 shows the effect of the changed height.

You can control where in the cell the data will appear. For this purpose we use the ALIGN attribute. In the fourth table in Figure 3.6, we have used a *center* alignment. Similarly, *right* and *left* can be used for right and left alignment respectively. The default value is **ALIGN=*left***. This is the value that the browser assumes if you have not told it otherwise.

You can also control where data will appear vertically in a cell. For this purpose you specify the VALIGN attribute. In the fourth table, we have used the value *center*. You can also use *top* or *bottom*. Images can also be placed and manipulated in a table data cell. In the folder that contains the document with the HTML code, substitute an **IMG** tag for text**.** This is shown in the fifth table. You can also include size attributes with all your image tags. We will not go into the details here, but doing so makes it easier for the browser to display the table and avoids any nasty little surprises while resizing the browser window, for instance.

Now, let us look at multiple rows. Suppose three more friends from across the street see what is going on and want to be in your table. We think we will give them their own row. Each row can be assigned a different width for its columns. In the sixth table we have used 60/20/20 as the relative percentage widths of the three columns. The last table shows how to create a table with multiple rows and columns. The **WIDTH** attributes in the first row carry over to the second row.

3.4.2 Cell Spacing and Cell Padding

Next let us look at a couple of attributes called **CELLPADDING** and **CELLSPACING**. Both are part of the **<TABLE>** tag. **CELLPADDING** is the amount of space between the border of the cell and the contents of the cell. The default value for this attribute is 1. This is so that any text in the cells does not appear to be sticking to the border. However, you can specify a value of 0 if you wish.

The **CELLSPACING** attribute has a somewhat different meaning. It determines the spacing between adjacent cells. Its default value is 2. The following example shows these and some other important attributes of the <TABLE> tag.

```
<HTML>
  <HEAD>
    <TITLE> IGNOU</TITLE>
  </HEAD>
  <BODY>
  First: Table with cellspacing and cellpadding
  <TABLE BORDER=3 CELLSPACING=7 CELLPADDING=7>
    <TR>
      <TD>Ajay</TD>
      <TD>Vijay</TD>
      <TD>Rohan</TD>
    </TR>
    <TR>
      <TD>Pankaj</TD>
      <TD>Vikas</TD>
      <TD>Sanjay</TD>
    </TR>
  </TABLE>
  Second: Table with colspan
  <TABLE BORDER=1>
    <TR>
      <TD COLSPAN=2>Ajay</TD>
      <TD>Vijay</TD>
    </TR>
    <TR>
      <TD>Vikas</TD>
      <TD>Pankaj</TD>
      <TD>Rohan</TD>
    </TR>
  </TABLE>
```

```
Third: Table with rowspan
<TABLE BORDER=1>
  <TR>
    <TD ROWSPAN=2>Ajay</TD>
    <TD>Vijay</TD>
    <TD>Rohan</TD>
  </TR>
  <TR>
    <TD>Pankaj</TD>
    <TD>Deepak</TD>
  </TR>
</TABLE>
Fourth :Table with rowspan and colspan
<TABLE BORDER=1>
  <TR>
    <TD ROWSPAN=2>Ajay</TD>
    <TD COLSPAN=2>Vijay</TD>
  </TR>
  <TR>
    <TD>Pankaj</TD>
    <TD>Rohan</TD>
  </TR>
</TABLE>
Fifth: Table with strong tag and a link in cell's data
<TABLE BORDER=1>
  <TR>
    <TD COLSPAN=3
    ALIGN=CENTER><STRONG><AHREF="http://IGNOU.org/">Ajay</STRO
NG>
    </TD>
  </TR>
  <TR>
    <TD>Vijay</TD>
    <TD>Vikas</TD>
    <TD>Pankaj</TD>
  </TR>
</TABLE>
</BODY>
</HTML>
```

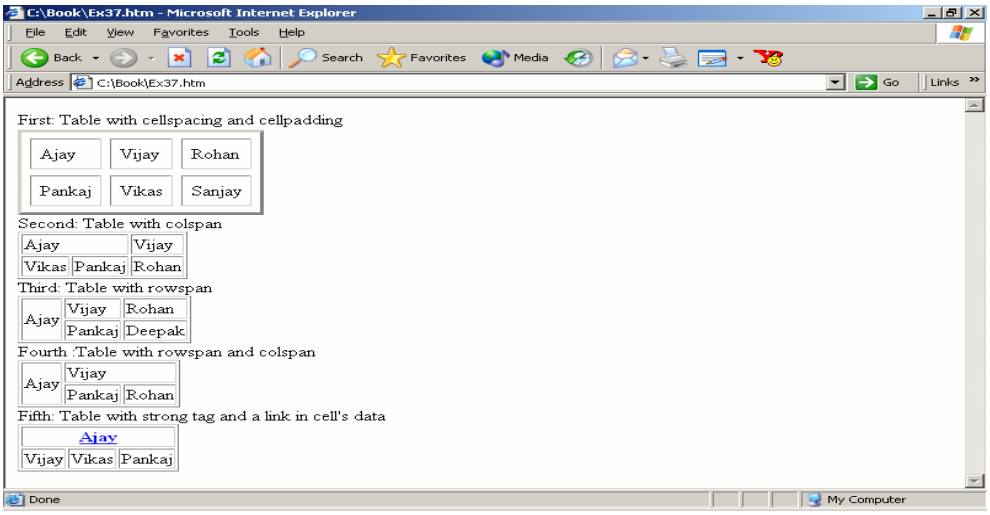


Figure 3.2: Some Important Attributes of the Table Tag Family

The first table in Figure 3.7 illustrates cellspacing and cellpadding.

3.4.3 Colspan and Rowspan

Now let us see how to work with **COLSPAN** (Column Span) and **ROWSPAN** (Row Span).

If we want the cell containing Ajay in Figure 3.7 to be extended to the next cell as well and make that area part of it own, we can use the **COLSPAN** attribute. This amounts to merging the two cells into one. Those of you who are familiar with spreadsheets such as Microsoft Excel will recognize this as similar to its "merge cells" feature. As you may have guessed, **<ROWSPAN>** is just like **<COLSPAN>**. The second table in Figure 3.7 shows COLSPAN, the third table shows ROWSPAN and the fourth table shows an example featuring both COLSPAN and ROWSPAN.

Check Your Progress 3

1. Design a web page that has 5 equal columns. The table should look the same in all screen resolutions.

2. Make out a brief bio-data of yours and code it as an HTML Page. You can consider using tables to show your academic history.

3.5 FRAMES

Now let us understand how to make frames for web documents. The intelligent use of frames can give your pages a cleaner look and make them easier to navigate.

The basic concept goes like this: Each frame is a regular, complete HTML document. If you wanted to lay out your page into two frames placed side by side, then you would put one complete HTML document in the left frame and another complete HTML document in the right frame. In addition you need to write a *third* HTML document. This *MASTER PAGE* contains the **<FRAME>** tags that specify what goes where. Dividing a page into frames is actually quite simple.
There are only two major frame tags that you need to learn: **<FRAMESET>** and **<FRAME>**. The easiest way to explain them is to start making some frames. As an aspiring computer professional, you would be well advised to learn the HTML tags in detail. Do not use HTML editors unless you know HTML, just as you would not let a child use a calculator unless she already knows her arithmetic well. Once you have a thorough grasp of HTML, you can then save labour by using different tools. The tools might not offer the flexibility that you can get through handcoding HTML. So they should be used judiciously. Of course, this advice is not meant for lay persons who want to make web pages.

We will now need a few HTML documents. We will give each document a name. Create a new folder somewhere and create the first HTML file as **One.htm**. Now make another html document. Save this in the same folder as **Two.htm**. Now do the same for Three, Four, Five, and Six. Save them just like the others. You should now have a folder that contains 6 complete standalone HTML documents.

3.5.1 Frameset

Now make a master page in which you write the following code.

```
<HTML>
<HEAD>
<TITLE>My Frame Page -- The Master Page</TITLE>
</HEAD>
```

```
<FRAMESET>
</FRAMESET>
</HTML>
```

Now, save it in your folder (with all the other files) as **index.htm**. If you try to open it with your browser now it will be blank. All you have done so far is make a title.

Now let us start defining just how things are going to look. Tell the browser to split the main window into 2 columns, each occupying 50% of the window.

```
<FRAMESET COLS="50%,50%">
</FRAMESET>
```

This will still give a blank output as you have not specified what goes into the windows. We have one more thing to do before our code displays some output.

3.5.2 Frame Tag

As you can guess, this tag is used for placing an HTML file in the frame created. We must now tell the browser what to put in each frame.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50%,50%">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
</FRAMESET>
</HTML>
```

You also need to note here that **<FRAMESET>** is a *container* tag, and **<FRAME>** is not. A container tag has an opening **<TAG>** and a closing **</TAG>**. So notice that the <FRAME> tag has no delimiter to terminate it. Everything is in its attributes.

The **<FRAMESET>** tag does all the dividing of the page into different windows. It also has attributes that specify *how* to divide them up. Can we divide the page into more than 2 pieces? Yes, just make sure that you specify a page to occupy each section or the browser will get confused. Look at the code and the output in Figure 3.10.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="20%,20%,20%,20%,20%">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
<FRAME SRC="Five.htm">
</FRAMESET>
</HTML>
```

Figure 3.3: A Web Page with Five Frames

It is only a small step to making the frames all of different sizes. Just make sure your arithmetic is correct and that the percentages you specify add up to 100, or the browser will come up with its own interpretation.

If we divide the page into **ROWS** instead of **COLS** we get a different layout.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET ROWS="10%,20%,30%,15%,25%">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
<FRAME SRC="Five.htm">
</FRAMESET>
</HTML>
```

Let us now take another example with only 2 frames. We can specify 50 to indicate that number of pixels instead of 50%. We can also use * instead of a number. The * means whatever is left over.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50,*">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
</FRAMESET>
</HTML>
```

When you use frames you have to be very careful to code properly to ensure that all viewers are able to look at reasonably consistent views. Let us suppose that you make a frame 100 pixels wide on the left and 100 pixels wide on the right. If some users are running an 800 × 600 screen they see the middle area as 600 pixels wide. Other users may have a screen set at 640 × 480. What do they see? The middle area for them is only 440 pixels wide. So if you use any absolute dimensions in your **<FRAMESET>** tags you should have at least one * that will produce an elastic frame. That way everything will look at least reasonably good. If you do not do that, your page might need to scroll on one resolution and not on another. As far as possible you might want to avoid absolute values in your frames and work on relative numbers so that things get taken care of automatically by the browser.

We can have more than one leftover frame and specify a size relationship between them. Try it yourself and see the result.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
```

```
<FRAMESET COLS="50,*,2*">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAME SRC="Three.htm">
</FRAMESET>
</HTML>
```

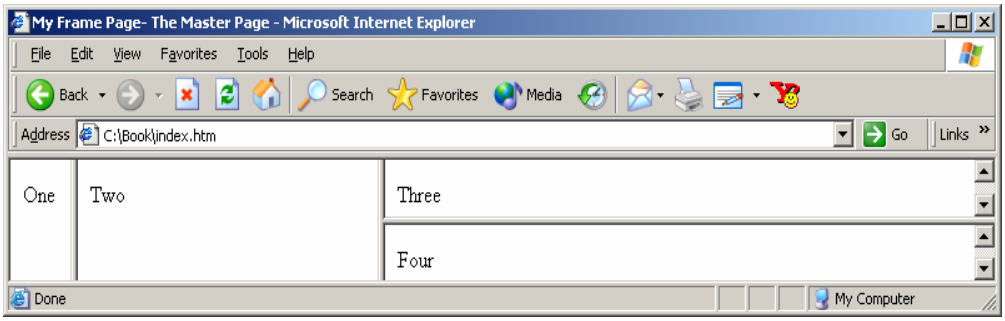
The above code means: Make 3 frames. Make the first 50 pixels wide. Divide the rest between frames 2 and 3. But make frame 3 twice as big as frame 2. Put One.html/ in the first frame, Two.html/ in the second and Three.html/ in the third.

It is important to note that everything is done in order. The first **<FRAME>** is displayed according to the first size attribute in the **<FRAMESET>** tag (**50/One**), the second frame with the second (***/Two**) and the third frame with the third attribute set (**2*/Three**).

Frames inside other frames

Here we will discuss how to divide frames into different frames i.e. how to put horizontal frames in a vertical one and vice-versa. Here we are going to divide a frame in half horizontally.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50,*,2*">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAMESET ROWS="50%,50%">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
</FRAMESET>
</FRAMESET>
```



Here the frame three is at the top and Four at the bottom. This has been done just to illustrate how to achieve the effect and is not meant to suggest that you actually divide up your pages like this. Too many frames on a page do not look good. A good rule of thumb is not to have more than 3 frames on your page. If you can, avoid frames altogether.

3.5.3 Noframes Tag

The <NOFRAMES> tag can be used for those browsers that are not able to interpret <FRAME> tags. Although most, if not all, of your visitors will be able to see frames, there is still a small number of such browsers and there still are many users around

who do not have the latest in equipment. To address as wide an audience as possible, you could write a no-frames version of your page.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50,*,2*">
<FRAMESET ROWS="50,*,*">
<FRAME SRC="One.htm">
<FRAME SRC="Five.htm">
<FRAME SRC="Six.htm">
</FRAMESET>
<FRAME SRC="Two.htm">
<FRAMESET ROWS="50%,50%">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
</FRAMESET>
</FRAMESET>
<BODY><NOFRAMES> your browser does not handle frames! </NOFRAMES>
</BODY>
</HTML>
```

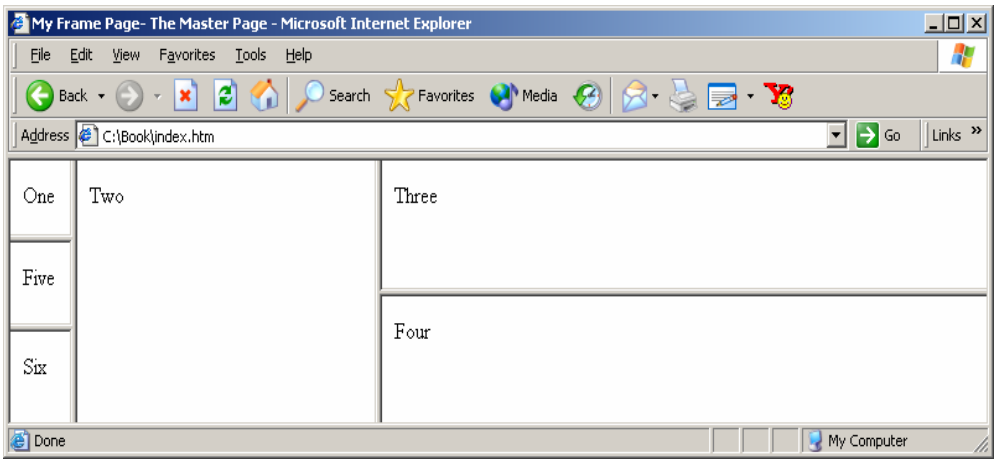


Figure 3.4: Putting in a NOFRAMES Version

Put your non-frames page down between the <NOFRAMES> tags. If someone is using an old browser, it will skip everything above and come straight down here. Frames-capable browsers will ignore what is between the <NOFRAMES> tags.

We can also put images in the frames. Let us see how to put in **"world.gif"** as an example.

```
<FRAME SRC="world.gif" WIDTH=146 HEIGHT=162>
```

The scrollbars that you see can be specified as **YES**, **NO** or **AUTO**. **YES** means the window gets scrollbars - whether they are needed or not. **NO** means there will be no scrollbars. **AUTO** is the default. If scrollbars are needed, they appear while if they are not needed they stay conveniently out of the way. So let us see how to get rid of our scrollbars.

```
<FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO>
```

You would notice a problem with this code. The image is not in the right frame. The next two attributes deal with *margins*. The browser automatically gives each frame some empty space around its contents. This is normally useful for aesthetic purposes.

You can control the size of these margins by using **MARGINWIDTH** and **MARGINHEIGHT**. They control the left and right and top and bottom margins respectively. We will set them both to 1. (1 is the minimum)

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="146,*">
<FRAMESET ROWS="162,*">
<FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO
MARGINWIDTH=1 MARGINHEIGHT=1>
<FRAME SRC="One.htm">
</FRAMESET>
<FRAME SRC="Two.htm">
</FRAMESET>
```

Comment: Please check out all the code for FRAMES by running it exactly as it appears.

3.6 FORMS

Now let us get a grip on how to add interactivity to your web documents by way of the **<FORM>** tag. With this tag you can add to your web pages a guestbook, order forms, surveys, get feedback or put in any other form that you wish.

3.6.1 FORM and INPUT Tag

A good way to learn about forms is to use your notepad editor and create a new HTML document. Save it as **form1.htm** in some folder somewhere. You might want to create a separate folder for learning this tag. Start up your browser. Use it to open **form1.html** and run Notepad and the browser side by side. This way you can create your pages and almost instantaneously see the results of your work. Remember to hit the refresh button on your browser.

Next we must tell the browser where to send the data we gather and how to send it. There are two ways you can do this.

1) You can send the data to a cgi script or use some other mechanism for processing, or

2) You can have the data emailed to you. The first option is outside the scope of discussion of this unit.
The second, or *mailto*, form should have the following attributes in the **<FORM>** tag.

Note: Microsoft's Internet Explorer 3.0 does not support *mailto* forms. When you try to submit the information, the new mail message window pops up. It does support forms sent to a CGI script.

```
<HTML>
<HEAD>
<TITLE> Welcome to IGNOU </TITLE>
</HEAD>
<BODY>
<FORM METHOD=POST ACTION="mailto:xxx@xxx.xxx"
ENCTYPE="application/x-www-form-urlencoded">
</FORM>
```

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

The line containing the *mailto* keyword in the Figure 3.17 is very important. The only thing you have to do is specify your email address after *mailto:* The rest must be written exactly as shown. The words FORM**,** METHOD**,** POST and ACTION do not have to be capitalized but there must be a space between every two attributes, between FORM and METHOD**,** between POST and ACTION, and between the e-mail address and ENCTYPE.

Some mail programs are capable of converting the data without needing a separate program. You may want to try this method first. Just remove the instruction

ENCTYPE="application/x-www-form-urlencoded" and in its place use **ENCTYPE="text/plain"**.

The following example shows a general form that includes some of the commonly used controls.

```
<HTML>
<HEAD>
<TITLE> Welcome to IGNOU </TITLE>
</HEAD>
<BODY>
<FORM METHOD=POST>

Text box with name, value and size attributes
<INPUT TYPE=TEXT NAME="ADDRESS" VALUE="44 XYZ" SIZE=10>

<BR>Text box with password and maxlength
<INPUT TYPE=PASSWORD MAXLENGTH=10>

<BR>Radio button with default checked<BR>
Who is your best friend?<BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="Ajay" CHECKED>
Ajay <BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="Rohan"> Rohan <BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="Tarun"> Tarun<P>

<BR>Checkbox<BR>
<INPUT TYPE=CHECKBOX NAME="AJAY" VALUE="YES"> Ajay <BR>
<INPUT TYPE=CHECKBOX NAME="Rohan" VALUE="YES"> Rohan <BR>
<INPUT TYPE=CHECKBOX NAME="BU" VALUE="YES"> Bunty<P>

<BR>Drop down list<BR>
<SELECT NAME="BEST FRIEND" SIZE=4>
<OPTION VALUE="Ajay">Ajay
<OPTION VALUE="Rohan">Rohan
<OPTION VALUE="Tarun">Tarun
<OPTION VALUE="Gagan">Gagan
<OPTION VALUE="Harish">Harish
<OPTION VALUE="Manjit">Manjit

</SELECT>

</FORM>
```

</HTML>

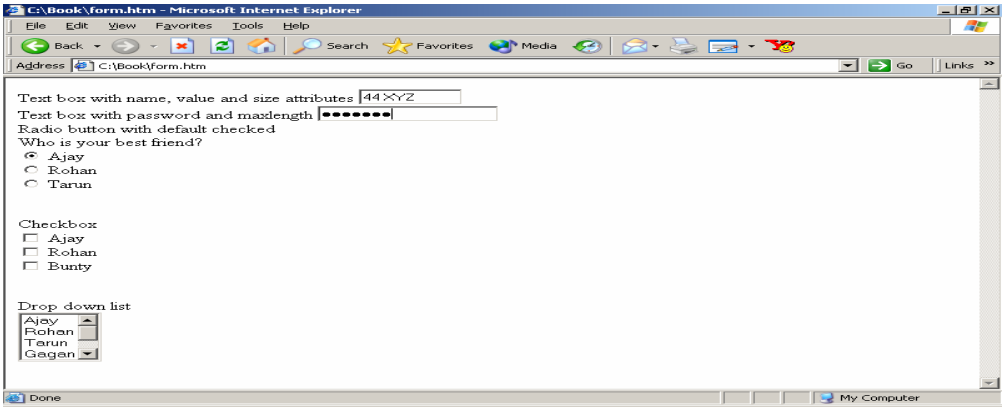


Figure 3.5: A Web Page with a Form

3.6.2 Text Box

The <INPUT> tag is used to indicate where user input is expected. It has different attributes, of which the TYPE attribute is used to specify the kind of input that is to be provided. The most common value of this attribute of the <INPUT> tag is TEXT. As shown in Figure 3.18, every INPUT needs a NAME. When the user types in his address (for example 1234 ABC), it will become the input's *value* and be paired with NAME so the end result after running it through the *Mailto* Formatter will be ADDRESS=1234 ABC.

We can, if we want, type in a VALUE.

<INPUT TYPE=TEXT NAME="ADDRESS" VALUE="44 XYZ">

This will automatically pair the value 44 XYZ with the name ADDRESS, unless the user changes it. Take care to use quotes as specified in the example.

We can specify the size of the text input box.

<INPUT TYPE=TEXT NAME="ADDRESS" VALUE="44 XYZ" SIZE=10>

The default value is 20. You already know that the *default value* is the value that the browser assumes if you have not told it otherwise.

Go ahead and remove VALUE="44 XYZ".
If we want, we can specify how many characters a user can input.
Experiment with this and try to input more than 10 characters! The MAXLENGTH attribute is used to restrict the number of characters to be entered in the textbox.

<INPUT TYPE=TEXT NAME="ADDRESS" SIZE=20 MAXLENGTH=10>

Very similar to the TYPE=TEXT is the TYPE=PASSWORD. It is exactly the same, except that for security it displays *** instead of the actual input. The text entered as password would not be echoed on the page. So you can use this whenever you want to accept a password or some other sensitive information from the user.

<INPUT TYPE=PASSWORD>

Remember that each <INPUT> must have a NAME**, that gives the name of the field.**

<INPUT TYPE=PASSWORD NAME="USER PASSWORD">

The SIZE, VALUE, and MAXLENGTH attributes work here also just as they do with TEXT

3.6.3 Radio Button

Radio buttons are used when only one out of the group of options is to be chosen. In the example code we have put a line break after each button.

Each of the Radio Buttons must be assigned a VALUE and you must label each button. The code given in Figure 3.18 illustrates this. You can also modify these labels with other HTML tags if you wish.

This takes care of the basics of your radio buttons. You can tidy things up by adding a statement above the buttons, and if you want choose a default selection (optional). To choose a default selection you need to add “CHECKED” with the appropriate radio button. The user of course can only choose one option. The choice will be returned to you as the name/value pair BEST FRIEND=Ajay (or whatever the user picks).

3.6.4 Checkbox

Checkboxes are used when one or more out of the group of options is to be chosen. Building Check boxes is very similar to radio buttons. Figure 3.18 illustrates the use of Checkbox.

The user can choose any number such as 1, 2, none or all of the options. The choices will be returned to you as the name/value pairs
Ajay=YES
Tarun=YES
(or whatever the user chooses. If nothing, nothing will be returned to you)

The following is code for making a table containing different options for 3 different questions. Try it yourself and see the result.

```
<CENTER>
<TABLE WIDTH=600 BORDER=1 CELLSPACING=1><TR>
<TD WIDTH=199>
Which of these guys are your friends?<BR>
<INPUT TYPE=CHECKBOX NAME="Friend?..Ajay" VALUE="YES"> Ajay
<BR>
<INPUT TYPE=CHECKBOX NAME="Friend?..Rohan" VALUE="YES"> Rohan
<BR>
<INPUT TYPE=CHECKBOX NAME="Friend?..Tarun" VALUE="YES">
Tarun<BR>
<INPUT TYPE=CHECKBOX NAME="Friend?..BU" VALUE="YES"> Bunty<P>
</TD>
<TD WIDTH=200>
Which of these guys would you lend money to?<BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...Ajay" VALUE="YES">
Ajay <BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...Rohan" VALUE="YES">
Rohan <BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...Tarun" VALUE="YES">
Tarun<BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...BU" VALUE="YES">
Bunty<P>
</TD>
<TD WIDTH=199>
Which of these guys would you trust with your sister?<BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...Ajay" VALUE="YES"> Ajay
```

```
<BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...Rohan" VALUE="YES">
Rohan <BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...Tarun" VALUE="YES">
Tarun<BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...BU" VALUE="YES">
Bunty<P>
</TD>
</TR></TABLE>
</CENTER>
```

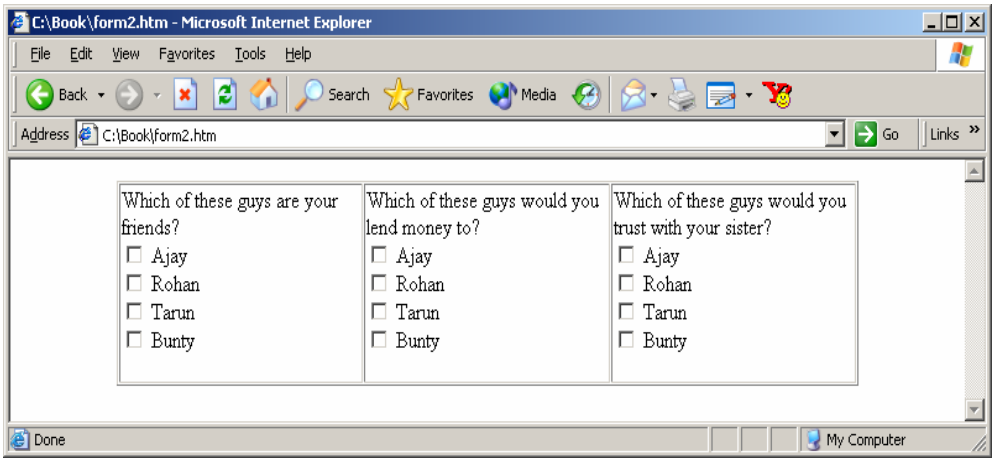


Figure 3.6: A Form on a Web Page with Checkboxes

3.6.5 SELECT Tag and Pull Down Lists

The next type of input is a Pull Down List. With this type you have to use <SELECT> instead of <INPUT> and it also has a closing tag. This control is used when we have a long list of items to be displayed. The user can also choose any item. All we have to do to turn it into a Scrolling List is to add a SIZE attribute to the <SELECT> tag. The SIZE is simply how many options show in the window.

3.6.6 Hidden

Yet another type of input is HIDDEN input.

<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Friend Form 1">

A HIDDEN input is a name/value pair that is returned to you but does not show up anywhere on the web page. The hidden input above is needed for use with the *mailto* Formatter (MTF). It is how MTF recognizes the forms it has to parse.

Suppose you were a company trying to generate leads for a new product. You have a standard form for gathering information that has name, company, phone, products interested in, etc. The only problem is there are 6 slightly different versions of the form in 6 different places. You need to know what is coming from where. What to do?

You could add a HIDDEN input to your forms like this:

```
<HTML>
<BODY>
<FORM METHOD=POST>
<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Version 1"> ...for the
first version
```

<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Version 2"> ...for the second version
<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Version 3"> ...for the third version
</FORM>
</HTML>

By the way, it doesn't matter what the name/value pair in the hidden input is (or *any* input for that matter).

<INPUT TYPE=HIDDEN NAME="E" VALUE="Mc^2"> HIDDEN inputs are also useful for cgi scripts. For example, many Internet Service Providers have a script you can have your forms sent to. It then sends the form back to you properly formatted. The hidden input can be used to tell the cgi script who you are, where to send the parsed data, and so on.

3.6.7 Submit and Reset

Submit and Reset are special types of input buttons. Submit is used to send the data to the server and Reset clears/resets the form.

<INPUT TYPE=SUBMIT>
<INPUT TYPE=RESET>

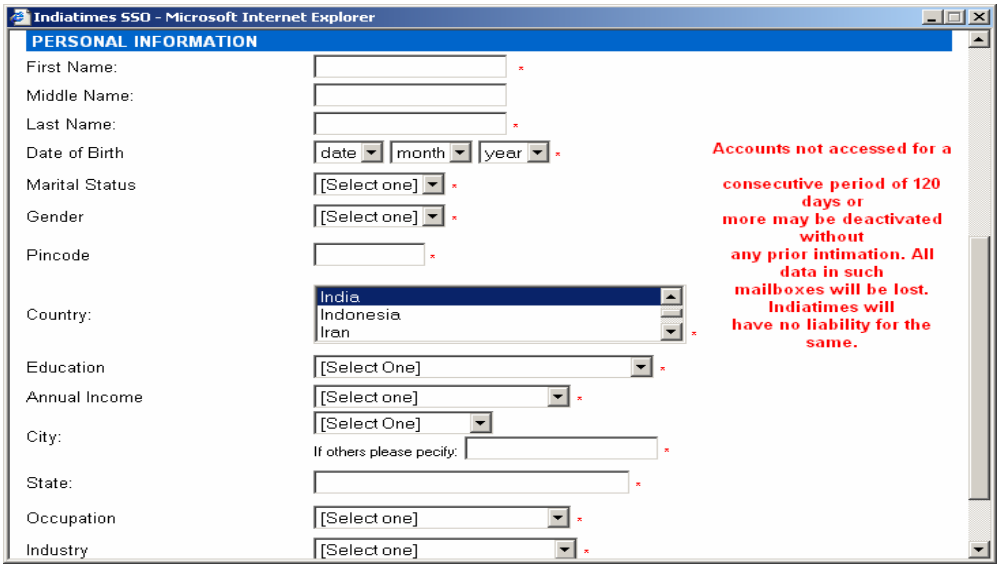
We can easily change what the buttons say.

<INPUT TYPE=SUBMIT VALUE="Send it off!">
<INPUT TYPE=RESET VALUE="Clear the form!"><P>

If necessary, the SUBMIT button can also have a NAME. You would need this if, for whatever reason, you had more than one SUBMIT button.

Check Your Progress 4

Create a Web page similar to the following:



3.7 SOME SPECIAL TAGS

We now look at some new tags that have been added to HTML in the latest versions.

3.7.1 COLGROUP

<COLGROUP> defines a group of columns in the table and allows you to set the properties of those columns. <COLGROUP> goes immediately after the <TABLE> tag and before any <TR>. <COLGROUP> works very much like <COL>, but you should note that <COLGROUP> requires both an opening and a closing tag.

```
<TABLE BORDER=1 CELLPADDING=4 RULES=GROUPS FRAME=BOX>
<COLGROUP></COLGROUP>
<COLGROUP SPAN=3></COLGROUP>
```

Comment: This code fragment does not at all explain the tag. Please give a more informative example

3.7.2 THEAD, TBODY, TFOOT

<THEAD>, <TBODY>, and <TFOOT> form groups of rows. <THEAD> indicates that a group of rows are the header rows at the top of the table. <TBODY> indicates that a group of rows are body rows. <TFOOT> indicates that a group of rows are the footer rows at the bottom of the table.

The most popular use for these three tags, which are currently only recognized by MSIE 4 and up, is to put borders between groups of rows instead of between every two rows. For example, suppose you have a table in which you want borders around the top row, the bottom row, and around the entire block of rows in between. You could do that with the following code. Note that in addition to <THEAD>, <TBODY>, and <TFOOT> you also must use <TABLE RULES=GROUPS>. The following example shows the use of these tags:

```
<HTML>
<BODY>
<TABLE CELLPADDING=6 RULES=GROUPS  FRAME=BOX>
<THEAD>
<TR> <TH>Weekday</TH> <TH>Date</TH> <TH>Manager</TH>
<TH>Qty</TH> </TR>
</THEAD>
<TBODY>
<TR> <TD>Mon</TD> <TD>09/11</TD> <TD>Komal</TD>  <TD>639</TD>
</TR>
<TR> <TD>Tue</TD> <TD>09/12</TD> <TD>Lovely</TD> <TD>596</TD>
</TR>
<TR> <TD>Wed</TD> <TD>09/13</TD> <TD>Rohan</TD>   <TD>1135</TD>
</TR>
<TR> <TD>Thu</TD> <TD>09/14</TD> <TD>Suresh</TD>   <TD>1002</TD>
</TR>
<TR> <TD>Fri</TD> <TD>09/15</TD> <TD>Rohan</TD>   <TD>908</TD>
</TR>
<TR> <TD>Sat</TD> <TD>09/16</TD> <TD>Lovely</TD> <TD>371</TD>
</TR>
<TR> <TD>Sun</TD> <TD>09/17</TD> <TD>Suresh</TD>   <TD>272</TD>
</TR>
</TBODY>
<TFOOT><TR> <TH ALIGN=LEFT COLSPAN=3>Total</TH> <TH>4923</TH>
</TR></TFOOT>
</TABLE>
```

</HTML>

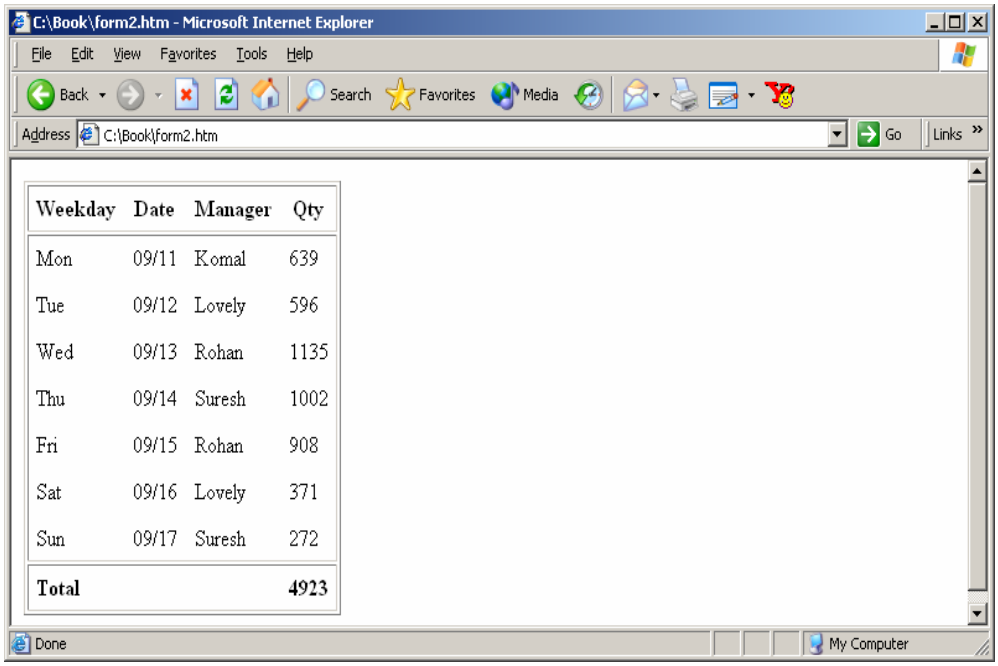


Figure 3.7: Using the THEAD, TBODY and TFOOT Tags

3.7.3 _blank, _self, _parent, _top

These all are attributes of the <A> tag. The following example explains each of these attributes.

```
<HTML>
<BODY>
<UL>
<LI>TARGET = "_blank"</LI>
<A HREF="newwindow.html" TARGET="_blank">a new window</A> <BR>

<LI> TARGET = "_self"</LI>
go to <A HREF="selftarget.html" TARGET="_self">next</A> page<BR>

<LI>TARGET = "_top"</LI>
<A HREF="selftarget.html" TARGET="_top">top</A><BR>
</UL>
</HTML>
```

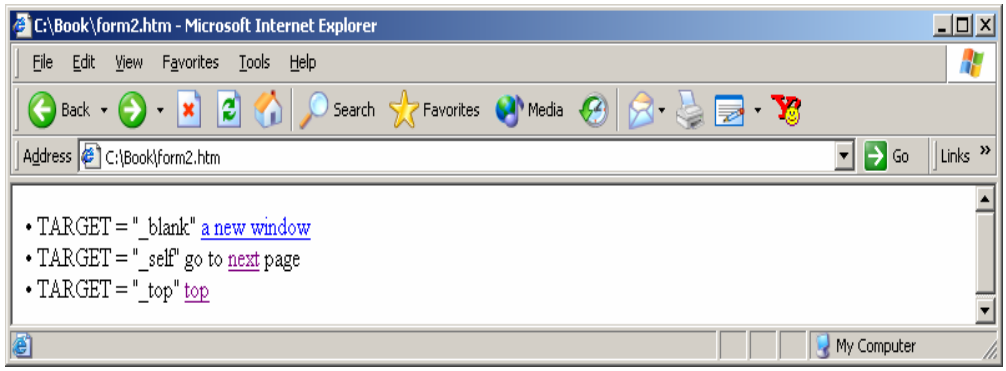


Figure 3.8: Attributes of the <A> Tag

- **TARGET = "_blank"**

"_blank" opens the new document in a new window. Run the code given in Figure 3.22 and check how it works. This value does not require the use of any frames. "_blank" is popular in web pages which are devoted to links to "other resources on the net". By opening a new window for each resource, the user has a sense of a "main" page (the list of resources) and "secondary" pages (each individual resource). Also, the web site providing the link does not risk being supplanted by a link that it has provided.

Note: Some versions of MSIE do not support "_BLANK"

- **TARGET = "_self"**

"_self" puts the new document in the same window and frame as the current document. "_self" works the same as if you had not used TARGET at all. For an example see Figure 3.22.

- **TARGET = "_parent"**

"_parent" is used in a situation where a frameset file is nested inside another frameset file. A link in one of the inner frameset documents which uses "_parent" will load the new document where the inner frameset file had been.

If the current document's frameset file does not have any "parent", then "_parent" works exactly like "_top": the new document is loaded in the full window. Note that "_parent" does not work in a frameset, which is merely nested inside another frameset in the same frameset file.

- **TARGET = "_top"**

"_top" loads the linked document in the topmost frame, that is, the new page fills the entire window.
Refer to Figure 3.22.

3.7.4 IFRAME

<IFRAME> is an HTML 4.0 addition to the frames toolbox. Currently only MSIE supports <IFRAME>. Unlike frames created using <FRAMESET> and <FRAME>, <IFRAME> creates a frame that sits in the middle of a regular non-framed web page. <IFRAME> works like <IMG>, only instead of putting a picture on the page, it puts another web page.

For example, suppose within the same directory as this page there is a file called "hello.html". This code puts hello.html into an inline frame:

<IFRAME SRC="hello.html" WIDTH=450 HEIGHT=100>

If you can see this, your browser does not understand IFRAME. However, we'll still link <A HREF="hello.html">link</A> you to the file.

</IFRAME>
which gives us this inline frame:

Here’s what the code means:

IFRAME

The name of the <IFRAME> tag
SRC="hello.html"
The URL of the document to show in the inline frame.
WIDTH=450 HEIGHT=100

The dimensions of the inline frame.

If you can see this, your browser doesn't understand IFRAME. However, we'll still <A HREF="hello.html">link</A> you to the file.

Code between <IFRAME> and </IFRAME> is not displayed by browsers that understand <IFRAME>. Browsers that do not understand <IFRAME> will display this code (because they don't know how to ignore it).

You can do most of the things with <IFRAME> that you can do with regular frames, including setting the frame border, internal margins, and setting information on scroll bars. You can use the attribute so that you can set links to the target frame.

3.7.5 <LABEL>

<LABEL>, an HTML 4.0 element supported by MSIE and Netscape 6, defines a set of text that is associated with a particular form element. For example, the code below indicates that the phrase "send more information" is associated with the “moreinfo” checkbox because the checkbox is within the <LABEL> element:

```
<HTML>
<BODY>
<LABEL FOR="moreinfo">
send more information
<INPUT NAME="moreinfo" TYPE=CHECKBOX ID="moreinfo">
</LABEL>
</HTML>
```

The FOR attribute is required in the above example. The value of FOR should be the same as the value of ID in the form field that the label applies to.

You can also associate a <LABEL> with a field that is not within its contents using the FOR attribute.

3.7.6 Attribute for <SELECT>

TABINDEX = *integer*
TABINDEX is supported by MSIE 4.x and higher and Netscape 6.

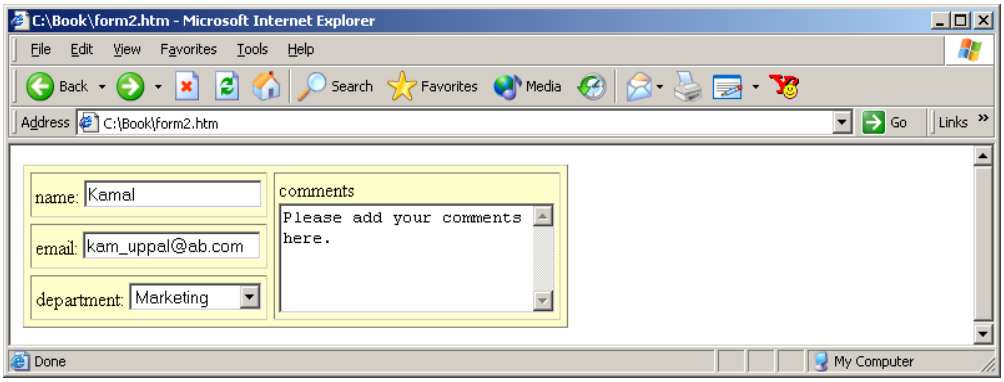
Normally, when the user tabs from field to field in a form (in a browser that allows tabbing, not all browsers do) the tab order is the order in which the fields appear in the HTML code.

However, sometimes you want the tab order to flow a little differently. In that case, you can number the fields using TABINDEX. The tabs then flow in order from the one with the lowest TABINDEX to the highest.

The code below illustrates this:

```
<HTML>
<BODY>
<TABLE BORDER CELLPADDING=3 CELLSPACING=5
BGCOLOR="#FFFFCC">
<TR>
    <TD>name: <INPUT NAME="realname" TABINDEX=1></TD>
    <TD ROWSPAN=3>comments<BR>
```

```
    <TEXTAREA COLS=25 ROWS=5
TABINDEX=4></TEXTAREA></TD></TR>
<TR> <TD>email: <INPUT NAME="email" TABINDEX=2></TD></TR>
<TR> <TD>department: <SELECT NAME="dep" TABINDEX=3>
    <OPTION VALUE="">...
    <OPTION VALUE="mkt">Marketing
    <OPTION VALUE="fin">Finance
    <OPTION VALUE="dev">Development
    <OPTION VALUE="prd">Production</SELECT></TD></TR>
</TABLE>

</HTML>
```



TABINDEX can also be used with <A>, <INPUT>, <TEXTAREA>, and <BUTTON>.

3.7.7 TEXTAREA

<TEXTAREA> indicates a form field where the user can enter large amounts of text. In most respects, <TEXTAREA> works like an <INPUT> field. It can have a name and a default value. It has to be noticed that <TEXTAREA> is a container tag, so it has a start tag and an ending tag.

In its simplest form, <TEXTAREA> requires the NAME, COLS and ROWS attributes, and nothing between <TEXTAREA ...> and </TEXTAREA>. Look at the code fragment shown below

```
<FORM ACTION="../cgi-bin/mycgi.pl" METHOD=POST>
your comments:<BR>
<TEXTAREA NAME="comments" COLS=40 ROWS=6></TEXTAREA>

<P><INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
```

gives us this form:

your comments:

The matter between <TEXTAREA ...> and </TEXTAREA> are used as the default value

```
<FORM ACTION="../cgi-bin/mycgi.pl">
your response:<BR>
<TEXTAREA NAME="comments" COLS=40 ROWS=6>
Kamal said
: I think it's a great idea
: but it needs more thought
</TEXTAREA>
```

Comment: The browser output has been given for all the three code fragments. It would be better to make it clear that they are fragments and are not complete code that will run directly.

Comment: Maybe you would like this sentence to be inside a <PRE> tag.

<P><INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
gives us your response:

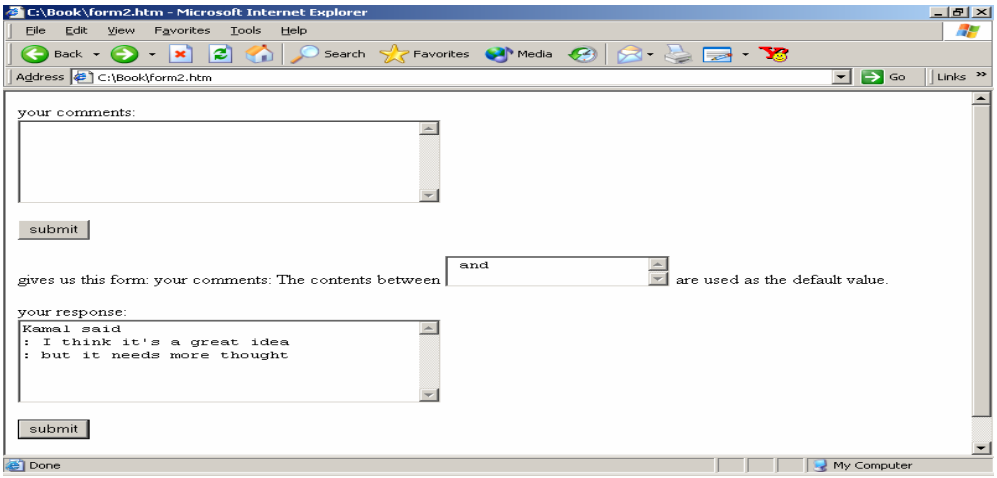


Figure 3.10: Using the TEXTAREA Tag

The contents are interpreted as text only; HTML markup is ignored. Theoretically the user can type unlimited amounts of text into the textarea field. In practice the browser sets a limit that is usually no more than 32 K. If you want users to send in their latest novel, consider using some file upload mechanism.

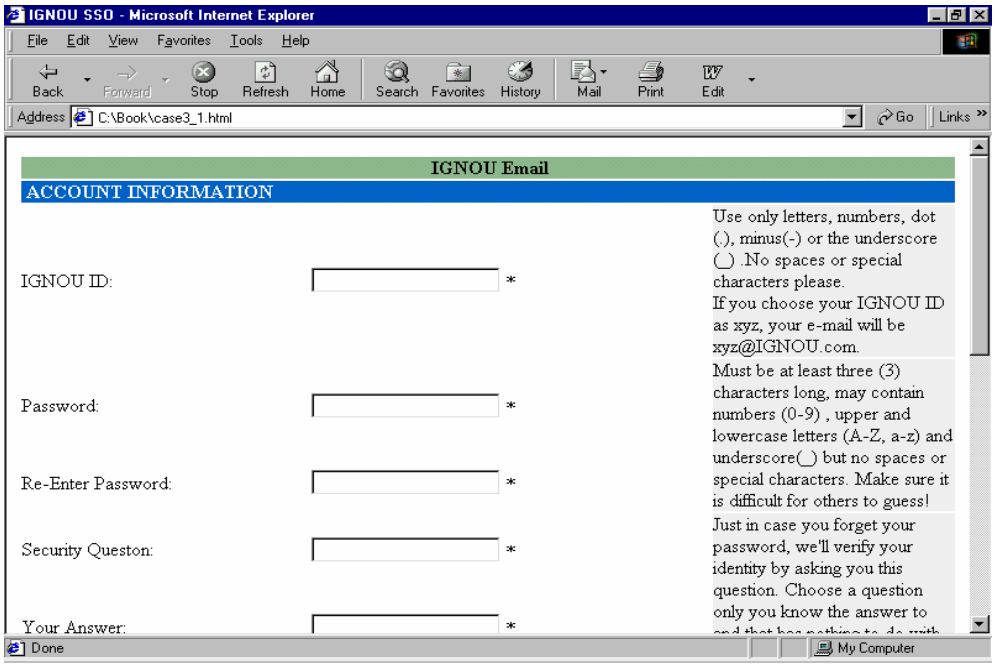
Case Study: Suppose your boss has asked you to create a Web page from which customers can order computer equipment. You need to collect the customer's name, address, phone number, age, credit card information, and what the customer wants to order.

```
<HTML>
  <HEAD>
    <TITLE>ComputoRama Order Form</TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION="mailto:mail@abc.com" METHOD=POST>
      <TABLE BORDER="2" CELLPADDING="1">
        <TR>
          <TD ROWSPAN="2">Who Are You?</TD>
          <TD><INPUT TYPE="text" NAME="FirstName" SIZE=2Ø></TD>
          <TD><INPUT TYPE="text" NAME="MiddleInitial" SIZE=1></TD>
          <TD><INPUT TYPE="text" NAME="LastName" SIZE=2Ø></TD>
          <TD><INPUT TYPE="text" NAME="Age" SIZE=3></TD>
        </TR>
        <TR>
          <TD><FONT SIZE="-2">First Name</FONT></TD>
          <TD><FONT SIZE="-2">MI</FONT></TD>
          <TD><FONT SIZE="-2">Last Name></TD>
          <TD><FONT SIZE="-2">Age</TD>
        </TR>
       <TR>
          <TD ROWSPAN="3">How Do We Contact You?</TD>
          <TD COLSPAN="4" VALIGN="TOP">Street Address: <TEXTAREA
          name="StreetAddress" rows=2 cols=3Ø></TEXTAREA></TD>
        </TR>
        <TR>
          <TD COLSPAN="2">City: <INPUT TYPE="text" NAME="City"
          SIZE=20></TD>
```

```
            <TD COLSPAN="2">State: <INPUT TYPE="text" NAME= "State"
            SIZE=2></TD>
          </TR>
          <TR>
            <TD COLSPAN="2">ZIP Code: <INPUT TYPE="text"
NAME="ZIPCode"
            SIZE=10></TD>
            <TD COLSPAN="2">Daytime Phone
              (<INPUT TYPE="text" NAME="Phone1" SIZE=3>)
              <INPUT TYPE="text" NAME="Phone2" SIZE=3>-
              <INPUT TYPE="text" NAME="Phone3" SIZE=4></TD>
          </TR>
          <TR>
            <TD>Credit Card
              <INPUT TYPE="radio" NAME="CreditCardType" VALUE="Visa"
              CHECKED>Visa
              <INPUT TYPE="radio" NAME="CreditCardType"
              VALUE="MasterCard">M/C</TD>
            <TD COLSPAN="2" ALIGN="CENTER">
              <INPUT TYPE="text" NAME="CreditCardNumber1" SIZE=4>
              <INPUT TYPE="text" NAME="CreditCardNumber2" SIZE=4>
              <INPUT TYPE="text" NAME="CreditCardNumber3" SIZE=4>
              <INPUT TYPE="text" NAME="CreditCardNumber4" SIZE=4></TD>
            <TD COLSPAN="2">Expiration Date:
              <INPUT TYPE="text" NAME="ExpirationMonth" SIZE=2>/
              <INPUT TYPE="text" NAME="ExpirationYear" SIZE=2></TD>
          </TR>
          <TR>
            <TD>Merchandise</TD>
            <TD COLSPAN=4><SELECT MULTIPLE NAME="Merchandise"
SIZE=1>
              <OPTION SELECTED> HAL-47Ø <OPTION> Banana9ØØØ
              <OPTION> High Res Monitor <OPTION> Low Res Monitor
              <OPTION> Deluxe Keyboard <OPTION> Regular Keyboard
              <OPTION> Laser Printer <OPTION> Inkjet Printer <OPTION>
              Dot Matrix Printer
              <OPTION> Mouse <OPTION> Trackball
              <OPTION> Scanner
              </SELECT></TD>
          </TR>
          <TR>
            <TD ALIGN=CENTER COLSPAN="5">
              <H1>Thank You For Your Order!</H1>
            </TD>
          </TR>
        </TABLE>
        <CENTER>
          <INPUT TYPE="submit" VALUE="Ship It!"> <INPUT TYPE="reset"
          VALUE="Clear Entries">
        </CENTER>
      </FORM>
    </BODY>
</HTML>
```

Check Your Progress 4

Suppose you are asked to design a registration form for members of a web site. Registration information will include hobbies, interests, assets owned by the member etc. Create the following Web page.**(Case study)**



3.8 SUMMARY

In this unit we have learned some important and advanced topics of HTML. You should now be able to develop interactive Web pages also. We have discussed ways of linking to different locations in a Web site. We have learned about lists that allow us to develop Web pages with elementary data. Lists can be of three types i.e. Ordered Lists, Unordered Lists and Definition Lists. We all know how important a table is in any document -- from someone's bio data to a Chief Executive's report a table is the simplest way to understand the information. If we display the data in paragraph form then it would look unorganized and would be difficult to comprehend. On the other hand, a table presents all the data at a glance. We have also seen how frames are used to divide a page into multiple sections. For example if we want to create a Web page that contains information on three different topics, then we could use frames.
One of the most important points in Web pages is the interaction with the end user. For creating interactive Web pages we use Forms. In a form we can have different controls like Text box, Radio buttons, check box, drop down lists etc. All these controls allow us to interact with the user accept inputs from the user. In the final section we have learned about some of the recently introduced tags that are available in the latest version of HTML.

3.9 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. Following is the code to design a Web page that provides links to five different Web sites.

```
<HTML>
<TITLE> Link to five different Web sites </title>
<BODY>
```

```
Web site 1 <A HREF="http://www.hotmail.com/">Hotmail</A>
Web site 2 <A HREF="http://www.google.com/">google</A>
Web site 3 <A HREF="http://www.astalavista.com/">altavista</A>
Web site 4 <A HREF="http://www.education.com/">education</A>
Web site 5 <A HREF="http://www.computer.com/">computer</A>
</BODY>
</HTML>
```

Comment: The output of this code is not very pleasing – all the options are on the same physical line. Could be improved

2. Following is the code for the given page

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<B><FONT SIZE=5><P>GENERAL TABLE OF CONTENTS</P>
</FONT><P>So, you want to make a Web page</P></B> LESSONS:
<P><A HREF="../../makapage/introduction.htm">Introduction</A></P>
<P><A HREF="../../makapage/lesson01.html">Lesson 1</A>: getting started, saving
as html <BR>
<A HREF="../../makapage/lesson02.html">Lesson 2</A>: backgrounds, bold, italic,
font &amp;amp; fontsize <BR>
<A HREF="../../makapage/lesson03.html">Lesson 3</A>: basic formatting, line
breaks, inserting images <BR>
<A HREF="../../makapage/lesson04.html">Lesson 4</A>: links, images, thumbnails,
anchors <BR>
<A HREF="../../makapage/lesson05.html">Lesson 5</A>: lists, more formatting,
horizontal rules, comments <BR>
<A HREF="../../makapage/lesson06.html">Lesson 6</A>: more resources <BR>
<A HREF="../../makapage/picker/index.html">Color Picker</A> <BR>
<A HREF="../../makapage/dafonts/master.html">Handy Dandy Font Viewer</A> -
<A HREF="../../makapage/dafonts/index.html">(intro)</A> <BR>
<A HREF="../../makapage/special.html">Special Characters</A> <BR>
<A HREF="../../makapage/upload.html">Upload your page to the Web</A> </P>
<P>&amp;nbsp;</P>
<FONT SIZE=2><P> </P></FONT></BODY>
</HTML>
```

Check Your Progress 2

Code of the given page

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<P> Following are the key to success: </P>
<UL>
<LI><img src = “arrow.gif”>
Devotion </LI>
<LI><img src = “arrow .gif”>
Hardwork </LI>
<LI><img src =”arrow.gif”>
Smart work </LI>
</UL>
<P> Following are the key to success: </P>
<OL>
<LI> Devotion </LI>
<LI> Hardwork </LI>
<LI> Smartwork </LI>
```

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

Check Your Progress 3

1. Following is the code for a Web page that has 5 equal columns.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<TABLE BORDER CELLSPACING=3 BORDERCOLOR="#000000">
<TR><TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
</TR>
</TABLE>
<P> </P></BODY>
</HTML>
```

2. Design your Bio-Data in HTML Page.

```
<HTML>
<HEAD>
</HEAD>
<BODY>

<B><U><P>Qualifications</P></B></U>
<P ALIGN="JUSTIFY"><B>X &amp;amp; XII</B> from CBSE Board.</P>
<P ALIGN="JUSTIFY"><B>BCOM (PASS)</B> from Delhi University in
1999.</P>
<P ALIGN="JUSTIFY"><B>O Level</B> from DOEACC Society India in
2000.</P>
<P ALIGN="JUSTIFY"><B>MCP </B>(Microsoft Certified Professional) in 2001 in
Analyzing Requirements and Defining Solutions Architecture with
<U>93.9%</U>.</P>
<P ALIGN="JUSTIFY"><B>CIC</B> (Certificate In Computing) from Indira
Gandhi National Open University with <U>Ist Division</U>.  </P>
<P ALIGN="JUSTIFY"><B>PGDCA</B> (Post Graduate Diploma in Computer
Applications) from IGNOU with <U>Ist Division</U>.</P>
<P ALIGN="JUSTIFY"><B>ADCA </B>(Advance Diploma in Computer
Applications) from IGNOU with <U>Ist Division</U>.</P>
<P ALIGN="JUSTIFY"><B>MCA </B>from Indira Gandhi National Open
University with <U>Ist Division</U>. </P>
<B><U><P>Languages Known</P>
<P ALIGN="JUSTIFY"></B></U>C including data structures, C++ including data
structure, MS COBOL 4.5, MS FORTRAN 77, Assembly using MASM, Unix Shell
Programming,<FONT SIZE=5> </FONT>HTML, Corman Common LISP, Visual
Basic 6.0.</P>
<B><U><P>Personal Details</P>
</B></U><P>Name                    ABC</P>
```

```
<P>Father’s Name<B>&#9;</B>XYZ&#9;</P>
<P>Date of Birth<B>&#9;</B>23<SUP>rd</SUP> Jan,1979.&#9;</P>
<P>Address<B>&#9;</B>A-2, East Of Kailash, New Delhi.&#9;</P>
<P>Tel.<B>&#9;</B>29090909</P>
<P>Email<B>&#9;</B>ram@hotmail.com</P>
<P>Sex<B>&#9;</B>Male&#9;</P>
<P>Marital Status<B>&#9;</B>Single</P>
<B><U><P>Interests and activities</P>
</B></U><P>Troubleshooting hardware and software problems.&#9;</P>
<B><U><P>Additional Assets</P>
</B></U><P>Hard work, devotion and curiosity to learn and adapt to new
environments.&#9;</P>
<FONT SIZE=2><P> </P>
</FONT><B><P>Date:</B><FONT SIZE=2>
<B> (RAM)</P>
<P>Place:</P></B></FONT></BODY>
</HTML>
```

Check Your Progress 4

```
<head>
<title>IGNOU SSO</title>
<META HTTP-EQUIV="expires" CONTENT="Sat, 15 Mar 2003 23:59:59 GMT">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="Content-Style-Type" content="text/css">
</head>
<body>
<form name= "theForm" action= "/suggest/emailRegistrationServlet" method=POST
onsubmit="return validate()">
  <input type="Hidden" name="successURL"
value="http://email.IGNOU.com/success.html">
  <input type="Hidden" name="pageTitle" value="IGNOU Email">
  <input type="Hidden" name="sourceChannel" value="Email">
  <table border=0 cellpadding=0 width="100%">
    <tbody>
      <tr bgcolor=#8fbc8f>
        <td colspan=3 align=center><b class="tdw"> IGNOU Email</b></td>
      </tr>
      <tr bgcolor=#0066cc>
        <td colspan=3><b> <font color="#FFFFFF"><span >ACCOUNT
          INFORMATION</span></font></b></td>
      </tr>
      <tr>
        <td valign=center width="31%" class="td"> IGNOU ID: </td>
        <td valign=center width="43%"> <input name=username> <span >*</span>
        </td>
        <td bgcolor=#f0f0f0 width="26%" class="td1">Use only letters, numbers,
          dot(.), minus(-) or the underscore (_) .No spaces or special characters
          please.<br> <span >If you choose your IGNOU ID as xyz,
          your e-mail will be xyz@IGNOU.com.</span> </td>
      </tr>
      <tr>
        <td height=42 width="31%" class="td">Password:</td>
        <td height=42 width="43%"> <input name=password type=password>
<span>*</span></td>
        <td bgcolor=#f0f0f0 rowspan=2 width="26%" class="td1">Must be at least
          three (3) characters long, may contain numbers (0-9) , upper and lowercase
          letters (A-Z, a-z) and underscore(_) but no spaces or special characters.
```

```
        Make sure it is difficult for others to guess! </td>
      </tr>
      <tr>
        <td width="31%" class="td">Re-Enter Password:</td>
        <td width="43%"> <input name=confirmPassword type=password> <span
>*</span></td>
      </tr>
      <tr>
        <td width="31%" class="td">Security Queston:</td>
        <td width="43%"> <input name=secretQus> <span >* </span></td>
        <td bgcolor=#f0f0f0 rowspan=2 width="26%" class="td1">Just in case you
          forget your password, we'll verify your identity by asking you this
          question. Choose a question only you know the answer to and that has
          nothing to do with your password whatsoever.</td>
      </tr>
      <tr>
        <td height=2 width="31%" class="td">Your Answer:</td>
        <td height=2 width="43%"> <input name=secretAns> <span >*</span>
        </td>
      </tr>
      <tr>
        <td width="31%" class="td">Alternative Email</td>
        <td width="43%"> <input maxlength=100 name=altEmail size=35>
  <span >(optional)
        </span></td>
        <td bgcolor=#f0f0f0 width="26%" class="td1"> This will help you to retrieve
          your IGNOU ID <br>
          and password in case you forget it. </td>
      </tr>
      <tr bgcolor=#0066cc>
        <td colspan=3><b class="tdw"> <font color="#FFFFFF"><span
class="tdb">PERSONAL
          INFORMATION</span></font></b></td>
      </tr>
      <tr>
        <td width="31%" class="td">First Name:</td>
        <td width="43%"> <input name=firstName
document.theform.firstname.focus();>
          <input type=hidden name=frmURL>
        <input type=hidden name="education" value="u">
        <span >* </span></td>
        <td width="26%"> </td>
      </tr>
      <tr>
        <td width="31%" class="td">Middle Name:</td>
        <td width="43%"> <input name=MiddleName> </td>
        <td width="26%"> </td>
      </tr>
      <tr>
        <td width="31%" class="td">Last Name:</td>
        <td width="43%"> <input name=lastName> <span >*</span> </td>
        <td width="26%"> </td>
      </tr>
      <tr>
        <td width="31%" class="td">Date of Birth</td>
        <td width="43%"> <font face="MS Sans Serif" size=2>
          <select name="date">
```

```
<option selected><font face="Arial, Helvetica, sans-
serif">date</font></option>
<option value="01"><font face="Arial, Helvetica, sans-
serif">01</font></option>
<option value="02"><font face="Arial, Helvetica, sans-
serif">02</font></option>
<option value="03"><font face="Arial, Helvetica, sans
serif">03</font></option>
<option value="04"><font face="Arial, Helvetica, sans-
serif">04</font></option>
<option value="05"><font face="Arial, Helvetica, sans-
serif">05</font></option>
<option value="06"><font face="Arial, Helvetica, sans-
serif">06</font></option>
<option value="07"><font face="Arial, Helvetica, sans-
serif">07</font></option>
<option value="08"><font face="Arial, Helvetica, sans-
serif">08</font></option>
<option value="09"><font face="Arial, Helvetica, sans-
serif">09</font></option>
<option value="10"><font face="Arial, Helvetica, sans-
serif">10</font></option>
<option value="11"><font face="Arial, Helvetica, sans-
serif">11</font></option>
<option value="12"><font face="Arial, Helvetica, sans-
serif">12</font></option>
<option value="13"><font face="Arial, Helvetica, sans-
serif">13</font></option>
<option value="14"><font face="Arial, Helvetica, sans-
serif">14</font></option>
<option value="15"><font face="Arial, Helvetica, sans-
serif">15</font></option>
<option value="16"><font face="Arial, Helvetica, sans-
serif">16</font></option>
<option value="17"><font face="Arial, Helvetica, sans-
serif">17</font></option>
<option value="18"><font face="Arial, Helvetica, sans-
serif">18</font></option>
<option value="19"><font face="Arial, Helvetica, sans-
serif">19</font></option>
<option value="20"><font face="Arial, Helvetica, sans-
serif">20</font></option>
<option value="21"><font face="Arial, Helvetica, sans-
serif">21</font></option>
<option value="22"><font face="Arial, Helvetica, sans-
serif">22</font></option>
<option value="23"><font face="Arial, Helvetica, sans-
serif">23</font></option>
<option value="24"><font face="Arial, Helvetica, sans-
serif">24</font></option>
<option value="25"><font face="Arial, Helvetica, sans-
serif">25</font></option>
<option value="26"><font face="Arial, Helvetica, sans-
serif">26</font></option>
<option value="27"><font face="Arial, Helvetica, sans-
serif">27</font></option>
<option value="28"><font face="Arial, Helvetica, sans-
serif">28</font></option>
```

```
<option value="29"><font face="Arial, Helvetica, sans-
serif">29</font></option>
<option value="30"><font face="Arial, Helvetica, sans-
serif">30</font></option>
<option value="31"><font face="Arial, Helvetica, sans-
serif">31</font></option>
</select>
<select name="month">
<option selected><font face="Arial, Helvetica, sans-
serif">month</font></option>
<option value="01"><font face="Arial, Helvetica, sans-
serif">01</font></option>
<option value="02"><font face="Arial, Helvetica, sans-
serif">02</font></option>
<option value="03"><font face="Arial, Helvetica, sans-
serif">03</font></option>
<option value="04"><font face="Arial, Helvetica, sans-
serif">04</font></option>
<option value="05"><font face="Arial, Helvetica, sans-
serif">05</font></option>
<option value="06"><font face="Arial, Helvetica, sans-
serif">06</font></option>
<option value="07"><font face="Arial, Helvetica, sans-
serif">07</font></option>
<option value="08"><font face="Arial, Helvetica, sans-
serif">08</font></option>
<option value="09"><font face="Arial, Helvetica, sans-
serif">09</font></option>
<option value="10"><font face="Arial, Helvetica, sans-
serif">10</font></option>
<option value="11"><font face="Arial, Helvetica, sans-
serif">11</font></option>
<option value="12"><font face="Arial, Helvetica, sans-
serif">12</font></option>
</select>
<select
size=1 name=year>
<option selected>year</option>
<option value="1970">1970</option>
<option value="1971">1971</option>
<option value="1972">1972</option>
<option value="1973">1973</option>
<option value="1974">1974</option>
<option value="1975">1975</option>
<option value="1976">1976</option>
<option value="1977">1977</option>
<option value="1978">1978</option>
<option value="1979">1979</option>
<option value="1980">1980</option>
<option value="1981">1981</option>
<option value="1982">1982</option>
<option value="1983">1983</option>
<option value="1984">1984</option>
<option value="1985">1985</option>
<option value="1986">1986</option>
<option value="1987">1987</option>
<option value="1988">1988</option>
<option value="1989">1989</option>
```

```
      <option value="1990">1990</option>
      <option value="1991">1991</option>
      <option value="1992">1992</option>
      <option value="1993">1993</option>
      <option value="1994">1994</option>
      <option value="1995">1995</option>
      <option value="1996">1996</option>
      <option value="1997">1997</option>
      <option value="1998">1998</option>
      <option value="1999">1999</option>
      <option value="2000">2000</option>
      <option value="2001">2001</option>
      <option value="2002">2002</option>
    </select>
    </font><span > *</span> </td>
  <td rowspan=5 align="center"> <font face=arial size=2 color=red><b>Accounts
    not accessed for a <br>
    consecutive period of 120 days or<br>
    more may be deactivated without<br>
    any prior intimation. All data in such<br>
    mailboxes will be lost. IGNOU will<br>
    have no liability for the same.</b> </font> </td>
</tr>
<tr>
  <td height=25 width="31%" class="td">Marital Status </td>
  <td height=25 width="43%"> <select name=maritalStatus size=1 id="select">
      <option selected value="">[Select one]</option>
      <option value="S">Single</option>
      <option value="M">Married</option>
    </select> <span >*</span> </td>
</tr>
<tr>
  <td width="31%" class="td">Gender</td>
  <td width="43%"> <font face="Arial, Helvetica, sans-serif"
      size=2>
    <select size=1 name=gender>
      <option value=""
  selected>[Select one]</option>
      <option value=M>Male</option>
      <option
  value=F>Female</option>
    </select>
    </font> <span >*</span> </td>
</tr>
<tr>
  <td height=35 width="31%" class="td">Pincode </td>
  <td height=35 width="43%"> <span class=td1r>
       <input name=pin size=10 maxlength=15 onKeyPress="if (event.keyCode <
       45 || event.keyCode > 57) event.returnValue = false;">
    </span> <span >*</span> </td>
</tr>
<tr>
  <td height=24 width="31%" class="td">Country:</td>
  <td height=24 width="43%"> <select name="country" size=3>
      <option value="in">Select One
      <option value="us">United States of America
      <option value="bs">Bahamas
      <option value="bh">Bahrain
```

```
                <option value="bd">Bangladesh
                <option value="bb">Barbados
                <option value="by">Belarus
                <option value="be">Belgium
                <option value="bz">Belize
                <option value="bj">Benin
                <option value="bm">Bermuda
                <option value="bt">Bhutan
                <option value="cv">Cape Verde
                <option value="zm">Zambia
                <option value="zw">Zimbabwe </select> <span >* </span></td>
            </tr>
            <tr>
              <td height=27 width="31%" class="td"> Education </td>
              <td height=27 width="43%"> <select name=education id="select2">
                 <option selected value="">[Select One]</option>
                  <option value=Grad_Post_Prof>Graduate/Post Graduate-
                  Professional</option>
                 <option value=Grad_Post_Gen>Graduate/Post Graduate-General</option>
                 <option value=SomeColl>SomeCollege but not Graduate</option>
                 <option value=SSC_HSC>SSC/HSC</option>
                 <option value=5to9>School 5-9 Years</option>
                 <option value=lessthan4>School upto 4 years</option>
               </select> <span >* </span></td>
              <td height=27 width="26%"> </td>
            </tr>
            <tr>
              <td height=2 width="31%" class="td">Annual Income </td>
                  <td height=2 width="43%"> <select name=incomeGroup size=1 id="select3"
                  style="HEIGHT: 22px; WIDTH: 190px">
                 <option selected value="">[Select one]</option>
                 <option value=0to1>Rs 1 Lakh &amp;amp; below</option>
                 <option value=1to3>Rs 1 Lakh -2.99 Lakhs</option>
                 <option value=3to5>Rs 3 - 4.99 Lakhs</option>
                 <option value=5to10>Rs 5 - 9.99 Lakhs</option>
                 <option value=mysterious>a mysterious sum of money</option>
                 <option value=morethan10>above Rs 10 Lakhs</option>
               </select> <span > * </span></td>
              <td height=2 width="26%"> </td>
            </tr>
            <TR>
              <TD width="31%"><FONT face="Arial, Helvetica, sans-serif"
size=2>City:</FONT></TD>

            <TD colSpan=2> <FONT face="Arial, Helvetica, sans-serif" size=2>
<SELECT name=city>
                        <option value="" selected>[Select One] </option>
                        <option value="Ahmedabad">Ahmedabad </option>
                        <option value="Bangalore">Bangalore </option>
                        <option value="Patna">Patna </option>
                        <option value="Noida">NOIDA </option>
                        <option value="Pune">Pune </option>
                        <option value="Surat">Surat </option>
                        <option value="Varanasi">Varanasi </option>
                        <option value="Vishakapatnam">Vishakapatnam</option>
                       </SELECT>
              </FONT> <FONT color=#ff0000 face=Arial size=1>*</FONT> </TD>
            </TR>
```

```
    <TR>
      <TD height=35><FONT face="Arial, Helvetica, sans-serif"
size=2>State:</FONT></TD>
      <TD height=35>
      <INPUT maxLength=100 name=state size=35>
       <FONT color=#ff0000 face=Arial size=1>*</FONT>
      </TD>
      <td height=2> </td>
    </TR>
    <TR>
      <TD height=25><FONT face="Arial, Helvetica, sans-serif"
size=2>Occupation</FONT></TD>
      <TD height=25>
      <SELECT name=occupation size=1 style="HEIGHT: 22px; WIDTH: 190px">
        <option value="" selected>[Select one]</option>
          <option value="Artist">Artist</option>
          <option value="Social worker">Social worker</option>
          <option value="Student">Student</option>
          <option value="Teacher/Administrator">Teacher / Administrator</option>
          <option value="Unemployed/Between_jobs">Unemployed / Between
jobs</option>
          <option value="No_Work">What ? me work</option>
          <option value="Others">others</option>
        </SELECT> <font color=#ff0000 face=Arial size=1>*</font> </TD>
      <TD height=25>  </TD>
    </TR>
    <TR>
      <TD height=29><FONT face="Arial, Helvetica, sans-serif"
size=2>Industry</FONT></TD>
      <TD height=29>
      <SELECT name=industry>
        <OPTION selected value="">[Select one]</OPTION>
          <OPTION value=Sales>Sales</OPTION>
          <OPTION value=Marketing>Marketing</OPTION>
          <OPTION value=Production>Production</OPTION>
          <OPTION value="Self_Employed">Self Employed</OPTION>
          <OPTION value=Student>Student</OPTION>
          <OPTION value=Others>Others</OPTION>
        </SELECT> <font color=#ff0000 face=Arial size=1>*</font> </TD>
      <TD height=29> </TD>
    </TR>
  </table>
</form>
</body>
</html>
```

Page 60: [1] Comment **milind**

Check out the code for all the FRAMES related examples. Do we need a <BODY> tag?

Page 62: [2] Comment **milind**

This is not clear. The problem is more likely with absolute numbers where 50 pixels might be a good size on a 640 X 480 screen but might be miniscule on a 1600 X 1200 screen.

UNIT 4 INTRODUCTION TO JAVASCRIPT

Structure **Page No.**

4.0 INTRODUCTION

JavaScript is the programming language of the Web. It is used mainly for validating forms. JavaScript and Java can be related to each other. There exist many other differences between the two. The client interprets JavaScript, whereas in Java, one can execute a Java file only after compiling it. JavaScript is based on an object model. In this unit you will learn how to write JavaScript code and insert them into your HTML documents, and how to make your pages more dynamic and interactive.

Besides basic programming constructs and concepts, you will also learn about Object-based programming in JavaScript. We will also discuss some commonly used objects, message boxes and forms. One of the most important parts of JavaScript is Event handling that will allow you to write Event-driven code.

4.1 OBJECTIVES

After going through this unit you would be able to learn and use the following features of the JavaScript:

- Operators;
- Loop constructs;
- Functions;
- Objects such as Math object, Date object;
- Input and output boxes;

- Event handlers;
- Form object; and
- Form array.

4.2 JAVASCRIPT VARIABLES AND DATATYPES

Let us first see the skeleton of a JavaScript file.

```
<HTML>
  <HEAD>
    <TITLE>IGNOU </TITLE>
    <SCRIPT LANGUAGE = “JavaScript”>
    </SCRIPT>
  </HEAD>
  <BODY>
  </BODY>
</HTML>
```

JavaScript code should be written between the <SCRIPT> and </SCRIPT> tags. The value LANGUAGE = “JavaScript” indicates to the browser that Javascript code has been used in the HTML document. It is a good programming practice to include the Javascript code within the <HEAD> and </HEAD> tags.

Now let us start with variables. Variables store and retrieve data, also known as "values". A variable can refer to a value, which changes or is changed. Variables are referred to by name, although the name you give them must conform to certain rules. A JavaScript identifier, or name, must start with a letter or underscore ("_"); subsequent characters can also be digits (0-9). Because JavaScript is case sensitive, letters include the characters "A" through "Z" (uppercase) and the characters "a" through "z" (lowercase). Typically, variable names are chosen to be meaningful and related to the value they hold. For example, a good variable name for containing the total price of goods orders would be *total_price*.

4.2.1 Declaring Variables

You can declare a variable with the var statement:

var strname = *some value*

You can also declare a variable by simply assigning a value to the variable. But if you do not assign a value and simply use the variable then it leads to an error.
Strname = *some value*

You assign a value to a variable like this:

var strname = "Hello"

Or like this:

strname = "Hello"

The variable name is on the left hand side of the expression and the value you want to assign to the variable is on to the right side. Thus the variable "strname" shown above gets the value "Hello" assigned to it.

Life span of variables

When you declare a variable within a function, the variable can be accessed only within that function. When you exit the function, the variable is destroyed. These

variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

4.2.2 Data Types

A value, the data assigned to a variable, may consist of any sort of data. However, JavaScript considers data to fall into several possible *types*. Depending on the type of data, certain operations may or may not be allowed on the values. For example, you cannot arithmetically multiply two string values. Variables can be of these types:

Data Types	Description
Number	3 or 7.987 are the examples of Integer and floating-point numbers. Integers can be positive, 0, or negative; Integers can be expressed in decimal (base 10), hexadecimal (base 16), and octal (base 8). A decimal integer literal consists of a sequence of digits without a leading 0 (zero). A leading 0 (zero) on an integer literal indicates it is in octal; a leading 0x (or 0X) indicates hexadecimal. Hexadecimal integers can include digits (0-9) and the letters a-f and A-F. Octal integers can include only the digits 0-7. A floating-point number can contain either a decimal fraction, an "e" (uppercase or lowercase), that is used to represent "ten to the power of" in scientific notation, or both. The exponent part is an "e" or "E" followed by an integer, which can be signed (preceded by "+" or "-"). A floating-point literal must have at least one digit and either a decimal point or "e" (or "E").
Boolean	True or False. The possible Boolean values are true and false. These are special values, and are not usable as 1 and 0. In a comparison, any expression that evaluates to 0 is taken to be false, and any statement that evaluates to a number other than 0 is taken to be true.
String	"Hello World!" Strings are delineated by single or double quotation marks. (Use single quotes to type strings that contain quotation marks.)
Object	MyObj = new Object();
Null	Not the same as zero – no value at all. A null value is one that has no value and means nothing.
Undefined	A value that is undefined is a value held by a variable after it has been created, but before a value has been assigned to it.

4.3 STATEMENTS AND OPERATORS

Assignment Operators		
Operator	**Functionality**	**Example/Explanation**
=	Sets one value equal to another	counter=0 Sets the counter to equal the number 0
+=	Shortcut for adding to the current value.	clicks += 2 Sets the variable named counter to equal the current value plus two.
-=	Shortcut for subtracting from the current value.	clicks -= 2 Sets the variable named counter to equal the current value minus two.
*=	Shortcut for multiplying the current value.	clicks *= 2 Sets the variable named counter to equal the current value multiplied by two.
/=	Shortcut for dividing the current value.	clicks /= 2 Sets the variable named counter to equal the current value divided by two.

Comparison Operators		
Description: The comparison operators compare two items and return a value of "true"if the condition evaluates to true, else they return false.		
Operator	**Functionality**	**Example/Explanation**
==	Returns a true value if the items are the same	Counter == 10 Returns the value "true" if the counter's value is currently equal to the number 10
!=	Returns a true value if the items are **not** the same	Counter != 10 Returns the value "true" if the counter's value is any value except the number 10
>	Returns a true value if the item on the left is greater than the item on the right	counter>10 Returns the value "true" if the counter's value is larger than the number 10
>=	Returns a true value if the item on the left is equal to or greater than the item on the right	counter>=10 Returns the value "true" if the counter's value is equal to or larger than the number 10
<	Returns a true value if the item on the left is less than the item on the right	counter<10 Returns the value "true" if the counter's value is smaller than the number 10
<=	Returns a true value if the item on the left is equal to or less than the item on the right	counter<=10 Returns the value "true" if the counter's value is equal to or less than the number 10

Computational Operators		
Description: The computational operators perform a mathematical function on a value or values, and return a single value.		
Operator	**Functionality**	**Example/Explanation**
+	Adds two values together	counter+2 Returns the sum of the counter plus 2
-	Subtracts one value from another	counter-2 Returns the sum of the counter minus 2
*	Multiplies two values	counter*10 Returns the result of the variable times 10
/	Divides the value on the left by the one on the right and returns the result	counter/2 Divides the current value of the counter by 2 and returns the result
++X	Increments the value, and then returns the result	++counter Looks at the current value of the counter, increments it by one, and then returns the result. If the counter has a value of 3, this expression returns the value of 4.
X++	Returns the value, and then increments the value	counter++ Returns the value of the counter, then increments the counter. If the counter has a value of 3, this expression returns the value of 3, then sets the counter value to 4.
--X	Decreases the value, and then returns the result	--counter Looks at the current value of the counter, decreases it by one, and then returns the result. If the counter has a value of 7, this expression returns the value of 6.
X--	Returns the value, and then decreases the value	counter-- Returns the value of the counter, then decreases the counter value. If the counter has a value of 7, this expression returns the value of 7, then sets the counter value to 6.

Logical Operators
Description: The logical operators evaluate expressions and then return a true or false value based on the result.

Operator	Functionality	Example/ Explanation
&&	Looks at two expressions and returns a value of "true" if the expressions on the left and right of the operator are both true	If day = 'friday' && date=13 then alert("Are You Superstitious?") Compares the value of the day and the value of the date. If it is true that today is a Friday **and** if it is also true that the date is the 13th, then an alert box pops up with the message "Are You Superstitious?"
||	Looks at two expressions and returns a value of "true" if either one -- but not both -- of the expressions are true.	if day='friday'&&date=13 then alert("Are You Superstitious?") else if day='friday'||date=13 then alert("Aren't you glad it isn't Friday the 13th?") Compares the value of the day and the value of the date. If it is true that today is a Friday **and** if it is also true that the date is the 13th, then an alert box pops up with the message "Are You Superstitious?" If both are not true, the script moves onto the next line of code... Which compares the value of the day and the value of the date. If either one -- but not both -- is true, then an alert box pops up with the message "Aren't you glad it isn't Friday the 13th?"

4.4 CONTROL STRUCTURES

JavaScript supports the usual control structures:

- The conditionals if, if...else, and switch;
- The iterations for, while, do...while, break, and continue;

4.4.1 Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have three conditional statements:

- if statement - use this statement if you want to execute a set of code when a condition is true
- if...else statement - use this statement if you want to select one of two sets of code to execute
- switch statement - use this statement if you want to select one of many sets of code to execute

```
if( myVariable == 2 ) {
   myVariable = 1;
} else {
   myVariable = 0;
}
```

If the value of myVariable in Figure 4.1 is 2 then the first condition evaluates to true and the value of myVariable is set to 1. If it is anything other than 2 then the else part gets executed.
Now let us see an example of a nested if statement in Figure 4.2.

```
if ( myVariable == 2 ) {
   myVariable = 1;
```

```
} else {
   If (myVariable == 5 ) {
      myVariable = 3;
   } else {
      myVariable = 4;
   }
}
```

Switch Statement

If there exist multiple conditions, the switch statement is recommended. This is because only one expression gets evaluated based on which control directly jumps to the respective case.

```
switch(myVar) {
   case 1:
      //if myVar is 1 this is executed
   case 'sample':
      //if myVar is 'sample' (or 1, see the next paragraph)
      //this is executed
   case false:
      //if myVar is false (or 1 or 'sample', see the next paragraph)
      //this is executed
   default:
      //if myVar does not satisfy any case, (or if it is
      //1 or 'sample' or false, see the next paragraph)
      //this is executed
}
```

As shown in Figure 4.3, depending on the value of "myvar", the statement of the respective case gets executed. If a case is satisfied, the code beyond that case will also be executed unless the break statement is used. In the above example, if myVar is 1, the code for case 'sample', case 'false' and 'default' will all be executed as well.

Comment: Please be consistent. If one value is in quotes, it is better to put the others in quotes as well.

4.4.2 Loop Statements

A loop is a set of commands that executes repeatedly until a specified condition is met. JavaScript supports two loop statements: for and while. In addition, you can use the break and continue statements within loop statements. Another statement, for...in, executes statements repeatedly but is used for object manipulation.

- **For Statement**

A for loop repeats until a specified condition evaluates to false. The JavaScript `for` loop is similar to the Java and C for loops. A for statement looks as follows:

Comment: Please be careful about upper and lower case – especially in JavaScript.

```
for ([initial-expression]; [condition]; [increment-expression]) {
  Statements
}
```

When a for loop executes, the following sequence of operations occur:

1. The initializing expression *initial-expression*, if any, is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.
2. The *condition* expression is evaluated. If the value of *condition* is true, the loop statements execute. If the value of *condition* is false, the loop terminates.
3. The update expression *increment-expression* executes.
4. The *statements* get executed, and control returns to step 2. Actually the syntax

provides for a single statement; when enclosed in braces '{' and '}', any number of statements are treated as a single statement.

The following function contains a for loop that counts the number of selected options in a scrolling list (a *select* object that allows multiple selections). The for loop declares the variable *i* and initializes it to zero. It checks that *i* is less than the number of options in the *select* object, performs the succeeding if statement, and increments *i* by one after each pass through the loop.

```
<HTML>
<HEAD>
<TITLE>IGNOU </TITLE>
<SCRIPT LANGUAGE = "JavaScript">
function howMany(selectObject) {
        var numberSelected=0
        for (var i=0; i < selectObject.options.length; i++) {
                if (selectObject.options[i].selected==true)
                        numberSelected++
        }
        return numberSelected
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="selectForm">
    <P><B>Choose some music types, then click the button below:</B>
    <BR><SELECT NAME="musicTypes" MULTIPLE>
        <OPTION SELECTED> R&B
        <OPTION> Jazz
        <OPTION> Blues
        <OPTION> New Age
        <OPTION> Classical
        <OPTION> Opera
    </SELECT>
    <P><INPUT TYPE="button" VALUE="How many are selected?"
    onClick="alert ('Number of options selected: ' +
    howMany(document.selectForm.musicTypes))">
</FORM>
</BODY>
</HTML>
```

- **While Statement**

The while statement defines a loop that iterates as long as a condition remains true. In the following example the control waits until the value of a text field becomes "go":

while (Document.Form1.Text1.Value != "go") {*Statements* }

In a while loop the condition is evaluated first before executing the statements.

- **For In Statement**

This is a different type of loop, used to iterate through the properties of an object or the elements of an array. For example consider the following statement that loops through the properties of the Scores object, using the variable x to hold each property in turn:

For (x in Scores) {Statements}

- **Break Statement**

The break statement is used for terminating the current While or For loop and then transferring program control to the statement just after the terminated loop. The

following function has a break statement that terminates the **while** loop when *i* becomes equal to 3, and then returns the value 3 * *x*.

```
function testBreak(x) {
        var i = 0
        while (i < 6) {
                if (i == 3)
                        break
                i++
        }
        return i*x
}
```

Comment: The example could be slightly more realistic – why would one want to terminate at 3?

- **Continue Statement**

A continue statement terminates execution of the block of statements in a while or for loop and continues execution of the loop with the next iteration. In contrast to the break statement, continue does not terminate the execution of the loop entirely. Instead,

- In a while loop, it jumps back to the *condition*.
- In a for loop, it jumps back to the *increment-expression*.

The following example shows a While loop with a continue statement that executes when the value of *i* becomes equal to three. Thus, *n* takes on the values one, three, seven, and twelve.

```
i = 0
n = 0
while (i < 5) {
        i++
        if (i == 3) continue
        n += i
}
```

4.5 OBJECT-BASED PROGRAMMING

JavaScript is a very powerful object-based (or prototype-based) language. JavaScript is not a full-blown OOP (Object-Oriented Programming) language, such as Java, but it is an object-based language. Objects not only help you better understand how JavaScript works, but in large scripts, you can create self-contained JavaScript objects, rather than the procedural code you may be using now. This also allows you to reuse code more often.

4.5.1 Functions

Functions are the central working units of JavaScript. Almost all the scripting code uses one or more functions to get the desired result. If you want your page to provide certain a user-defined functionality, then functions are a convenient way of doing so. Therefore it is important that you understand what a function is and how it works.

First let us understand the basic syntax of a function; then we look at how to call it. After that you must know how to pass arguments and why you need to do this. Finally, you have to know how to return a value from a function. The following code

shows the implementation of a function.

```
function example(a,b)
{
        number += a;
        alert('You have chosen: ' + b);
}
```

The function made above can be called using the following syntax.
Example(1,'house')

In fact, when you define the function Example, you create a new JavaScript command that you can call from anywhere on the page. Whenever you call it, the JavaScript code inside the curly brackets {} is executed.

Calling the Function

You can call the function from any other place in your JavaScript code. After the function is executed, the control goes back to the other script that called it.
alert('Example 1: the House');
example(1,'house');
(write more code)

So this script first generates an alert box, then calls the function and after the function is finished it continues to execute the rest of the instructions in the calling code.

Arguments

You can pass arguments to a function. These are variables, either numbers or strings, which are used inside the function. Of course the output of the function depends on the arguments you give it.

In the following example we pass two arguments, the number 1 and the string 'house':
example(1,'house');

When these arguments arrive at the function, they are stored in two variables, a and b. You have to declare these in the function header, as you can see below.
function example(a,b)

```
<HTML>
  <HEAD>
    <TITLE>IGNOU </TITLE>
    <SCRIPT Language = "JavaScript">
      function example(a, b)
      {
            var number;
            number += a ;
            alert('You have chosen: ' + b);
                  }
    </SCRIPT>
  </HEAD>
  <BODY>
  <FORM NAME="selectForm">
    <P><B>Click the button below:</B>
    <BR>
    <P><INPUT TYPE="button" VALUE="Click" onClick="example(1,'house')">
  </FORM>
  </BODY>
```


... [1]

</HTML>

Figure 4.1: Using Functions

It adds 1 to number and displays "You have chosen: house". Of course, if you call the function like example (5,'palace'), then it adds 5 to number and displays "You have chosen: palace". The output of the function depends on the arguments you give it.

Returning a value

One more thing a function can do is to return a value. Suppose we have the following function:

```
<HTML>
  <HEAD>
    <TITLE>IGNOU </TITLE>
    <SCRIPT Language = "JavaScript">
    function calculate(a,b,c)
    {
          d = (a+b) * c;
          return d;
    }
    </SCRIPT>
  </HEAD>
  <BODY>
  <SCRIPT Language = "JavaScript">
    var x = calculate(4,5,9);
    var y = calculate((x/3),3,5);
    alert('calculate(4,5,9) = ' +x + ' and ' + ' calculate((x/3),3,5) = ' + y);
  </SCRIPT>
  </BODY>
</HTML>
```

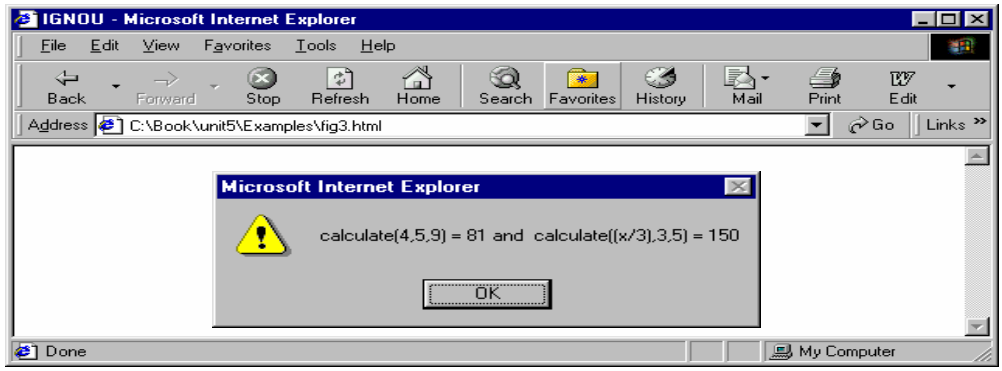


Figure 4.2: Using a Function that Returns a Value

The function shown in Figure 4.8 calculates a number from the numbers you pass to it. When it is done it returns the result of the calculation. The function passes the result back to the function that called it. When the function executes the return statement, control goes back to the calling program without executing any more code in the function, if there is any.

The calling of the function is done using the following two statements in the figure:

```
var x = calculate(4,5,9);
var y = calculate((x/3),3,5);
```

It means that you have declared a variable x and are telling JavaScript to execute calculate( ) with the arguments 4, 5 and 9 and to put the returned value (81) in x. Then you declare a variable y and execute calculate( ) again. The first argument is x/3, which means 81/3 = 27, so y becomes 150.Of course you can also return strings or even Boolean values (true or false). When using JavaScript in forms, you can write a function that returns either true or false and thus tells the browser whether to submit a form or not.

4.5.2 Executing Deferred Scripts

Deferred scripts do not do anything immediately. In order to use deferred commands, you must call them from outside the deferred script. There are three ways to call deferred scripts:

- From immediate scripts, using the function mechanism
- By user-initiated events, using event handlers
- By clicking on links or image-map zones that are associated with the script

Calling Deferred Code from a Script

A function is a deferred script because it does not do anything until an event, a function, a JavaScript link, or an immediate script calls it. You have probably noticed that you can call a function from within a script. Sometimes you are interested in calling a function from the same script, and in other cases you might want to call it from another script. Both of these are possible.

Calling a function from the same script is very simple. You just need to specify the name of the function, as demonstrated in Figure 4.9.

```
<HTML>
<HEAD>
<TITLE>Calling deferred code from its own script</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
makeLine(30)
function makeLine(lineWidth) {
  document.write("<HR SIZE=" + lineWidth + ">")
}
makeLine(10)
// -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

4.5.3 Objects

A JavaScript object is an instance of datatype. Object is given a unique name and the collection of properties of the corresponding object may be accessed using the dot syntax. As a quick introduction to the concept of JavaScript objects, here is the code that creates an instance of an object called myObj:

```
var myObj = new Object();
myObj.business = "Voice and Data Networking"; myObj.CEO = "IGNOU";
myObj.ticker = "CSCO";
```

Comment: Please put in a short description of the contents of myObj;

After having run that code (in a scope situation that allowed myObj to be accessed as required), you could run this...

```
document.write("My Object ticker symbol is " + myObj.ticker + ".");
```

...and get a complete sentence in your HTML document.

- **Document Object**

The document object is a property of the window object. This object is the container for all HTML HEAD and BODY objects associated within the HTML tags of an HTML document

Document Object Properties

Property	Description
AlinkColor	The color of active links
BgColor	The background color of the web page. It is set in the <BODY> tag. The following code sets the background color to white. document.bgColor = "#FFFFFF"
Cookie	Used to identify the value of a cookie
Domain	The domain name of the document server
Embeds	An array containing all the plugins in a document
FgColor	The text color attribute set in the <body> tag
FileCreatedDate	Use this value to show when the loaded HTML file was created
fileModifiedDate	Use this value to show the last change date of the HTML file currently loaded
lastModified	The date the file was modified last
Layers	An array containing all the layers in a document
LinkColor	The color of HTML links in the document. It is specified in the <BODY> tag.
Title	The name of the current document as described between the header <TITLE> tags.
URL	The location of the current document
VlinkColor	The color of visited links as specified in the <BODY> tag

Document Object Methods

Method	Description
Clear	This is depreciated
Close	Closes an output stream that was used to create a document object
contextual	It can be used to specify style of specific tags. The following example specified that text in blockquotes is to be blue: document.contextual(document.tags.blockquote).color = "blue"; Multiple styles may be specified in the contextual method to set the value of text in a <H3> tag that is underlined to the color blue, for example. document.contextual(document.tags.H3, document.tags.U).color = "blue";
elementFromPoint(x, y)	Returns the object at point x, y in the HTML document.
getSelection	Get the selected text (if any is selected)

open([mimeType])	Opens a new document object with the optional mime type.
write(expr1[,expr2...exprN])	Add data to a document. Writes the values passed to the write function to the document. For example document.write("<H3>") document.writeln("This is a Header") document.write("</H3>")
writeln(expr1[,expr2...exprN])	Adds the passed values to the document appended with a new line character.

- **Predefined Objects**

Let us consider some of the most frequently used predefined objects provided in Javascript.

- **Math object**

In most applications we need to perform calculations, whether it is accounting software or scientific software. Programmers are often typecast as good mathematicians. Every mathematician needs a calculator sometimes, or in the case of JavaScript, the Math object. If we want to calculate "2.5 to the power of 8" or "Sin0.9" in your script, then JavaScript's virtual calculator is what you need. The Math object contains a number of manipulating functions:

The Math object

Methods	Description
Math.abs(x)	Return absolute value of x
Math.acos(x)	Return arc cosine of x in radians
Math.asin(x)	Return arc sine of x in radians
Math.atan(x)	Return arc tan of x in radians
Math.atan2(x, y)	Counterclockwise angle between x axis and point (x,y)
Math.ceil(x)	Rounds a number up
Math.cos(x)	Trigonometric cosine of x (x in radians)
Math.exp(x)	Exponential method e^x
Math.floor(x)	Rounds a number down
Math.log(x)	Natural logarithm of x (base e)
Math.max(a, b)	Returns the larger of two values
Math.min(a, b)	Returns the smaller of two values
Math.pow(x, y)	Returns x^y
Math.round(x)	Rounds x to the closest integer
Math.sin(x)	Trigonometric sine of x (x in radians)
Math.sqrt(x)	Square root of x
Math.tan(x)	Trigonometric tangent of x (x in radians)
Properties	**Description**
Math.E	Euler's constant (~ 2.718)
Math.LN10	Natural logarithm of 10 (~ 2.302)
Math.LN2	Natural logarithm of 2 (~ 0.693)
Math.LOG10E	Base 10 logarithm of Euler's constant (~ 0.0434)
Math.LOG2E	Base 2 logarithm of Euler's constant (~1.442)
Math.PI	The ratio of a circle's circumference to its diameter (~3.141)
Math.SQRT1_2	Square root of 0.5 (~ 0.707)
Math.SQRT2	Square root of 2.0 (~ 1.414)

Let us have Javascript perform some mathematical calculations:

```
//calculate e^5
Math.exp(5)
//calculate cos(2PI)
Math.cos(2*Mat )
```

Please put statements everywhere

The "with" statement

If you intend to invoke Math multiple times in your script, a good statement to remember is "with." Using it you can omit the "Math." prefix for any subsequent Math properties/methods:

```
with (Math){
var x= sin(3.5)
var y=tan(5)
var result=max(x,y)
}
```

- **Date Object**

The Date object is used to work with dates and times.

Creating a Date Instance

You should create an instance of the Date object with the "new" keyword. The following line of code stores the current date in a variable called "my_date":

var my_date=new Date( )

After creating an instance of the Date object, you can access all the methods of the object from the "my_date" variable. If, for example, you want to return the date (from 1-31) of a Date object, you should write the following:

my_date.getDate( )

You can also write a date inside the parentheses of the Date( ) object. The following line of code shows some of the date formats available.

new Date("Month dd, yyyy hh:mm:ss"), new Date("Month dd, yyyy"), new Date(yy,mm,dd,hh,mm,ss), new Date(yy,mm,dd), new Date(milliseconds)

Here is how you can create a Date object for each of the ways above:

var my_date=new Date("October 12, 1988 13:14:00"), var my_date=new Date("October 12, 1988"), var my_date=new Date(88,09,12,13,14,00), var my_date=new Date(88,09,12), var my_date=new Date(500)

- **Some Date Methods**

Methods	Explanation
Date( )	Returns a Date object
GetDate( )	Returns the date of a Date object (from 1-31)
GetDay( )	Returns the day of a Date object (from 0-6. 0=Sunday, 1=Monday, etc.)
GetMonth( )	Returns the month of a Date object (from 0-11. 0=January, 1=February, etc.)
GetFullYear( )	Returns the year of the Date object (four digits)
GetHours( )	Returns the hour of the Date object (from 0-23)
GetMinutes( )	Returns the minute of the Date object (from 0-59)
GetSeconds( )	Returns the second of the Date object (from 0-59)

Examples:
Date

Returns the current date, including date, month, and year. Note that the getMonth method returns 0 in January, 1 in February etc. So add 1 to the getMonth method to display the correct date.

```
<HTML>
<BODY>
<SCRIPT LANGUAGE="JAVASCRIPT">
var d = new Date()
document.write("date = ")
document.write(d.getDate( ))
document.write(".")
document.write(d.getMonth() + 1)
document.write(".")
document.write(d.getFullYear())
document.write("time = ")
document.write(d.getHours())
document.write(".")
document.write(d.getMinutes() + 1)
document.write(".")
document.write(d.getSeconds())
</SCRIPT>
</BODY>
</HTML>
```

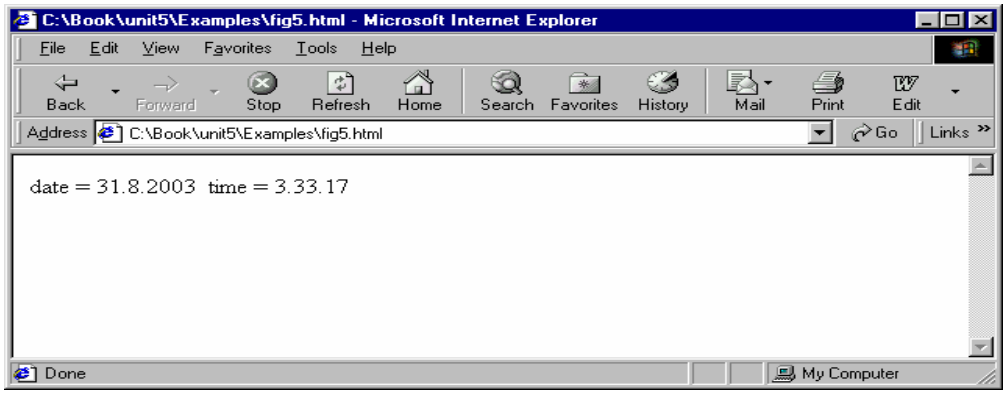


Figure 4.3: Using the Date Object

Time

Returns the current time for the timezone in hours, minutes, and seconds as shown in Figure 5.11. To return the time in GMT use getUTCHours, getUTCMinutes etc.

- **Array Object**

An Array object is used to store a set of values in a single variable name. Each value is an element of the array and has an associated index number. You can refer to a particular element in the array by using the name of the array and the index number. The index number starts at zero.

You create an instance of the Array object with the "new" keyword.

var family_names=new Array(5)

The expected number of elements goes inside the parentheses, in this case it is 5.

You assign data to each of the elements in the array like this:

family_names[0]="Sharma" family_names[1]="Singh" family_names[2]="Gill" family_names[3]="Kumar" family_names[4]="Khan"

The data can be retrieved from any element by using the index of the array element you want:

Mother=family_names[0] father=family_names[1]

The Most Commonly Used Array Methods

Methods	Explanation
Length	Returns the number of elements in an array. This property is assigned a value when an array is created
reverse( )	Returns the array reversed
slice( )	Returns a specified part of the array
Sort( )	Returns a sorted array

- **History Object**

The History object is a predefined JavaScript object which is accessible through the history property of a window object. The window.history property is an array of URL strings, which reflect the entries in the History object. The History object consists of an array of URLs, accessible through the browser's Go menu, which the client has visited within a window. It is possible to change a window's current URL without an entry being made in the **History** object by using the location.replace method. The History object contains 4 properties and 3 methods as summarized here:

Property	Summary for History Object
Current	Specifies the URL of the current history entry.
Next	Specifies the URL of the next history entry.
Previous	Specifies the URL of the previous history entry.
Length	Reflects the number of entries in the history list.

Method	Summary for History Object
Back( )	Loads the previous URL in the history list.
Forward( )	Loads the next URL in the history list.
Go( )	Loads a URL from the history list.

- **Location Object**

The Location object is part of a Window object and is accessed through the window.location property. It contains the complete URL of a given Window object, or, if none is specified, of the current Window object. Syntax **:** All of its properties are strings representing different portions of the URL, which generally takes the following form:

<protocol>//<host>[:<port>]/<pathname>[<hash>][<search>]
You can create a Location object by simply assigning a URL to the location property of an object:

Syntax**:** window.location = "file:///C:/Projects"

Comment: Throughout this chapter there has been little regard for the correctness of the case of a letter. Please ensure that the correct case is used unless the case is not significant. In Javascript, it usually is significant.

Property	Description
Hash	The **hash** property is a string beginning with a hash (#), that specifies an anchor name in an HTTP URL
Host	The **host** property is a string comprising the **hostname** and **port** strings.
hostname	The **hostname** property specifies the server name, subdomain and domain name (or IP address) of a URL.
Href	The **href** property is a string specifying the entire URL, and of which all other **link** properties are substrings.
pathname	The **pathname** property is a string portion of a URL specifying how a particular resource can be accessed.
Port	The **port** property is a string specifying the communications port that the server uses.
protocol	The **protocol** property is the string at the beginning of a URL, up to and I ncluding the first colon (:), which specifies the method of access to the URL.

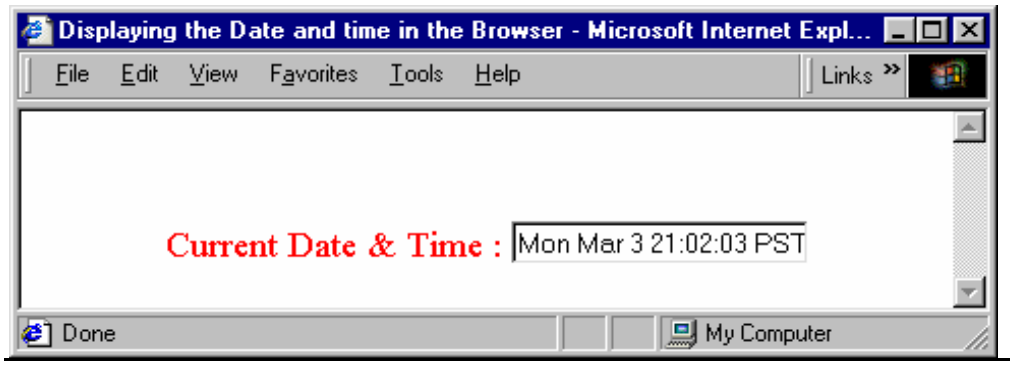
Property	Description
search	The **search** property is a string beginning with a question mark that specifies any query information in an HTTP URL.

Some Common Methods

Method	Description
reload	The reload method forces a reload of the windows current document, i.e. the one contained in the Location.href
Replace	The replace method replaces the current history entry with the specified URL. After calling the replace method to change the history entry, you cannot navigate back to the previous URL using the browser's Back b[illegible]n.

Check Your Progress 1

1. Write a JavaScript code block using arrays and generate the current date in words. This should include the day, the month and the year.
2. Write JavaScript code that converts the entered text to upper case.
3. Write JavaScript code to validate Username and Password. Username and Password are stored in variables.
4. Design the following Web page



4.6 MESSAGEBOX IN JAVASCRIPT

In this section, we cover the topic of dialog boxes. This topic is important for understanding the way parameters are passed between different windows. This mechanism is a key component of some of the new capabilities of Internet Explorer 5.5 and 6.0, such as print templates. You use dialog boxes all over. The alert box is probably the most popular one.

4.6.1 Dialog Boxes

In this the user cannot switch to the window without closing the current window. This kind of dialog box is referred to as a modal dialog box. You create a modal dialog box with showModalDialog().

Syntax: showModalDialog (“Message”)

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JAVASCRIPT">

function fnOpenModal(){
  window.showModalDialog("test.htm")
}
```

```
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME = IGNOU>
<INPUT TYPE="button" VALUE="Push Me" onclick="fnOpenModal()">
</BODY>
</HTML>
```

4.6.2 Alert Boxes

Alert boxes can be utilized for a variety of things, such as to display when an input field has not been entered properly, to display a message on document open or close, or to inform someone that they have clicked a link to leave your site. Whatever the use, the construction is essentially the same.
Syntax : alert(“message”)

The following example will generate a simple alert box based on clicking either a link or a form button.

```
<html>
<title>Codeave.com(JavaScript: Alert Box)</title>
<body bgcolor="#ffffff">
<!-- Example of a form button that will open an alert box -->
<form>
<input type="button" value="Open Alert Box"
onClick='alert("Place your message here... \n Click OK to continue.")'>
</form>
<p>
<!-- Example of a link that will open an alert box -->
<a href='javascript:onClick=alert("Place your message here... \n Click OK to
continue.")'>
Open Alert Box</a>

</body>
</html>
```

Figure 4.4: Using an Alert Box

4.6.3 Confirm Boxes

The JavaScript confirm alert box differs from a regular alert box in that it provides two choices to the user, OK and Cancel. Typically, you’ll see confirmation boxes utilized on links where the destination is outside the domain of the page you are currently on or to alert you of potentially objectionable material. The following example will display a confirmation alert box when either a link or form button is clicked. Once either OK or Cancel are selected an alert box will follow to display which was chosen.

```
<html>
```

```
<body bgcolor="#FFFFFF">
<title>CodeAve.com(JavaScript: Confirm Alert Box)</title>
<script language="JavaScript">
<!--
function confirm_entry()
        {
        input_box=confirm("Click OK or Cancel to Continue");
        if (input_box==true)
                {
                // Output when OK is clicked
                alert ("You clicked OK");
                }
        else
                {
                // Output when Cancel is clicked
                alert ("You clicked cancel");
                }
        }
-->
</script>
Click <a href="JavaScript:confirm_entry()">here</a>
<p>
<form onSubmit="confirm_entry()">
<input type="submit" >
</form>
</body>
</html>
```

4.6.4 Prompt Boxes

The prompt box allows the user to enter information. The benefits of using a prompt are fairly limited and the use of forms would often be preferred (from a user perspective).
The prompt box title text cannot be changed and the same applies to button text. You can have 2 lines of text using \n where the new line starts (please note that the opera browser up to version 7 will only display 1 line of text)

The text entry field is similar to a form input type="text". The 2 buttons are OK and Cancel. The value returned is either the text entered or null.

The syntax for the prompt is

prompt("your message",""); (script tags omitted)
"your message","" the ,"" holds the default text value "" = empty.

4.7 JAVASCRIPT WITH HTML

4.7.1 Events

Events are actions that can be detected by JavaScript. An example would be the onMouseOver event, which is detected when the user moves the mouse over an object. Another event is the onLoad event, which is detected as soon as the page is finished loading. Usually, events are used in combination with functions, so that the function is called when the event happens. An example would be a function that would animate a button. The function simply shifts two images. One image that shows the button in an "up" position, and another image that shows the button in a "down" position. If this function is called using an onMouseOver event, it will make it look as if the button is pressed down when the mouse is moved over the image.

4.7.2 Event Handlers

An event handler executes a segment of a code based on certain events occurring within the application, such as onLoad or onClick. JavaScript event handlers can be divided into two parts: interactive event handlers and non-interactive event handlers. An interactive event handler is the one that depends on user interaction with the form or the document.

For example, onMouseOver is an interactive event handler because it depends on the user's action with the mouse. An example of a non-interactive event handler is onLoad, as it automatically executes JavaScript code without the need for user interaction. Here are all the event handlers available in JavaScript.

- **onLoad and onUnload**

onLoad and onUnload are mainly used for popups that appear when the user enters or leaves the page. Another important use is in combination with cookies that should be set upon arrival or when leaving your pages.

For example, you might have a popup asking the user to enter his name upon his first arrival to your page. The name is then stored in a cookie. Furthermore, when the visitor leaves your page a cookie stores the current date. Next time the visitor arrives at your page, it will have another popup saying something like: "Welcome Ajay, this page has not been updated since your last visit 8 days ago".

Another common use of the onLoad and onUnload events is in making annoying pages that immediately open several other windows as soon as you enter the page.

- **OnFocus and onBlur**

The onFocus event handler executes the specified JavaScript code or function on the occurrence of a focus event. This is when a window, frame or form element is given the focus. This can be caused by the user clicking on the current window, frame or form element, by using the TAB key to cycle through the various elements on screen, or by a call to the window.focus method. Be aware that assigning an alert box to an object's onFocus event handler will result in recurrent alerts as pressing the 'OK' button in the alert box will return focus to the calling element or object.

The onFocus event handler uses the Event object properties.

type - this property indicates the type of event.
target - this property indicates the object to which the event was originally sent.

The following example shows the use of the onFocus event handler to replace the default string displayed in the text box. Note that the first line is HTML code and that the text box resides on a form called 'myForm'.

Syntax : onfocus = script

```
 <HTML>
<HEAD>

</HEAD>
<BODY>
<FORM NAME = "myform" >
<input type="text" name="myText" value="Give me focus" onFocus =
"changeVal()">
</BODY>
```

```
<SCRIPT LANGUAGE="JAVASCRIPT">

s1 = new String(myform.myText.value)
function changeVal() {
s1 = "I'm feeling focused"
document.myform.myText.value = s1.toUpperCase()
}
</SCRIPT>
</HTML>
```

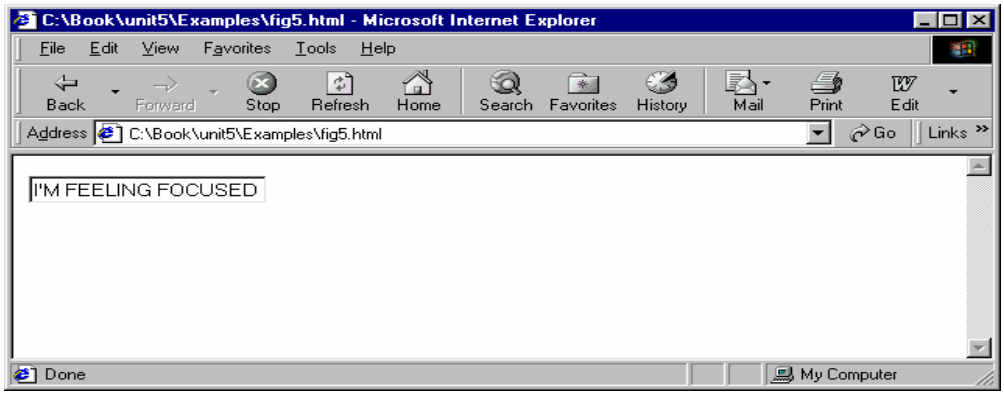


Figure 4.5: Using the onFocus Method

onblur = script

The onBlur event handler executes the specified JavaScript code or function on the occurrence of a blur event. This is when a window, frame or form element loses focus. This can be caused by the user clicking outside of the current window, frame or form element, by using the TAB key to cycle through the various elements on screen, or by a call to the window.blur method.

The onBlur event handler uses the Event object properties.

type - this property indicates the type of event.
target - this property indicates the object to which the event was originally sent.

The following example shows the use of the onBlur event handler to ask the user to check that the details given are correct. Note that the first line is HTML code.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM NAME = "myform" >
Enter email address <INPUT TYPE="text" VALUE="" NAME="userEmail"
onBlur=addCheck( )>
</BODY>
<SCRIPT LANGUAGE="JAVASCRIPT">
function addCheck() {
alert("Please check your email details are correct before submitting")
}
</SCRIPT>
</HTML>
```

- **OnError**

The onError event handler executes the specified JavaScript code or function on the occurrence of an error event. This happens when an image or document causes an error during loading. A distinction must be made between a browser error, when the user types in a non-existent URL, for example, and a JavaScript runtime or syntax error. This event handler will only be triggered by a JavaScript error, not a browser error.

Apart from the onError handler triggering a JavaScript function, it can also be set to onError="null", which suppresses the standard JavaScript error dialog boxes. To suppress JavaScript error dialogs when calling a function using onError, the function must return true (Example 2 below demonstrates this).

There are two things to bear in mind when using window.onerror. First, this only applies to the window containing window.onerror, not any others, and secondly, window.onerror must be spelt all lower-case and contained within <script> tags; it cannot be defined in HTML (this obviously does not apply when using onError with an image tag, as in example 1 below).

The onFocus event handler uses the Event object properties.

type - this property indicates the type of event.
target - this property indicates the object to which the event was originally sent.

The first example suppresses the normal JavaScript error dialogs if a problem arises when trying to load the specified image, while Example 2 does the same, but applied to a window, by using a return value of true in the called function, and displays a customized message instead.

Syntax : Object.onError = [function name]

Example 1

```
<IMG NAME="imgFaculty" SRC="dodgy.jpg onError="null">
```

Example 2

```
<script type="text/javascript" language="JavaScript">
s1 = new String(myForm.myText.value)
window.onerror=myErrorHandler
function myErrorHandler() {
alert('A customized error message')
return true
}
</script>
<body onload=nonexistantFunc( )>
```

Check Your Progress 2

Design a Web page that displays a welcome message whenever the page is loaded and an Exit message whenever the page is unloaded.

4.8 FORMS

Each form in a document creates a form object. Since a document can contain more than one form, Form objects are stored in an array called forms.

4.8.1 Forms Array

Using the forms[ ] array we access each Form object in turn and show the value of its name property in a message box. Let us have a look at an example that uses the forms array. Here we have a page with three forms on it.

```
<HTML>
<HEAD>
   <SCRIPT LANGUAGE=JavaScript>
   function window_onload()
   {
     var numberForms = document.forms.length;
     var formIndex;
     for (formIndex = 0; formIndex < numberForms; formIndex++)
     {
       alert(document.forms[formIndex].name);
     }
   }
   </SCRIPT>
</HEAD>
<BODY LANGUAGE=JavaScript onLoad="window_onload()">
    <FORM NAME="form1">
          <P>This is inside form1</P>
    </FORM>
    <FORM NAME="form2">
          <P>This is inside form2</P>
    </FORM>
    <FORM NAME="form3">
          <P>This is inside form3</P>
    </FORM>
</BODY>
</HTML>
```

Within the body of the page we define three forms. Each form is given a name, and contains a paragraph of text. Within the definition of the <BODY> tag, the window_onload( ) function is connected to the window object's onLoad event handler.

```
<BODY LANGUAGE=JavaScript onLoad="return
   window_onload( )">
```

This means that when the page is loaded, our window_onload() function will be called. The window_onload( ) function is defined in a script block in the HEAD of the page. Within this function we loop through the forms[ ] array. Just like any other JavaScript array, the forms[ ] array has a length property, which we can use to determine how many times we need to loop. Actually, as we know how many forms there are, we could just write the number in. However, here we are also demonstrating the length property, since it is then easier to add to the array without having to change the function. Generalizing your code like this is a good practice to follow.

The function starts by getting the number of Form objects within the forms array and stores it in the variable numberForms.

```
function window_onload( )
{
  var numberForms = document.forms.length;
```

Next we define a variable, formIndex, to be used in our for loop. After this comes the for loop itself.

```
var formIndex;
for (formIndex = 0; formIndex < numberForms; formIndex++)
{
  alert(document.forms[formIndex].name);
}
```

Remember that since the indices for arrays start at zero, our loop needs to go from an index of 0 to an index of numberForms - 1. We do this by initializing the formIndex variable to zero, and setting the condition of the for loop to formIndex < numberForms.

Within the for loop's code, we pass the index of the desired form (that is, formIndex) to document.forms[ ], which gives us the Form object at that array index in the forms array. To access the Form object's name property, we put a dot at the end and the name of the property, name.

- **Form Object**

Form is a property of the document object. This corresponds to an HTML input form constructed with the FORM tag. A form can be submitted by calling the JavaScript submit method or clicking the form SUBMIT button. Some of the form properties are:

- Action - This specifies the URL and CGI script file name the form is to be submitted to. It allows reading or changing the ACTION attribute of the HTML FORM tag.
- Button – An object representing a GUI control.
- Elements - An array of fields and elements in the form.
- Encoding - This is a read or write string. It specifies the encoding method the form data is encoded in, before being submitted to the server. It corresponds to the ENCTYPE attribute of the FORM tag. The default is "application/x-www-form-urlencoded". Other encoding includes text/plain or multipart/form- data.
- Length - The number of fields in the elements array, that is, the length of the elements array.
- Method - This is a read or write string. It has the value "GET" or "POST".
- Name - The form name. Corresponds to the FORM Name attribute.
- Password – An object representing a password field.
- Radio – An object representing a radio button field.
- Reset - An object representing a reset button.
- Select - An object representing a selection list.
- Submit - An object representing a submit button.
- Target - The name of the frame or window to which the form submission response is sent by the server.
  Corresponds to the FORM TARGET attribute.
- Text - An object representing a text field.
- Textarea - An object representing a text area field.

Some of the **Form Element Properties** are:

- Name – It provides access to an element’s name attribute. It applies to all form elements.
- Type – Identifies the object’s type. It applies to all form elements.
- Value - Identifies the object’s value. It applies to all, button, checkbox, hidden, password, radio, reset, submit, text or textarea.
- Checked - Identifies whether the element is checked. It applies to checkbox and radio.
- Default checked - Identifies whether the element is checked by default. It applies

to checkbox and radio.

- Default value – Identifies the object's default value. It applies to password, submit and textarea.
- Length - Identifies the length of a select list. It applies to select.
- Options – An array that identifies the options supported by the select list. It applies to select.

Syntax : <FORM Name = "myform" Action = "mailto:subscribe@abc.com" Method = POST Enctype = "multipart/form-data" onsubmit = "return check( )" >

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE = "javascript">
function check ()
        { if (document.myform.email.value == "") {
                alert("please enter your email-id");
                return false; }
        }
</SCRIPT>
</HEAD>
<BODY>
Enter your Email to subscribe : <p>
<FORM Name = "myform" Action = "mailto:subscribe@abc.com" Method = POST
Enctype = "multipart/form-data" onsubmit = "return check()" >
        <Input type = "text" Name = "email">
        <Input type = "submit" value = "submit">
</FORM>
</BODY>
</HTML>
```

The above code is used for obtaining the user's email-id. In case the user clicks the submit button without entering the email-id in the desired text field, he sees a message box indicating that the email-id has not been entered.

Case Study

Design a Web page with appropriate functionality to accept an order for a fast food outlet. It should check if the user has entered a valid name and email-id. It should also calculate the value of the order.

```
<HTML>
<HEAD>
  <TITLE>Donald Duck</TITLE>
<SCRIPT Language="JavaScript">
var m;
function chk_name()
{
if( document.form1.txt_name.value == "")
{
alert("Please enter your name");
document.form1.txt_name.focus();
}
}
function chk_email()
{
var str = document.form1.txt_email.value ;
var i;
```


... [3]

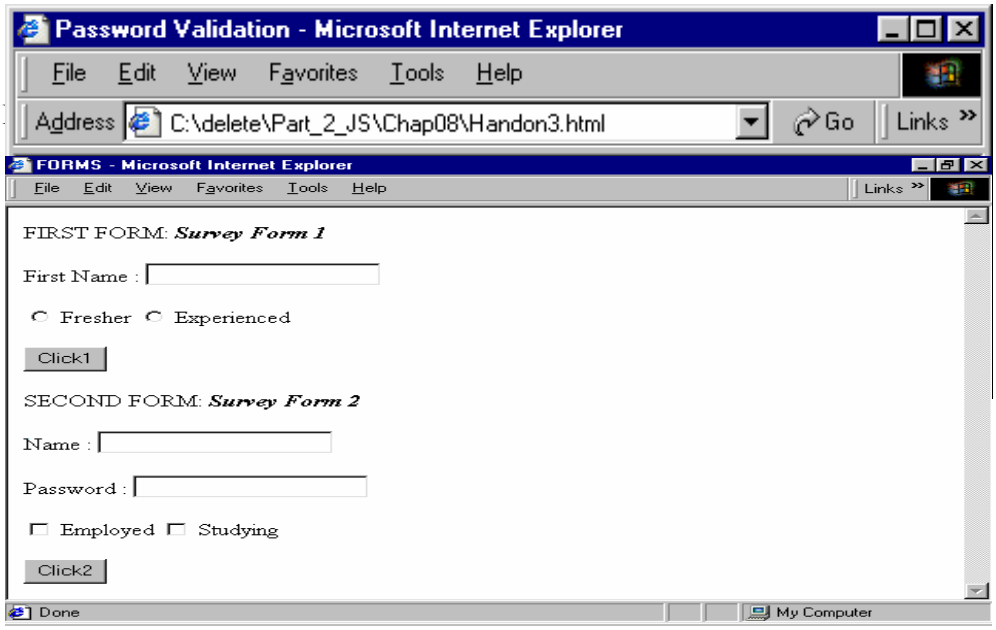
```
if ( document.form1.txt_email.value == "")
{
alert("Please enter your Email-ID");
document.form1.txt_email.focus();
}
i = str.indexOf("@");
if ( i< 0)
{
alert ("Please enter a valid Email-Id");
}
}

function mainitem(F1)
{ var z=" ";
                for(j=0;j<3;j++)
                {
                        for(i=0;i<F1.elements[j].length;i++)
                        {
                                if (F1.elements[j][i].selected)
                                        {
                                        var y=F1.elements[j].options[i].value;
                                        z=z+"\n"+y;
                                        F1.elements[3].value=z;
                                        }
                }
                }
        m=z;
}
function cal(F1)
{ var d=0;
        for(j=0;j<3;j++)
        {
                for(i=0;i<F1.elements[j].length;i++)
                {
        if (F1.elements[j][i].selected)
                        {
                        var y=F1.elements[j].options[i].value;
                        s=new String(y);
                        var a=s.indexOf(">");
                        var b=s.substring(a+1,a+3);
        c=parseInt(b);
        d=d+c;
        }
    }
   }
        p="Total cost of the selected items="+d;
        m=m+"\n"+p;
F1.elements[3].value=m;
}

function clr(F1)
{
F1.elements[3].value=" ";
}
</SCRIPT>
</HEAD>
<BODY>
<h2><font color="blue"><center> Welcome to the World renowned online Fast Food
Center </font>
```

```
<font color="red"> Donald Duck ! </center></font></h2>
<Form name="form1" ACTION = "mailto:dlc@ignou.ac.in" METHOD = POST>
Select the Menu Items of your choice and then click on the Total Cost to find the bill
amount-
<BR><BR>

<Table >
<TR valign=top ><td>
Major dishes :<BR>
<select name="s1" MULTIPLE onBlur="mainitem(this.form)">
<option value="Onion cheese capsicum Pizza->300" selected> Onion cheese
capsicum Pizza
<option value="Onion mushroom Pizza->200"> Onion mushroom Pizza
<option value="Chicken Tikka Pizza->460"> Chicken Tikka Pizza
<option value="Cheese Pizza->150"> Cheese Pizza
</select>
<BR><BR></td>
<td> </td><td> </td>
<td>
Soups :<BR>
<select name="s2" MULTIPLE onBlur="mainitem(this.form)">
<option value="Tomato Soup->70"> Tomato Soup
<option value="Sweet corn Soup->80">Sweet corn Soup
<option value="Sweet n Sour soup->90">Sweet n Sour soup
<option value="Mixed veg soup->50">Mixed veg soup
</select>
<BR><BR></td>
<td> </td><td> </td>
<td>
Miscellaneous :<BR>
<select name="s3" onBlur="mainitem(this.form)">
<option value=" ">'Desserts'
<option value="Milkshakes->35">Milkshakes
<option value="Soft drinks->20">Soft drinks
<option value="Ice cream sodas->25">Softy
</select>
<BR><BR></td>
<td> </td><td> </td>
</TR>
</Table>
<Table>
<TR valign=top><td>
The items selected form the Menu are :
<TEXTAREA Name="TA1" Rows=10 Cols=50>
</TEXTAREA><BR><BR></td>
<td> </td><td> </td>
<td><BR>
<input type="button" Value="Total Cost" onClick="cal(this.form)">
<input type="button" Value="Clear" onClick="clr(this.form)">
</td>
</TR>
</Table>
<HR noshade>
<h2><font color="red"> Personal Details </center></font></h2>
<Table >
<TR valign=top ><td>
Name:<BR>
<Input Type = "Text" Name = "txt_name" Onblur = "chk_name()">
```

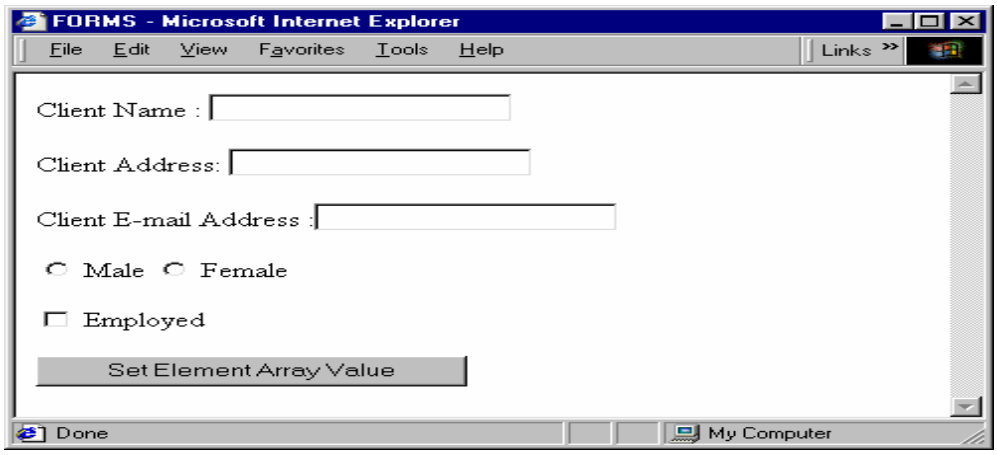
```
<BR><BR></td>
<td> </td><td> </td>
<td>
Contact Address :<br>
<TEXTAREA Name="TA2" Rows=3 Cols=10>
</TEXTAREA>
<BR><BR></td>
<td> </td><td> </td>
<td>
Email :<BR>
<Input Type = "Text" Name = "txt_email" Onblur = "chk_email()">
<BR><BR></td>
<td> </td><td> </td>

</TR>
</Table>
<Table>
<TR valign=top ><td>
Phone :<BR>
<Input Type = "Text" Name = "txt_phone">
<BR><BR></td>
<td> </td><td> </td>
<td>
<BR>
<Input Type = "submit" Name = "btnsubmit" Value = "          Submit          ">
<BR><BR></td>
<td> </td><td> </td>
</TR>
</Table>
</Form>
</BODY>
</HTML>
```

Check Your Progress 3

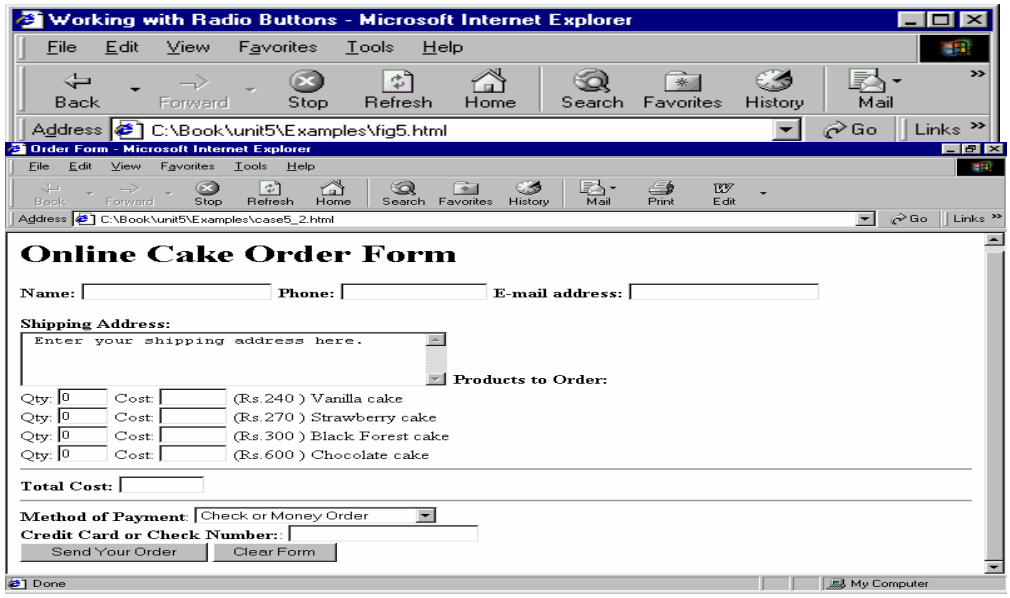
1. Design the following Web page.



2. Design the following Web page.



3. Design the following Web page in such a way that selecting any option from the radio button displays the appropriate result in the Result box.



4. Suppose you have an order form that updates automatically each time you enter or change a quantity, the cost for that item and the total cost are updated. Each field must be validated. Design the following web page. **(Case Study)**

4.9 SUMMARY

In this unit we have learned about the major features of JavaScript. Besides normal programming constructs, datatypes and variables, we have also discussed the most commonly used objects. These are: Document, Math, Date, History, Location etc. The Submit and Reset objects are used to submit the information to the server and reset the information entered, respectively. Finally, we discussed a very important object, the Form object.

4.10 SOLUTIONS/ ANSWERS

Check Your Progress 1

1.

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

```
<TITLE> Date validations </TITLE>
<SCRIPT>
        var monthNames = new Array(12);
        monthNames[0]= "January";
        monthNames[1]= "February";
        monthNames[2]= "March";
        monthNames[3]= "April";
        monthNames[4]= "May";
        monthNames[5]= "June";
        monthNames[6]= "July";
        monthNames[7]= "August";
        monthNames[8]= "September";
        monthNames[9]= "October";
        monthNames[10]= "November";
        monthNames[11]= "December";

        var dayNames = new Array(7);
        dayNames[0]= "Sunday";
        dayNames[1]= "Monday";
        dayNames[2]= "Tuesday";
        dayNames[3]= "Wednesday";
        dayNames[4]= "Thursday";
        dayNames[5]= "Friday";
        dayNames[6]= "Saturday";

        function customDateString (m_date)
        {
                var daywords = dayNames[m_date.getDay()];
                var theday = m_date.getDate();
                var themonth = monthNames[m_date.getMonth()];
                var theyear = m_date.getYear();
                return daywords + ",  " + themonth + " " + theday + ", " + theyear;
        }
</SCRIPT>
</HEAD>

<BODY>
<H1>WELCOME!</H1>
<SCRIPT>
        document.write(customDateString( new Date()))
</SCRIPT>
</BODY>
</HTML>
```

2.

```
<HTML>
<HEAD>
<SCRIPT  language="JavaScript">
        function checkData(column_data)
        {
                if (column_data != "" &&  column_data.value !=
column_data.value.toUpperCase( ))
                {
                        column_data.value = column_data.value.toUpperCase( )
                }
        }
</SCRIPT>
</HEAD>
```

Comment: This works, but what is the second text field for?

```
<BODY>
<FORM>
        <INPUT TYPE="text" NAME="collector" SIZE=10
onChange="checkData(this)">
        <INPUT TYPE="text" NAME="dummy" SIZE=10>
</FORM>
</BODY>
</HTML>
```

3.

```
<HTML>
<HEAD>
<TITLE>Password Validation</TITLE>
<SCRIPT language="JavaScript">
<!--
var userpassword = new Array(4);
userpassword[0]= "Ajay";
userpassword[1]= "ajay";
userpassword[2]= "Rohan";
userpassword[3]= "rohan";

function checkOut()
{
        var flag =0;
        var flag1 =0;
        var i =0;
        var j =0;

for (x=0; x<document.survey.elements.length; x++)
        {
        if (document.survey.elements[x].value == "")
        {
        alert("You forgot one of the required fields. Please try again")
        return;
        }
        }
        var user= document.survey.elements[0].value
        var password = document.survey.elements[1].value
        while(i<=3)
        {
        if(userpassword[i] == user)
        {
                j=i;
                j++;
                if(userpassword[j] == password)
                {
                flag=1;
                break;
                }
        }
        i+=2;
        }
if(flag == 0)
        {
        alert("Please enter a valid user name and password")
        return;
```

```
        }
        else
        {
        alert("Welcome !!\n" +document.forms[0].Username.value);
        }
return;
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM ACTION="" method="POST" NAME="survey"
 onSubmit="return checkOut(this.form)">
<INPUT TYPE="TEXT" NAME="Username" SIZE="15" MAXLENGTH="15">
User Name
<BR><INPUT TYPE="PASSWORD" NAME="Pasword" SIZE="15">Password
<BR><INPUT TYPE="SUBMIT" VALUE="Submit"><INPUT TYPE="RESET"
VALUE="Start Over">
</FORM>
</BODY>
</HTML>
```

4.

```
<HTML>
<HEAD>
<TITLE>Displaying the Date and time in the Browser</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
function begin(form)
                {
                form_name = form;
                time_out=window.setTimeout("display_date()",500)
                }

function display_date()
                {
                form_name.date.value=new Date();
                time_out=window.setTimeout("display_date()",1000)
                }
function display_clock()
{
document.write('<FONT COLOR=red SIZE=+1><FORM
NAME=time_form><CENTER><BR><BR>Current Date & Time :')
document.write('  <INPUT NAME=date size=19
value=""</FORM></FONT></CENTER>')
begin(document.time_form)
}
display_clock()
</SCRIPT>
</BODY>
</HTML>
```

Check Your Progress 2

1.

```
<HTML>
  <HEAD>
    <TITLE>Creating and Using User Defined Functions</TITLE>
      <SCRIPT LANGUAGE="JavaScript">
          var name = "";
          function hello( )
          {
              name = prompt('Enter Your Name:','Name');
              alert('Greetings ' + name + ', Welcome to my page!');
          }

          function goodbye( )
          {
              alert('Goodbye ' + name + ', Sorry to see you go!');
          }
       </SCRIPT>
  </HEAD>
  <BODY onLoad="hello( );" onUnload="goodbye( );">
       <IMG SRC="Images\Pinkwhit.gif ">
  </BODY>
</HTML>
```

Check Your Progress 3

1.

```
<HTML>
 <HEAD>
        <TITLE>FORMS</TITLE>
                <!-- This code allows to access the Form objects Elements Array //-->
   <SCRIPT Language="JavaScript">
function Ver(form1)
{
        v=form1.elements.length;
if (form1.elements[3].name=="Button1")
{
                alert('First form name : '+document.forms[0].name);
                alert('No. of Form elements of  ' +document.forms[0].name + ' = '+v);

}
        else if (form1.elements[4].name=="Button2")
        {
        alert('Second form name : '+document.forms[1].name);
        alert('No. of Form elements of  ' +document.forms[1].name + ' = '+v);
        }
        for(i=0;i<v;i++)
        alert(form1.elements[i].name+ ' is at position '+i);
}
  </SCRIPT>
</HEAD>
```

```
<BODY>
<FORM Name="Survey Form 1">
        FIRST FORM: <i><b>Survey Form 1 </b></i><BR><BR>
        First Name : <Input Type=Text Name="Text1" Value=""><BR><BR>
                <Input Type="radio" Name="Radio1" Value=""> Fresher
                <Input Type="radio" Name="Radio1" Value="">
Experienced<BR><BR>
                <Input Type="Button" Name="Button1" Value="Click1"
onClick="Ver(form)">
</FORM>

<FORM Name="Survey Form 2">
                SECOND FORM: <i><b> Survey Form 2 </b></i><BR><BR>
        Name : <Input Type="Text" Name="Text2" Value=""> <BR><BR>
                Password : <Input Type="Password" Name="Pass2" Value="">
<BR><BR>
                <Input Type="CheckBox" Name="Check1" Value="" > Employed
                <Input Type="CheckBox" Name="Check2" Value="" > Studying
<BR><BR>
                <Input Type="Button" Name="Button2" Value="Click2"
onClick="Ver(form)">
</FORM>
</BODY>
</HTML>
```

2.

```
<HMTL>
 <HEAD>
        <TITLE>FORMS</TITLE>

        <!-- This code checks the Checkbox when the button is clicked //-->

   <SCRIPT Language='JavaScript'>
  function Chk(f1)
{
 f1.Check.checked=true;
 alert(" The Checkbox just got checked ");
 f1.Check.checked=false;
 f1.Radio[0].checked=true;
 f1.Radio[1].checked=false;
 alert(" The Radio button just got checked ");
}
  </SCRIPT>
 </HEAD>
```

Comment: The desired functionality is not apparent in this question. It should be stated explicitly.

```
<BODY>
<FORM>
        Client Name : <Input Type=Text Name="Text" Value=""><BR><BR>
```

```
        Client Address: <Input Type=Text Name="Text1" Value=""> <BR><BR>
        Client E-mail Address :<Input Type=Text Name="Text2"
Value=""><BR><BR>
        <Input Type="radio" Name="Radio" Value=""> Male
        <Input Type="radio" Name="Radio" Value=""> Female<BR><BR>
        <Input Type="CheckBox" Name="Check" Value=""> Employed <BR><BR>
<Input Type="Button" Name="Bt" Value="Set Element Array Value"
onClick="Chk(this.form)">
</FORM>
</BODY>
</HTML>
```

3.

```
<HTML>
<HEAD>
<TITLE> Working with Radio Buttons </TITLE>
<SCRIPT LANGUAGE="JavaScript">
function calculate(form)
        {
      if(form.elements[2].checked)
        {
                form.result.value = form.entry.value * form.entry.value;
          }
    else
        {
                        form.result.value = form.entry.value * 2;
                }
      }
</SCRIPT>
</HEAD>
<BODY>
<FORM >
<CENTER><BR>
<B>Value:</B>
<INPUT  TYPE="text"  NAME="entry"  VALUE=0>
<BR><BR>
<SPACER Size= 190>
<B>Action:<B><BR>

<SPACER Size = 225>
<INPUT  TYPE="radio"  NAME="action1"  VALUE="twice"
  onClick="calculate(this.form);">Double<BR>
<SPACER Size = 225>
<INPUT  TYPE="radio"  NAME="action1"  VALUE="square"
          onClick="calculate(this.form);">Square <BR><BR>
<B>Result:</B>
<INPUT  TYPE=text  NAME="result" onFocus = "this.blur();">
</CENTER>
</FORM>
```

</BODY>
</HTML>

4. (Case Study)

Comment: This code keeps saying "Invalid Phone" or "Invalid Name". This needs to be corrected.

```
<HTML>
<HEAD><TITLE>Order Form</TITLE>
<SCRIPT>
// function to calculate the total cost field
function Total()
{ var tot = 0;
tot += (240 * document.order.qty1.value);
tot += (270 * document.order.qty2.value);
tot += (300 * document.order.qty3.value);
tot += (600 * document.order.qty4.value);
document.order.totalcost.value = tot; }
// function to update cost when quantity is changed
function UpdateCost(number, unitcost)
{ costname = "cost" + number; qtyname = "qty" + number;
var q = document.order[qtyname].value;
document.order[costname].value = q * unitcost;
Total();
}

// function to copy billing address to shipping address
function CopyAddress()
{ if (document.order.same.checked)
{ document.order.shipto.value = document.order.billto.value;
}
}
//global variable for error flag
var errfound = false;

//function to validate by length
function ValidLength(item, len)
{ return (item.length >= len); }

//function to validate an email address
function ValidEmail(item)
{ if (!ValidLength(item, 5)) return false;
if (item.indexOf ('@', 0) == -1)
return false;
return true;
}

// display an error alert
```

```
function error(elem, text)
{
// abort if we already found an error
 if (errfound) return;
window.alert(text);
elem.select();
elem.focus();
errfound = true;
}
// main validation function
function Validate()
{
errfound = false;
if (!ValidLength(document.order.name1.value,6))
error(document.order.name1,"Invalid Name");
if (!ValidLength(document.order.phone.value,10))
error(document.order.phone,"Invalid Phone");
if (!ValidLength(document.order.billto.value,30))
error(document.order.billto,"Invalid Billing Address");
if (!ValidLength(document.order.shipto.value,30))
error(document.order.shipto,"Invalid Shipping Address");
if (!ValidEmail(document.order.email.value))
error(document.order.email, "Invalid Email Address");
if (document.order.totalcost.value == "")
error(document.order.qty1, "Please Order at least one item.");
if (document.order.payby.selectedIndex != 1)
{
if (!ValidLength(document.order.creditno.value,2))
error(document.order.creditno,"Invalid Credit/Check number");
}
return !errfound; /* true if there are no errors */ }
</SCRIPT> </HEAD>
<BODY>
<H1>Online Cake Order Form</H1>
<FORM NAME="order" onSubmit="return Validate();">
<B>Name:</B>
<INPUT TYPE="text" NAME="name1" SIZE=20>
 <B>Phone: </B>
<INPUT TYPE="text" NAME="phone" SIZE=15>
<B>E-mail address:</B>
<INPUT TYPE="text" NAME="email" SIZE=20><BR><BR>
<B>Shipping Address:</B>
<BR>
```

```
<TEXTAREA NAME="shipto" COLS=40 ROWS=4 onChange="CopyAddress();">
Enter your shipping address here. </TEXTAREA>
<B>Products to Order:</B><BR>
Qty: <INPUT TYPE="TEXT" NAME="qty1" VALUE="0" SIZE=4 onChange =
"UpdateCost(1, 240);">
Cost: <INPUT TYPE="TEXT" NAME="cost1" SIZE=6> (Rs.240 ) Vanilla cake
<BR>
Qty: <INPUT TYPE="TEXT" NAME="qty2" VALUE="0" SIZE=4 onChange =
"UpdateCost(2, 270);">
Cost: <INPUT TYPE="TEXT" NAME="cost2" SIZE=6> (Rs.270 ) Strawberry cake
<BR>
Qty: <INPUT TYPE="TEXT" NAME="qty3" VALUE="0" SIZE=4 onChange =
"UpdateCost(3, 300);">
 Cost: <INPUT TYPE="TEXT" NAME="cost3" SIZE=6> (Rs.300 ) Black Forest
cake <BR>
Qty: <INPUT TYPE="TEXT" NAME="qty4" VALUE="0" SIZE=4 onChange =
"UpdateCost(4, 600);">
Cost: <INPUT TYPE="TEXT" NAME="cost4" SIZE=6> (Rs.600 ) Chocolate cake
<HR> <B>Total Cost:</B> <INPUT TYPE="TEXT" NAME="totalcost"
SIZE=8><HR> <B>
Method of Payment</B>:
<SELECT NAME="payby"> <OPTION VALUE="check" SELECTED>
Check or Money Order
<OPTION VALUE="cash">Cash or Cashier's Check
<OPTION VALUE="credit">Credit Card (specify number)
</SELECT><BR> <B>Credit Card or Check Number:</B>:
<INPUT TYPE="TEXT" NAME="creditno" SIZE="20"><BR>
<INPUT TYPE="SUBMIT" NAME="submit" VALUE="Send Your Order">
<INPUT TYPE="RESET" VALUE="Clear Form">
</FORM>
</BODY>
</HTML>
```

4.11 FURTHER READINGS

1. JavaScript Programmers Reference, Cliff Wootton. Publisher: Wrox Press Inc. Year 1999.

2. Beginning JavaScript, Paul Wilton. Publisher: Wrox Press Inc. 1st Edition

3. JavaScript: The Definitive Guide, David Flanagan. Publisher: O'Reilly. 4th Edition 2001.

4. The JavaScript Bible, Danny Goodman. Publisher: John Wiley & Sons Inc. Edition 2001.

Page 98: [1] Comment **milind**

What is number here? Will this code work? In any case the function does not do anything meaningful. What is the purpose of the argument ‘a’ here?

Page 106: [2] Comment **milind**

Please be accurate. If you do use the replace method but do not change the history entry, does the statement still hold good?

Page 114: [3] Comment **milind**

The solution to the case study does not work correctly. This needs to be taken care of.

UNIT 5 VB SCRIPT

Structure **Page No.**

5.0 INTRODUCTION

After learning JavaScript to make WebPages that require some logical processing, we will discuss another very common scripting language, called VB Script. VBScript is Microsoft's scripting language. It enables us to write programs that enhance the power of Web pages by allowing us to control their behaviour. Although HTML enables us to develop Web pages, we cannot incorporate into them conditions or business rules without using another tool like a scripting language.

In this unit we will discuss the programming constructs that are supported by VBScript. The constructs will enable you to implement all kinds of logic for your Web pages.

We will also discuss about Objects in VBScript and talk about the Dictionary object in detail as an example. Error handling is an important part of any kind of programming. In this unit we discuss error handling in VBScript. The Err object is used for the purpose by VBScript

5.1 OBJECTIVES

The objective of this unit is to explain the basics of VBScript. After completing this unit, you will be able to write code for Web pages using VBScript. You will be able to understand the following components of VBScript:

- Variables
- Loops
- Conditional statements
- Procedures
- Properties and Methods of Dictionary Object
- Properties and Methods of the ERR Object.

5.2 WHAT IS VBSCRIPT?

VBScript is a member of Microsoft's Visual Basic family of development products. Other members include Visual Basic (Professional and Standard Editions) and Visual Basic for Applications, which is the scripting language for Microsoft Excel. VBScript is a scripting language for HTML pages on the World Wide Web and corporate intranets. VBScript is powerful and has almost all the features of Visual Basic. One of the things you should be concerned about is the safety and security of client machines that access your Web site. Microsoft took this consideration into account when creating VBScript. Potentially dangerous operations that can be done in Visual Basic have been removed from VBScript, including the capability to access dynamic link libraries directly and to access the file system on the client machine.

5.3 ADDING VBSCRIPT CODE TO AN HTML PAGE

You can use the SCRIPT element to add VBScript code to an HTML page.

The <SCRIPT> Tag
VBScript code is written within paired <SCRIPT> tags. For example, a procedure to test a delivery date might appear as follows:

```
<SCRIPT LANGUAGE="VBScript">
  <!--
   Function CanDeliver(Dt)
      CanDeliver = (CDate(Dt) - Now()) > 2
   End Function
  -->
</SCRIPT>
```

Beginning and ending <SCRIPT> tags surround the code. The LANGUAGE attribute indicates the scripting language. You must specify the language because browsers can use other scripting languages, such as JavaScript. Notice that the CanDeliver function is embedded in comment tags (<!-- and -->). This prevents browsers that do not support the <SCRIPT> tag from displaying the code on the page.

Since the example is a general function - it is not tied to any particular form control - you can include it in the HEAD section of the page as shown in Figure 5.1 below:

```
<HTML>
<HEAD>
<TITLE>Place Your Order</TITLE>
<SCRIPT LANGUAGE="VBScript">
  <!--
   Function CanDeliver(Dt)
      CanDeliver = (CDate(Dt) - Now()) > 2
   End Function
  -->
</SCRIPT>
</HEAD>

<BODY>
...
```

You can use SCRIPT blocks anywhere in an HTML page. You can put them in both the BODY and HEAD sections. However, you will probably want to put all general-purpose scripting code in the HEAD section in order to keep all the code together. Keeping your code in the HEAD section ensures that all the code is read and decoded before it is needed by any calls from within the BODY section.

One notable exception to this rule is that you may want to provide inline scripting code within forms to respond to the events of objects in your form. For example, you can embed scripting code to respond to a button click in a form as shown in Figure 5.2:

```
<HTML>
<HEAD>
<TITLE>Test Button Events</TITLE>
</HEAD>
  <BODY>
     <FORM NAME="Form1">
       <INPUT TYPE="Button" NAME="Button1"
     VALUE="Click">
       <SCRIPT FOR="Button1" EVENT="onClick"
     LANGUAGE="VBScript">
          MsgBox "Button Pressed!"
       </SCRIPT>
     </FORM>
  </BODY>
</HTML>
```

This example will display a button on the Web Page. When you click on the button it would display a message box stating “Button Pressed”.

Most of your code will appear in either **Sub** or **Function** procedures and will be called only when the code you have written causes that function to execute. However, you can write VBScript code outside procedures, but still within a SCRIPT block. This code is executed only once, when the HTML page loads. This allows you to initialize data or dynamically change the look of your Web page when it loads.

5.4 VB SCRIPT BASICS

This section will discuss the basics of VBScript and describe concepts like datatypes, loops and others.

5.4.1 VBScript Data Types

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 is used. Because **Variant** is the only data type in VBScript, it is 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 are working with data that looks like numbers, VBScript assumes that it is a number and does whatever is most appropriate for numbers. Similarly, if you are working with data that can only be string data, VBScript treats it as string data. You can always make numbers behave as strings by enclosing them in quotation marks (" ").

Variant Subtypes

Beyond the simple numeric or string classifications, a **Variant** can make further distinctions about the specific nature of numeric information. For example, you can have numeric information that represents a date or a time. When used with other date or time data, the result is expressed as a date or a time. You can also have a rich variety of numeric information ranging from Boolean values to huge floating-point numbers. These different categories of information that can be contained in a **Variant** are called subtypes. Most of the time, you can just put the kind of data you want in a **Variant**, and the **Variant** behaves in a way that is most appropriate for the data it contains.

The following table shows the subtypes of data that a **Variant** can contain.

Subtype	Description
EmptyVariant	Uninitialized. Value is 0 for numeric variables or a zero-length string ("") for string variables.
NullVariant	Iintentionally 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	Contains a number that represents a date between January 1, 100 to December 31, 9999.
Time	Represents a time between 0:00:00 and 23:59:59
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.

You can use conversion functions to convert data from one subtype to another. In addition, the VarType function returns information about how your data is stored within a **Variant**.

5.4.2 VBScript Variables

A variable is a convenient placeholder that refers to a computer memory location where you can store program information that may change while your script is running. For example, you might create a variable called ClickCount to store the number of times a user clicks an object on a particular Web page. Where the variable is stored in computer memory is unimportant. What is important is that you only have to refer to a variable by name to see its value or to change its value. In VBScript, variables are always of one fundamental data type, Variant.

Declaring Variables

You declare variables explicitly in your script using the Dim statement, the Public statement, and the Private statement. For example:

Dim DegreesFahrenheit

You declare multiple variables by separating each variable name with a comma. For example:

```
Dim Top, Bottom, Left, Right
```

You can also declare a variable implicitly by simply using its name in your script. That is not generally a good practice because you could misspell the variable name in one or more places, causing unexpected results when your script is running. For that reason, the Option Explicit statement is available to require explicit declaration of all variables.

Naming Restrictions

Variable names follow the standard rules for naming anything in VBScript. A variable name:

- Must begin with an alphabetic character.
- Cannot contain an embedded period.
- Must not exceed 255 characters.
- Must be unique in the scope in which it is declared.

Scope and Lifetime of Variables

The scope of a variable is determined by where you declare it. When you declare a variable 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 you declare a variable outside a procedure, you make it visible to all the procedures in your script. This is a script-level variable, and it has script-level scope.

How long a variable exists defines its lifetime. The lifetime of a script-level variable extends from the time it is declared until the time the script is 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. Local variables are ideal as temporary storage space when a procedure is executing. You can have local variables of the same name in several different procedures because each is recognized only by the procedure in which it is declared.

Assigning Values to Variables

Values are assigned to variables creating an expression as follows: the variable is on the left side of the expression and the value you want to assign to the variable is on the right, with the '=' sign being the assignment operator. For example:

```
B = 200
```

Scalar Variables and Array Variables

Most of the time, you just want to assign a single value to a variable you have declared. A variable containing a single value is a scalar variable. At other times, it is convenient to assign more than one related value to a single variable. Then you can create a variable that can contain a series of values. This is called an array variable. Array variables and scalar variables are declared in the same way, except that the declaration of an array variable uses parentheses ( ) following the variable name. In the following example, a single-dimension array containing 11 elements is declared:

```
Dim A(10)
```

Although the number shown in the parentheses is 10, all arrays in VBScript are counted from base 0, so that this array actually contains 11 elements. In such an array, the number of array elements is always the number shown in parentheses plus one. This kind of array is called a fixed-size array.

You assign data to each of the elements of the array using an index into the array. Beginning at zero and ending at 10, data can be assigned to the elements of an array as follows:

```
A(0) = 256
A(1) = 324
A(2) = 100
. . .
A(10) = 55
```

Similarly, the data can be retrieved from any element using an index into the particular array element you want. For example:

```
. . .
SomeVariable = A(8)
. . .
```

Arrays are not limited to a single dimension. You can have as many as 60 dimensions, although most people cannot comprehend more than three or four dimensions. Multiple dimensions are declared by separating an array's size numbers in the parentheses with commas. In the following example, the MyTable variable is a two-dimensional array consisting of 6 rows and 11 columns:

```
Dim MyTable(5, 10)
```

In a two-dimensional array, the first number is always the number of rows; the second number is the number of columns.

You can also declare an array whose size changes while your script is running. This is called a dynamic array. The array is initially declared within a procedure using either the **Dim** statement or using the ReDim statement. However, for a dynamic array, no size or number of dimensions is placed inside the parentheses.

For example:

```
Dim MyArray()
ReDim AnotherArray()
```

To use a dynamic array, you must subsequently use **ReDim** to determine the number of dimensions and the size of each dimension. In the following example, **ReDim** sets the initial size of the dynamic array to 25. A subsequent **ReDim** statement resizes the array to 30, but uses the **Preserve** keyword to preserve the contents of the array as the resizing takes place.

```
ReDim MyArray(25)
. . .
ReDim Preserve MyArray(30)
```

There is no limit to the number of times you can resize a dynamic array, but you should know that if you make an array smaller than it was, you lose the data in the eliminated elements.

5.4.3 VBScript Constants

A constant is a meaningful name that takes the place of a number or string and never changes. VBScript defines a number of intrinsic constants. You can get detailed information about these intrinsic constants from the VBScript Language Reference.

Creating Constants

You create user-defined constants in VBScript using the Const statement. This lets you create string or numeric constants with meaningful names and allows you to assign them literal values. For example:

```
Const MyString = "This is my string."
Const MyAge = 49
```

Note that the string literal is enclosed in quotation marks (" "). Quotation marks are the most obvious way to differentiate string values from numeric values. Date literals and time literals are represented by enclosing them in number signs (#). For example:

```
Const CutoffDate = #6-1-97#
```

You may want to adopt a naming scheme to differentiate constants from variables. This will save you from trying to reassign constant values while your script is running. For example, you might want to use a "vb" or "con" prefix on your constant names, or you might name your constants in all capital letters. Differentiating constants from variables eliminates confusion as you develop more complex scripts.

5.4.4 VBScript Operators

VBScript has a full range of operators, including arithmetic operators, comparison operators, concatenation operators, and logical operators.

Operator Precedence

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called operator precedence. You can use parentheses to override the order of precedence and force some parts of an expression to be evaluated before others. Operations within parentheses are always performed before those outside. Within parentheses, however, standard operator precedence is maintained.

When expressions contain operators from more than one category, arithmetic operators are evaluated first, comparison operators are evaluated next, and logical operators are evaluated last. Comparison operators all have equal precedence; that is, they are evaluated in the left-to-right order in which they appear. Arithmetic and logical operators are evaluated in the following order of precedence.

1. Arithmetic operators		**2. Comparison operators**		**3. Logical operators**	
Description	**Symbol**	**Description**	**Symbol**	**Description**	**Symbol**
Exponentiation	^	Equality	=	Logical negation	Not
Unary negation	-	Inequality	<>	Logical conjunction	And
Multiplication	*	Less than	<	Logical disjunction	Or
Division	/	Greater than	>	Logical exclusion	Xor
Integer division	\	Less than or equal to	<=	Logical equivalence	Eqv
Modulus arithmetic	Mod	Greater than or equal to	>=	Logical implication	Imp
Addition	+	Object equivalence	Is		
Subtraction	-				
String concatenation	&				

The associativity of the operators is left to right. When multiplication and division occur together in an expression, each operation is evaluated as it occurs from left to right. Likewise, when addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

The string concatenation (&) operator is not an arithmetic operator, but in precedence it falls after all arithmetic operators and before all comparison operators. The **Is** operator is an object reference comparison operator. It does not compare objects or their values; it checks only to determine if two object references refer to the same object.

5.5 USING CONDITIONAL STATEMENTS

You can control the flow of your script with conditional statements and looping statements. Using conditional statements, you can write VBScript code that makes decisions and repeats actions. The following conditional statements are available in VBScript:

If...Then...Else statement
Select Case statement

Making Decisions Using If...Then...Else

The **If...Then...Else** statement is used to evaluate whether a condition is **True** or **False** and, depending on the result, to specify one or more statements to execute. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. For information about comparison operators, see Comparison Operators. **If...Then...Else** statements can be nested to as many levels as you need.

Running Statements if a Condition is True

To run only one statement when a condition is **True**, use the single-line syntax for the **If...Then...Else** statement. The following example shows the single-line syntax. Notice that this example omits the **Else** keyword.

```
Sub FixDate()
   Dim myDate
   myDate = #2/13/95#
   If myDate < Now Then myDate = Now
End Sub
```

To run more than one line of code, you must use the multiple-line (or block) syntax. This syntax includes the **End If** statement, as shown in the following example:

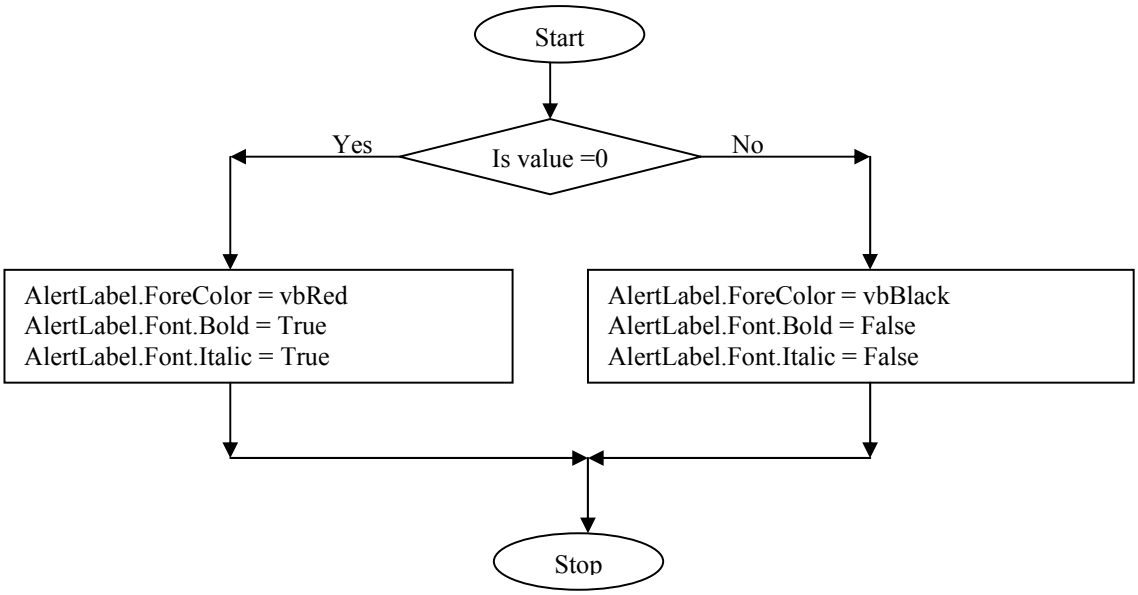
```
Sub AlertUser(value)
      If value = 0 Then
         AlertLabel.ForeColor = vbRed
         AlertLabel.Font.Bold = True
         AlertLabel.Font.Italic = True
      End If
   End Sub
```

Running Certain Statements if a Condition is True and Running Others if a Condition is False

You can use an **If...Then...Else** statement to define two blocks of executable statements: one block to run if the condition is **True**, the other block to run if the condition is **False**.

```
Sub AlertUser(value)
   If value = 0 Then
      AlertLabel.ForeColor = vbRed
      AlertLabel.Font.Bold = True
      AlertLabel.Font.Italic = True
   Else
      AlertLabel.Forecolor = vbBlack
      AlertLabel.Font.Bold = False
      AlertLabel.Font.Italic = False
   End If
End Sub
```

The following flow chart explains the flow of the above example.



Deciding Between Several Alternatives

A variation on the **If...Then...Else** statement allows you to choose from several alternatives. Adding **ElseIf** clauses expands the functionality of the **If...Then...Else** statement so that you can control program flow based on different possibilities. For example:

```
Sub ReportValue(value)
   If value = 0 Then
      MsgBox value
   ElseIf value = 1 Then
      MsgBox value
   ElseIf value = 2 then
      Msgbox value
```

```
Else
   Msgbox "Value out of range!"
End If
```

You can add as many **ElseIf** clauses as you need to provide alternative choices. Extensive use of the **ElseIf** clauses often becomes cumbersome. A better way to choose between several alternatives is the **Select Case** statement.

Making Decisions with Select Case

The **Select Case** structure provides an alternative to **If...Then...ElseIf** for selectively executing one block of statements from among multiple blocks of statements. A **Select Case** statement provides capability similar to the **If...Then...Else statement**, but it makes code more efficient and readable.

A **Select Case** structure works with a single test expression that is evaluated once, at the top of the structure. The result of the expression is then compared with the values for each **Case** in the structure. If there is a match, the block of statements associated with that **Case** is executed:

```
Select Case Document.Form1.CardType.Options(SelectedIndex).Text
   Case "MasterCard"
      DisplayMCLogo
      ValidateMCAccount
   Case "Visa"
      DisplayVisaLogo
      ValidateVisaAccount
   Case "American Express"
      DisplayAMEXCOLogo
      ValidateAMEXCOAccount
   Case Else
      DisplayUnknownImage
      PromptAgain
End Select
```

Notice that the **Select Case** structure evaluates an expression once at the top of the structure. In contrast, the **If...Then...ElseIf** structure can evaluate a different expression for each **ElseIf** statement. You can replace an **If...Then...ElseIf** structure with a **Select Case** structure only if each **ElseIf** statement evaluates the same expression.

5.6 LOOPING THROUGH CODE

Using Loops to Repeat Code

Looping allows you to run a group of statements repeatedly. Some loops repeat statements until a condition is **False**; others repeat statements until a condition is **True**. There are also loops that repeat statements a specific number of times. The following looping statements are available in VBScript:

Do...Loop: Loops while or until a condition is **True**.
While...Wend: Loops while a condition is **True**.
For...Next: Uses a counter to run statements a specified number of times.
For Each...Next: Repeats a group of statements for each item in a collection or

each element of an array.

Using Do Loops

You can use **Do...Loop** statements to run a block of statements an indefinite number of times. The statements are repeated either while a condition is **True** or until a condition becomes **True**.

Repeating Statements While a Condition is True

Use the **While** keyword to check a condition in a **Do...Loop** statement. You can check the condition before you enter the loop (as shown in the following ChkFirstWhile example), or you can check it after the loop has run at least once (as shown in the ChkLastWhile example). In the ChkFirstWhile procedure, if myNum is set to 9 instead of 20, the statements inside the loop will never run. In the ChkLastWhile procedure, the statements inside the loop run only once because the condition is already **False**.

```
Sub ChkFirstWhile()
   Dim counter, myNum
   counter = 0
   myNum = 20
   Do While myNum > 10
      myNum = myNum - 1
      counter = counter + 1
   Loop
   MsgBox "The loop made " & counter & " repetitions."
End Sub

Sub ChkLastWhile()
   Dim counter, myNum
   counter = 0
   myNum = 9
   Do
      myNum = myNum - 1
      counter = counter + 1
   Loop While myNum > 10
   MsgBox "The loop made " & counter & " repetitions."
End Sub
```

The above figure shows two example code fragments illustrating the do… loop and do while … loop statements. In the first loop the condition is checked before executing the code but in the second loop the statements are first executed once and then the condition is checked. In the first loop the instructions given in the loop's body will never be executed if the condition fails initially but in the second loop these instructions would be executed at least once. The following flow chart shows the execution plan of the second procedure i.e. CheckLastWhile:

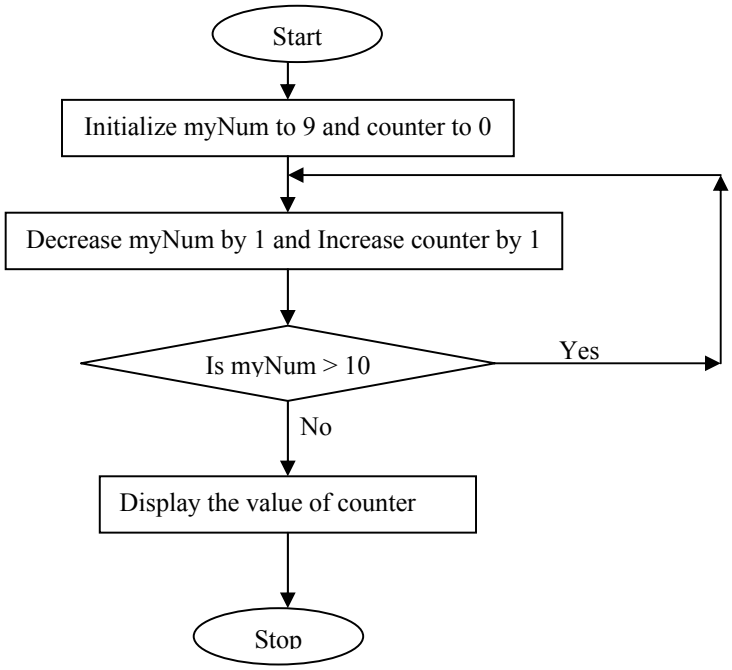


Figure 5.1: Execution Plan of the Second Loop in Figure 5.9

Repeating a Statement Until a Condition Becomes True

You can use the **Until** keyword in two ways to check a condition in a **Do...Loop** statement. You can check the condition before you enter the loop (as shown in the following ChkFirstUntil example), or you can check it after the loop has run at least once (as shown in the ChkLastUntil example). As long as the condition is **False**, the loop continues.

```
Sub ChkFirstUntil()
   Dim counter, myNum
   counter = 0
   myNum = 20
   Do Until myNum = 10
      myNum = myNum - 1
      counter = counter + 1
   Loop
   MsgBox "The loop made " & counter & " repetitions."
End Sub

Sub ChkLastUntil()
   Dim counter, myNum
   counter = 0
   myNum = 1
   Do
      myNum = myNum + 1
      counter = counter + 1
   Loop Until myNum = 10
   MsgBox "The loop made " & counter & " repetitions."
End Sub
```

Exiting a Do...Loop Statement from Inside the Loop

You can exit a **Do...Loop** by using the **Exit Do** statement. Because you usually want to exit only in certain situations, such as to avoid an endless loop, you

should use the **Exit Do** statement in the **True** statement block of an **If...Then...Else** statement. If the condition is **False**, the loop runs as usual. Otherwise, you probably did not need a loop in the first place.

In the following example, myNum is assigned a value that creates an endless loop. The **If...Then...Else** statement checks for this condition, preventing the endless repetition.

```
Sub ExitExample()
   Dim counter, myNum
   counter = 0
   myNum = 9
   Do Until myNum = 10
      myNum = myNum - 1
      counter = counter + 1
      If myNum < 10 Then Exit Do
   Loop
   MsgBox "The loop made " & counter & " repetitions."
End Sub
```

Using While...Wend

The **While...Wend** statement is provided in VBScript for those who are familiar with its usage. However, because of the lack of flexibility in **While...Wend**, it is recommended that you use **Do...Loop** instead.

Using For...Next

You can use **For...Next** statements to run a block of statements a specific number of times. For loops, use a counter variable whose value is increased or decreased with each repetition of the loop.

For example, the following procedure causes a procedure called MyProc to execute 50 times. The **For** statement specifies the counter variable x and its start and end values. The **Next** statement increments the counter variable by 1.

```
Sub DoMyProc50Times()
   Dim x
   For x = 1 To 50
      MyProc
   Next
End Sub
```

Using the **Step** keyword, you can increase or decrease the counter variable by the value you specify. In the following example, the counter variable j is incremented by 2 each time the loop repeats. When the loop is finished, total is the sum of 2, 4, 6, 8, and 10.

```
Sub TwosTotal()
   Dim j, total
   For j = 2 To 10 Step 2
      total = total + j
   Next
   MsgBox "The total is " & total
```

```
End Sub
```

To decrease the counter variable, you use a negative **Step** value. You must specify an end value that is less than the start value. In the following example, the counter variable myNum is decreased by 2 each time the loop repeats. When the loop is finished, total is the sum of 16, 14, 12, 10, 8, 6, 4, and 2.

```
Sub NewTotal()
   Dim myNum, total
   For myNum = 16 To 2 Step -2
      total = total + myNum
   Next
   MsgBox "The total is " & total
End Sub
```

You can exit any **For...Next** statement before the counter reaches its end value by using the **Exit For** statement. Because you usually want to exit only in certain situations, such as when an error occurs, you should use the **Exit For** statement in the **True** statement block of an **If...Then...Else** statement. If the condition is **False**, the loop runs as usual. Otherwise you probably did not need to use the loop in the first place.

Using For Each...Next

A **For Each...Next** loop is similar to a **For...Next** loop. Instead of repeating the statements a specified number of times, a **For Each...Next** loop repeats a group of statements for each item in a collection of objects or for each element of an array. This is especially helpful if you do not know how many elements are present in a collection.

In the following HTML code example, the contents of a **Dictionary** object are used to place text in several text boxes:

Comment: This code example does NOT illustrate the For Each…Next loop. If the count of d can be determined, then one need not be using this loop and the For…Next loop will do. Please give a better example. At a minimum, write this example to make use of the For Each…Next loop.

```
<HTML>
<HEAD><TITLE>Forms and Elements</TITLE></HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChange_OnClick
  Dim d                'Create a variable
  Set d = CreateObject("Scripting.Dictionary")
  d.Add "0", "Athens"     'Add some keys and items
  d.Add "1", "Belgrade"
  d.Add "2", "Cairo"

 a = d.items
  For i = 0 To d.Count
     document.frmForm.Elements(i).Value = a(i)
  Next
End Sub
-->
</SCRIPT>
<BODY>
```

```
<CENTER>
<FORM NAME="frmForm"> <p>
   <Input Type = "Text"><p>
   <Input Type = "Text"><p>
   <Input Type = "Text"><p>
   <Input Type = "Button" NAME="cmdChange" VALUE="Click
Here"><p>
</FORM>
</CENTER>
</BODY>
</HTML>
```

Check Your Progress 1

1. Write code to check whether a number is even and prime or odd and prime.

2. Write code in VBScript to generate the Fibonnacci series. This goes 1, 1, 2, 3, 5, 8,… and so on. From the third number onwards, the nth number is the sum of the (n – 1)th and the (n – 2) th numbers.

3. Write code to find the factorial of any given number. This is the product of all numbers from 2 to that number.

5.7 VBSCRIPT PROCEDURES

In VBScript there are two kinds of procedures; the Sub procedure and the Function procedure.

Sub Procedures

A **Sub** procedure is a series of VBScript statements, enclosed by **Sub** and **End Sub** statements, that perform actions but don't return a value. A **Sub** procedure can take arguments (constants, variables, or expressions that are passed by a calling procedure). If a **Sub** procedure has no arguments, its **Sub** statement must include an empty set of parentheses ().

Figure 5.17 shows a **Sub** procedure uses two intrinsic, or built-in, VBScript functions, MsgBox and InputBox, to prompt a user for some information. It then displays the results of a calculation based on that information. The calculation is performed in a **Function** procedure created using VBScript. The **Function** procedure is shown after the following discussion.

```
Sub ConvertTemp()
   temp = InputBox("Please enter the temperature in degrees F.", 1)
   MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub
```

Function Procedures

A **Function** procedure is a series of VBScript statements enclosed by the **Function** and **End Function** statements. A **Function** procedure is similar to a **Sub** procedure, but can also return a value. A **Function** procedure can take arguments (constants, variables, or expressions that are passed to it by a calling procedure). If a **Function** procedure has no arguments, its **Function** statement must include an empty set of parentheses (). A **Function** returns a value by

assigning a value to its name in one or more statements of the procedure. The return type of a **Function** is always a **Variant**.

In the following example, the Celsius function calculates degrees Celsius from degrees Fahrenheit. When the function is called from the ConvertTemp **Sub** procedure, a variable containing the argument value is passed to the function. The result of the calculation is returned to the calling procedure and displayed in a message box.

```
Sub ConvertTemp()
   temp = InputBox("Please enter the temperature in degrees F.", 1)
   MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub

Function Celsius(fDegrees)
   Celsius = (fDegrees - 32) * 5 / 9
End Function
```

Getting Data into and out of Procedures

Each piece of data is passed into your procedures using an argument. Arguments serve as placeholders for the data you want to pass into your procedure. You can name your arguments with any valid variable name. When you create a procedure using either the **Sub** statement or the **Function** statement, parentheses must be included after the name of the procedure. Any arguments are placed inside these parentheses, separated by commas. For example, in the following example, fDegrees is a placeholder for the value being passed into the Celsius function for conversion:

```
Function Celsius(fDegrees)
   Celsius = (fDegrees - 32) * 5 / 9
End Function
```

To get data out of a procedure, you must use a **Function**. Remember, a **Function** procedure can return a value; a **Sub** procedure cannot.

Using Sub and Function Procedures in Code

A **Function** in your code must always be used on the right side of a variable assignment or in an expression. For example:

```
Temp = Celsius(fDegrees)
```
or
```
MsgBox "The Celsius temperature is " & Celsius(fDegrees) & " degrees."
```

To call a **Sub** procedure from another procedure, you can just type the name of the procedure along with values for any required arguments, each separated by a comma. The Call statement is not required, but if you do use it, you must enclose any arguments in parentheses.

The following example shows two calls to the MyProc procedure. One uses the **Call** statement in the code; the other does not. Both do exactly the same thing.

```
Call MyProc(firstarg, secondarg)
MyProc firstarg, secondarg
```

Notice that the parentheses are omitted in the call when the **Call** statement isn't used.

5.8 VB SCRIPT CODING CONVENTIONS

Coding conventions are suggestions that may help you write code using Microsoft Visual Basic Scripting Edition. Coding conventions can include the following:

Naming conventions for objects, variables, and procedures
Commenting conventions
Text formatting and indenting guidelines

The main reason for using a consistent set of coding conventions is to standardize the structure and coding style of a script or set of scripts so that you and others can easily read and understand the code. Using good coding conventions results in precise, readable, and unambiguous source code that is consistent with other language conventions and is as intuitive as possible.

Constant Naming Conventions

Earlier versions of VBScript had no mechanism for creating user-defined constants. Constants, if used, were implemented as variables and distinguished from other variables using all uppercase characters. Multiple words were separated using the underscore (_) character. For example:

USER_LIST_MAX
NEW_LINE

While this is still an acceptable way to identify your constants, you may want to use an alternative naming scheme, now that you can create true constants using the Const statement. This convention uses a mixed-case format in which constant names have a "con" prefix. For example:

ConYourOwnConstant

Variable Naming Conventions

For purposes of readability and consistency, use the following prefixes with descriptive names for variables in your VBScript code.

Sub type	Prefix	Example
Boolean	Bln	BlnFound
Byte	Byt	BytQuantity
Date	Dtm	DtmStart
Double	Dbl	DblPrice
Error	Err	ErrOrderNum
Integer	Int	IntPrice
Long	Lng	LngDistance
Object	Obj	ObjCurrent
Single	Sng	SngAverage
String	Str	StrName

Variable Scope

Variables should always be defined with the smallest scope possible. VBScript variables can have the following scope.

Scope	Where Variable is Declared	Visibility
Procedure-level	Event, Function, or Sub procedure	Visible in the procedure in which it is declared
Script-level	HEAD section of an HTML page, outside any procedure	Visible in every procedure in the script

Variable Scope Prefixes

As script size grows, so does the value of being able to quickly differentiate the scope of variables. A one-letter scope prefix preceding the type prefix provides this, without unduly increasing the size of variable names.

Scope	Prefix	Example
Procedure-level	None	DblPrice
Script-level	S	SDblPrice

Descriptive Variable and Procedure Names

The body of a variable or procedure name should use mixed case and should be as complete as necessary to describe its purpose. In addition, procedure names should begin with a verb, such as InitNameArray or CloseDialog.

For frequently used or long terms, standard abbreviations are recommended to help keep name length reasonable. In general, variable names greater than 32 characters can be difficult to read. When using abbreviations, make sure they are consistent throughout the entire script. For example, randomly switching between Cnt and Count within a script or set of scripts may lead to confusion.

Object Naming Conventions

The following table lists recommended conventions for objects you may encounter while programming VBScript.

Code Commenting Conventions

All procedures should begin with a brief comment describing what they do. This description should not describe the implementation details (how it does it) because these often change over time, resulting in unnecessary comment maintenance work, or worse, erroneous comments. The code itself and any necessary inline comments describe the implementation.

Arguments passed to a procedure should be described when their purpose is not obvious and when the procedure expects the arguments to be in a specific range. Return values for functions and variables that are changed by a procedure, especially through reference arguments, should also be described at the beginning of each procedure.

Procedure header comments should include the following section headings. For examples, see the following table that explains the section heading and the comment contents to be inserted while developing code.

Section Heading	Comment Contents
Purpose	What the procedure does (not how).
Assumptions	List of any external variable, control, or other element whose state affects this procedure.
Effects	List of the procedure's effect on each external variable, control, or other element.
Inputs	Explanation of each argument that is not obvious. Each argument should be on a separate line with inline comments.
Return Values	Explanation of the value returned.

Remember the Following Points:

Every important variable declaration should include an inline comment describing the use of the variable being declared.
Variables, controls, and procedures should be named clearly enough that inline comments are only needed for complex implementation details.

At the beginning of your script, you should include an overview that describes the script, enumerating objects, procedures, algorithms, dialog boxes, and other system dependencies. Sometimes a piece of pseudocode describing the algorithm can be helpful.

Formatting Your Code

Screen space should be conserved as much as possible, while still allowing code formatting to reflect logic structure and nesting. Here are a few pointers:

- Standard nested blocks should be indented four spaces.
- The overview comments of a procedure should be indented one space.
- The highest level statements that follow the overview comments should be indented four spaces, with each nested block indented an additional four spaces. For example, see the code in Figure 5.19 below:

```
'*********************************************************
' Purpose:  Locates the first occurrence of a specified user
'          in the UserList array.
' Inputs:   strUserList():   the list of users to be searched.
'          strTargetUser:   the name of the user to search for.
' Returns:  The index of the first occurrence of the strTargetUser
'          in the strUserList array.
'          If the target user is not found, return -1.
'*********************************************************

Function intFindUser (strUserList(), strTargetUser)
   Dim i                ' Loop counter.
   Dim blnFound                  ' Target found flag
   intFindUser = -1
   i = 0                ' Initialize loop counter
   Do While i <= Ubound(strUserList) and Not blnFound
If strUserList(i) = strTargetUser Then
         blnFound = True   ' Set flag to True
         intFindUser = i   ' Set return value to loop count
      End If
      i = i + 1          ' Increment loop counter
   Loop
End Function
```

Check Your Progress 2

1. Write a menu - driven program in VBScript to check whether a number is even, odd and whether it is prime.

2. Write a procedure to check whether a string is a palindrome.

5.9 DICTIONARY OBJECT IN VBSCRIPT

The Dictionary object stores data as key, item pairs. A Dictionary object is the equivalent of a PERL associative array. Items, which can be any form of data, are stored in the array. Each item is associated with a unique key. The key is used to retrieve an individual item and is usually an integer or a string, but can be anything except an array.
The following code illustrates how to create and use a Dictionary object:

```
<HTML>
<HEAD><TITLE>Forms and Elements</TITLE></HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
  Dim d                'Create a Script Level variable
  Set d = CreateObject("Scripting.Dictionary")

Sub cmdAdd_OnClick
  d.Add "0", "Amar"    'Add some keys and items
  d.Add "1", "Bunty"
  d.Add "2", "Chaman"

  a = d.items
  For i = 0 To d.Count
    document.frmForm.Elements(i).Value = a(i)
  Next
End Sub

Sub cmdExist_onClick
  d.Add "a", "Amar"    'Add some  keys and items.
  d.Add "b", "Bunty"
  d.Add "c", "Chaman"
  If d.Exists("c") Then
        msgbox "C key exists."
  Else
    msgbox "C key does not exist."
  End If
End Sub

Sub cmdKey_onClick
  d.Add "a", "Amar"    'Add some keys and items.
  d.Add "b", "Bunty"
  d.Add "c", "Chaman"
  d.Key("c") = "d"       'Set key for "c" to "d".
End Function

End Sub
Sub cmdKeys_onClick
  d.Add "a", "Amar"    'Add some  keys and items.
  d.Add "b", "Bunty"
  d.Add "c", "Chaman"
  a = d.keys
  For i = 0 To d.Count
    document.frmForm.Elements(i).Value = a(i)
  Next
End Sub

Sub cmdRemove_onClick
  d.Add "a", "Amar"    'Add some  keys and items.
  d.Add "b", "Bunty"
  d.Add "c", "Chaman"

  d.Remove("b")
  a = d.keys
  For i = 0 To d.Count
    document.frmForm.Elements(i).Value = a(i)
  Next
```

Comment: Shoud not the </HEAD> be after the </SCRIPT>?

Comment: Where is the Function beginning? The code listings should be tested out using copy and paste.

```
End Sub

Sub cmdCompareMode_onClick
   d.CompareMode = vbTextCompare
   d.Add "a", "Amar"      'Add some keys and items.
   d.Add "b", "Bunty"
   d.Add "c", "Chaman"
   d.Add "B", "Baltimore"   'Add method fails on this line because the
                            'letter b already exists in the Dictionary.
End Sub
-->

</SCRIPT>
<BODY>
<CENTER>
<FORM NAME="frmForm"><p>
        <Input Type = "Text"><p>
        <Input Type = "Text"><p>
        <Input Type = "Text"><p>
        <Input Type = "Button" NAME="cmdAdd" VALUE="Add">
        <Input Type = "Button" NAME="cmdExist" VALUE="Exist">
        <Input Type = "Button" NAME="cmdKeys" VALUE="Keys">
        <Input Type = "Button" NAME="cmdRemove" VALUE="Remove"><p>

        <Input Type = "Button" NAME="cmdCompareMode" VALUE="Compare
Mode">
        <Input Type = "Button" NAME="cmdKey" VALUE="Key">

</FORM>
</CENTER>
</BODY>
</HTML>
```

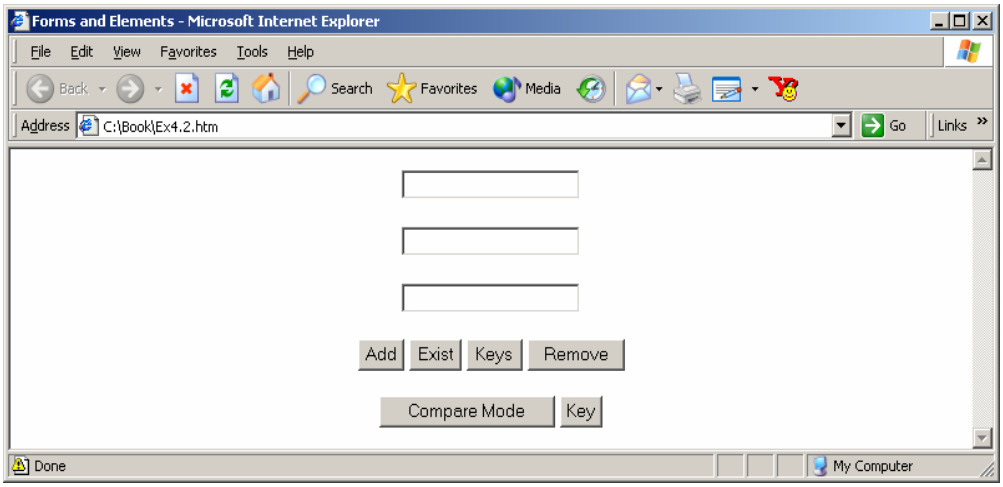


Figure 5.1: Code Using the Dictionary Object

5.9.1 Methods: VBScript Dictionary Object

Method	Description
Add Method	Adds a key, item pair.
Exists Method	Indicates if a specified key exists.
Items Method	Returns an array containing all items in a Dictionary object.
Keys Method	Returns an array containing all keys in a Dictionary object.
Remove Method	Removes a key, item pair.Properties: VBScript Dictionary Object

Add Method

Syntax	object.Add key, item
Object	The name of a Dictionary object. Required
Key	The key associated with the item being added. Required.
Item	The item associated with the key being added. Required.
Remarks	An error occurs if the key already exists.

Refer to Figure 5.20 for the example. In the cmdAdd_onClick procedure this method is used to add the Keys and Items to the Dictionary object.

Exists Method

Returns True if a specified key exists in the Dictionary object, False if it does not.

Syntax	object.Exists(key)
Object	The name of a Dictionary object. Required.
Key	The key value being searched for in the Dictionary object. Required.

Refer to Figure 5.20 for the example. In the cmdExists_onClick procedure this method is used to check the existence of any Key value in the Dictionary object.

Items Method

Returns an array containing all the items in a Dictionary object.

Syntax	object.Items
Object	The name of a Dictionary object. Required.

Refer to Figure 5.20 for the example code. In the cmdAdd_onClick procedure this method is used to retrieve the items in the array variable.

Keys Method

Returns an array containing all existing keys in a Dictionary object.

Syntax	object.Keys
Object	The name of a Dictionary object. Required.

Refer to Figure 5.20 for example code. In the cmdKeys_onClick procedure this method is used to retrieve the Keys in the array variable.

Remove Method

Removes a key, item pair from a Dictionary object. An error occurs if the specified key, item pair does not exist.

Syntax	object.Remove(key)
Object	The name of a Dictionary object. Required
Key	The Key associated with the key-item pair you want to remove from the Dictionary object. Required

Refer to Figure 5.20 for example code. In the cmdRemove_onClick procedure this method is used to remove the item from the dictionary object.

5.9.2 VBScript Dictionary Object Properties

Just like normal objects the Dictionary object also has certain properties. These properties can be set to any valid value and can be retrieved as and when required.

Property	Description
CompareMode Property	The comparison mode for string keys.
Count Property	The number of items in a Dictionary object.
Item Property	An item for a key.
Key Property	A key Syntax: VBScript Dictionary Object Scripting.Dictionary

CompareMode Property

Sets and returns the comparison mode for comparing string keys in a Dictionary object.

Syntax: object.CompareMode[= *compare*]

The CompareMode property has the following parts:

Part	Description
Object	Required. Always the name of a Dictionary object.
Compare	Optional. If provided, *compare* is a value representing the comparison mode used by functions such as StrComp.

The *compare* argument has the following settings:

Constant	Value	Description
VbBinaryCompare	0	Perform a binary comparison.
vbTextCompare	1	Perform a textual comparison.

Remarks:

Values greater than 2 can be used to refer to comparisons using specific Locale IDs (LCID). An error occurs if you try to change the comparison mode of a Dictionary object that already contains data.

The CompareMode property uses the same values as the *compare* argument for the StrComp function.

Refer to Figure 5.20 for example code. In the cmdCompareMode_onClick procedure this property is used to set the compare mode to Text, so that no two text keys that are the same can be added.

Count Property

Returns the number of items in a collection or Dictionary object. This property can only be read and cannot be set directly.

Syntax	object.Count
Object	The name of a Dictionary object. Required.

Refer to Figure 5.20 for example code. In the cmdAdd_onClick procedure this property is used to retrieve the number of items stored in the Dictionary object.

Item Property

Sets or returns an *item* for a specified *key* in a Dictionary object. For collections, returns an *item* based on the specified *key*. This property can be retrieved or set.

Syntax: object.Item(key) [= newitem]

The Item property has the following parts:

Part	Description
object	Required. Always the name of a collection or Dictionary object.
key	Required. *Key* associated with the *item* being retrieved or added.
newitem	Optional. Used for Dictionary object only; no application for collections. If provided, *newitem* is the new value associated with the specified *key*.

Remarks:

If *key* is not found when changing an *item*, a new *key* is created with the specified *newitem*. If *key* is not found when attempting to return an existing item, a new *key* is created and its corresponding item is left empty.

Key Property

Sets a key in a Dictionary object. If key is not found when changing a key, an error will occur.

Syntax: *object*.Key(*key*) = *newkey*

The **Key** property has the following parts:

Part	Description
object	Required. Always the name of a **Dictionary** object.
Key	Required. *Key* value being changed.
newkey	Required. New value that replaces the specified *key*.

Refer to Figure 5.20 for example code. In the cmdKey_onClick procedure this property is used to change the key value from “C” to “D”.

5.10 ERR OBJECT

The VBScript Err Object contains information about run-time errors. The properties of the Err object are set by the generator of an error - Visual Basic, an Automation object, or the VBScript programmer.

The default property of the Err object is Number. Err.Number contains an integer and can be used by an Automation object to return an SCODE.

When a run-time error occurs, the properties of the Err object are filled with information that uniquely identifies the error and information that can be used to handle it. To generate a run-time error in your code, use the VBScript Err Object Raise Method. The Err object's properties are reset to zero or zero-length strings ("") after an On Error Resume Next statement. The VBScript Err Object Clear Method can be used to explicitly reset Err object. The Err object is an intrinsic object with global scope - there is no need to create an instance of it in your code.

Consider the following example, which displays some of the possible errors that can be raised explicitly to generate run time errors.

```
<HTML>
<HEAD>

<TITLE>IGNOU</TITLE>

<SCRIPT LANGUAGE="VBScript">

<!--

Sub cmdSubmit_OnClick
```

```
On Error resume Next
' Check to see if the user entered anything.
  If (Len(document.form1.txtAge.value) = 0) Then
    MsgBox "You must enter your age before submitting."
    Exit Sub
  End If
' Check to see if the user entered a number.
  If (Not(IsNumeric(document.form1.txtAge.value ))) Then
    MsgBox "You must enter a number for your age."
    Exit Sub
  End If
' Check to see if the age entered is valid.
  If (document.form1.txtAge.value <= 0) Or  (document.form1.txtAge.value > 100)
Then
    MsgBox "The age you entered is invalid."
    Exit Sub
  End If
' Data looks okay so submit it.
  MsgBox "Thanks for providing your age."
  document.form1.submit
End Sub
Sub cmdtype_OnClick
On Error Resume Next
            for i = 1 to 35
            Err.clear
            Err.Raise i
              MsgBox Err.description & " - Error number - " &i
            Next
End Sub
-->
</SCRIPT>
</HEAD>
<BODY  bgColor="#ffffcc" Text="#000099" >
<H1>Error Handling and validations</H1>
<B><P> This example demonstrates validation techniques and Error handling in
VBScript. </P><?B>
<B>Please enter the age between 1 and 100. Otherwise, an error message would be
```

```
flashed on clicking the submit button.</B>
<FORM NAME="form1">
<TABLE>
<TR>
<TD>Enter your age:</TD>
<TD><INPUT TYPE="Text" NAME="txtAge" SIZE="2"></TD>
</TR>
<TR>
<TD><INPUT TYPE="Button" NAME="cmdSubmit" VALUE="Submit"></TD>
<TD></TD>
<TD><INPUT TYPE="Button" NAME="cmdtype" VALUE="      Type Of Errors
"></TD>
    <TD></TD>
</TR>
 </TABLE>
</FORM>
</BODY>
</HTML>
```

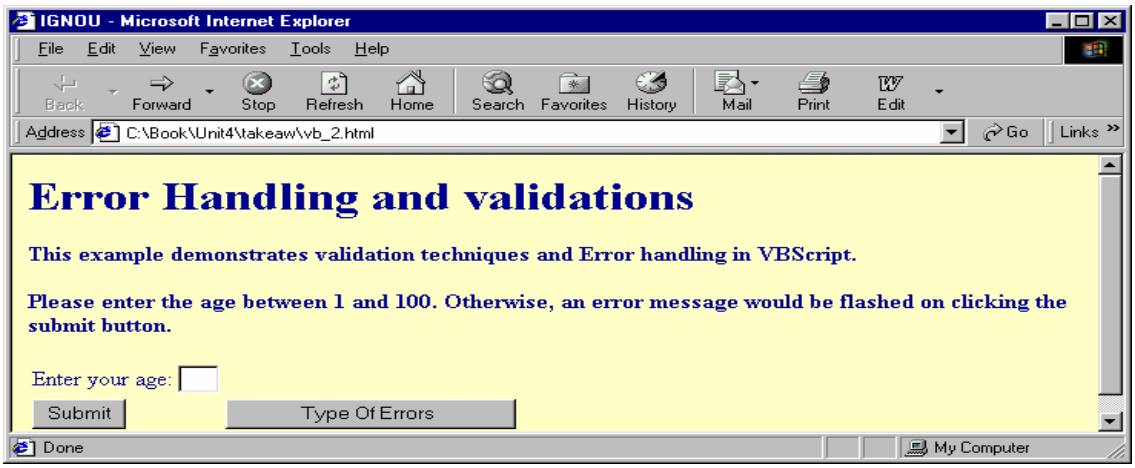


Figure 5.2: Error Handling in VBScript

5.10.1 Methods: VBScript Err Object

Clear Method	Clears all property settings.
Raise Method	Generate a run-time error.

- **VBScript Err Object Clear Method**

  Syntax : Err.Clear
  Clears all property settings of the Err object. Use Clear to explicitly clear the Err object after an error has been handled. This is necessary, for example, when you use deferred error handling with On Error Resume Next. VBScript calls the Clear method automatically whenever any of the following statements are executed:

  * On Error Resume Next
  * Exit Sub
  * Exit Function

- **VBScript Err Object Raise Method**

  Syntax: Err.Raise(number, source, description, helpfile, helpcontext)
  The On Error Resume Next statement, also called an error handler, is a procedure-level statement. It only remains in effect within the procedure that contains the on error declaration. The VBScript interpreter, like many

languages, raises an error to higher calling levels until it finds a procedure that handles the error. If none is found, the script halts and the user is presented with the error results by the interpreter. The Raise method is used for generating run-time errors. All the arguments of this method are optional except Number. However, if you use Raise, without specifying some arguments, and the property settings of the Err object contain values that have not been cleared, those values become the values for your error. When setting the number property to your own error code in an Automation object, you add your error code number to the constant vbObjectError. For example, to generate the error number 1050, assign vbObjectError + 1050 to the number property.

Arguments	Description
Number	A Long integer subtype that identifies the nature of the error. VBScript errors (both VBScript-defined and user-defined errors) are in the range 0-65535.
Source	A string expression naming the object or application that originally generated the error. When setting this property for an Automation object, use the form project.class. If nothing is specified, the programmatic ID of the current VBScript project is used.
Description	A string expression describing the error. If unspecified, the value in number is examined. If it can be mapped to a VBScript run-time error code, a string provided by VBScript is used as the description. If there is no VBScript error corresponding to number, a generic error message is used.
Helpfile	The fully qualified path to the Help file in which help on this error can be found. If unspecified, VBScript uses the fully qualified drive, path, and file name of the VBScript Help file.
Helpcontext	The context ID identifying a topic within helpfile that provides help for the error. If omitted, the VBScript Help file context ID for the error corresponding to the number property is used, if it exists

5.10.2 Properties: VBScript Err Object

Properties	Description
Description Property	The descriptive string associated with an error.
HelpContext Property	A context ID for a topic in a Windows help file.
HelpFile Property	A fully qualified path to a Windows help file.
Number Property	A numeric value identifying an error.
Source Property	The name of the object or application that originally generated the error.

- **Description Property**

  Returns or sets a descriptive string associated with an error. The Description property consists of a short description of the error. Use this property to alert the user to an error that you cannot or do not want to handle. When generating a user-defined error, assign a short description of your error to this property. If Description is not filled in, and the value of theVBScript Err Object Number Property corresponds to a VBScript run-time error, the descriptive string associated with the error is returned.

Syntax	Err.Description [= *stringexpression*]
Argument : Stringexpression	A string expression containing a description of the error.

- **HelpContext Property**

  Sets or returns a context ID for a topic in a Help File. If a Help file is specified

in the VBScript Err Object HelpFile Property, the HelpContext property is used to automatically display the Help topic identified. If both HelpFile and HelpContext are empty, the value of the VBScript Err Object Number Property is checked. If it corresponds to a VBScript run-time error value, then the VBScript Help context ID for the error is used. If the Number property does not correspond to a VBScript error, the contents screen for the VBScript Help file is displayed.

Syntax	Err.HelpContext [= *contextID*]
Argument : contextID	A valid identifier for a Help topic within the Help file. Optional.

- **HelpFile Property**

  Sets or returns a fully qualified path to a Help File. If a Help file is specified in HelpFile, it is automatically called when the user clicks the Help button (or presses the F1 key) in the error message dialog box. If the VBScript Err Object HelpContext Property contains a valid context ID for the specified file, that topic is automatically displayed. If no HelpFile is specified, the VBScript Help file is displayed.

Syntax	Err.HelpFile [= *contextID*]
Argument : contextID	The fully qualified path to the Help file. Optional.

- **Number Property**

  Returns or sets a numeric value specifying an error. Number is the Err object's default property. When returning a user-defined error from an Automation object, set Err.Number by adding the number you selected as an error code to the constant vbObjectError. For example, you use the following code to return the number 1051 as an error code:

```
Err.Raise Number:= vbObjectError + 1051, Source:= "SomeClass"
```

Syntax	Err.Number [= *errornumber*]
Argument : errornumber	An integer representing a VBScript error number or an SCODE error value.

- **Source Property**

  Returns or sets the name of the object or application that originally generated the error. The Source property specifies a string expression that is usually the class name or programmatic ID of the object that caused the error. Use Source to provide your users with information when your code is unable to handle an error generated in an accessed object. For example, if you access Microsoft Excel and it generates a Division by zero error, Microsoft Excel sets the VBScript Err Object Number Property to its error code for that error and sets Source to Excel application. Note that if the error is generated in another object called by Microsoft Excel, Excel intercepts the error and sets Err.Number to its own code for Division by zero. However, it leaves the other Err object (including Source) as set by the object that generated the error.

  Source always contains the name of the object that originally generated the error - your code can try to handle the error according to the error documentation of the object you accessed. If your error handler fails, you can use the Err object information to describe the error to your user, using Source and the other Err to inform the user which object originally caused the error, a description of the error, and so forth.

Syntax	Err.Source [= *stringexpression*]
Argument : stringexpression	A string expression representing the application that generated the error.

Case Study: Design a web page, which is used for accepting information regarding stocks. Make sure the form performs the possible validations. The simple example code below merely shows the form and accepts input but does not store the data or perform any action.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--


Sub cmd_submit_OnClick
' To continue the flow if any error occurs
on Error Resume Next

If document.form1.TxtName.Value = "" or  document.form1.TxtEMail.Value = ""
Then
Dim MyMessage
MyMessage = "Please enter your name and e-mail address."
MsgBox MyMessage, 0, "Incomplete Form Error"
   document.form1.TxtName.focus()
End If

' Check to see if the user entered anything.

  If (Len(document.form1.txtAge.value) = 0) Then

    MsgBox "You must enter your age before submitting."
    Exit Sub
    document.form1.txtAge.focus()


  End If
  If (Not(IsNumeric(document.form1.txtAge.value))) Then

    MsgBox "You must enter a number for your age."

    Exit sub

  End If

' Check to see if the age entered is valid.

  If (document.form1.txtAge.value < 0) Or  (document.form1.txtAge.value > 100)
Then

'    MsgBox "The age you entered is invalid."
          Err.clear
          Err.Raise 6
          Err.description = "Error - The age you entered is invalid."
          msgbox Err.description
    Exit Sub

  End If
```

```
' Data looks okay so submit it.

  MsgBox "Thanks for sharing your views."

  document.form1.submit

End Sub
-->

</SCRIPT>
</HEAD>
<BODY  bgColor="#ffffcc" Text="#000099" Link="#ff0000" VLink="#ff0000"
ALink="#009900">

<!--ENDOFADD -->
<FORM name = "form1" >
<CENTER>
<TABLE BORDER=8 CELLSPACING=1 CELLPADDING=0
WIDTH="600"><TR><TD>
<TABLE BORDER=5 CELLSPACING=1 CELLPADDING=0
WIDTH=100%><TR><TD>
<FONT SIZE=""  FACE="Arial"><B><CENTER><H2>Sample Stock
Survey</H2></CENTER></B></FONT></TD>
<A NAME="ItemAnchor1"></A>
</TR></TABLE>
<HR>
<TABLE BORDER=5 CELLSPACING=1 CELLPADDING=0 WIDTH=100%>
<TR>
<TD ALIGN=RIGHT>*Name:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtName">
</TD>
<TD ALIGN=RIGHT>*E-Mail:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtEMail"></TD>
</TR>
<TR><TD ALIGN=RIGHT>Address:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtAddress"></TD>
<TD ALIGN=RIGHT>Age:</TD>
<TD><INPUT TYPE="TEXT" NAME="txtAge"></TD>
<A NAME="ItemAnchor1"></A>
</TR></TABLE>

<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0
WIDTH=100%><TR><TD>
<CENTER>
<FONT SIZE=""  FACE="Arial">
<CENTER>
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0"
WIDTH="75%">
<BR></CENTER>

</FONT></CENTER></TD>
<TD>
<FONT SIZE=""  FACE="Arial">
</TR>
</TABLE>

<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0>
<TR>
```

```
<TD>
<TABLE BORDER=0><TR><TD  VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
<B>Describe your investment experience</B>
</FONT>
</TD>
</TR>
</TABLE>


<TABLE BORDER=0>
<TR>
<TD VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
<INPUT TYPE="RADIO" NAME="RESULT_RadioButton-3" VALUE="Radio-
0">beginner
</FONT>
</TD>
<TD VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
<INPUT TYPE="RADIO" NAME="RESULT_RadioButton-3" VALUE="Radio-
1">intermediate
</FONT>
</TD>
<TD VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
<INPUT TYPE="RADIO" NAME="RESULT_RadioButton-3" VALUE="Radio-
2">expert
</FONT>
</TD>
<TD VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
</FONT>
</TD>
</TR>
</TABLE>
</TD>
<A NAME="ItemAnchor4"></A>
</TR>
</TABLE>
<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0>
<TR>
<TD>
<TABLE BORDER=0>
<TR>
<TD>
<FONT  SIZE="2"  FACE="Arial" >
<B>Types of Investments you make</B>
</FONT>
</TD>
</TR>
</TABLE>
<TABLE BORDER=0><TR><TD  VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
<INPUT TYPE="CHECKBOX" NAME="RESULT_CheckBox-4"
VALUE="CheckBox-0">Individual Stocks<BR>
<INPUT TYPE="CHECKBOX" NAME="RESULT_CheckBox-4"
VALUE="CheckBox-1">Options<BR>
```

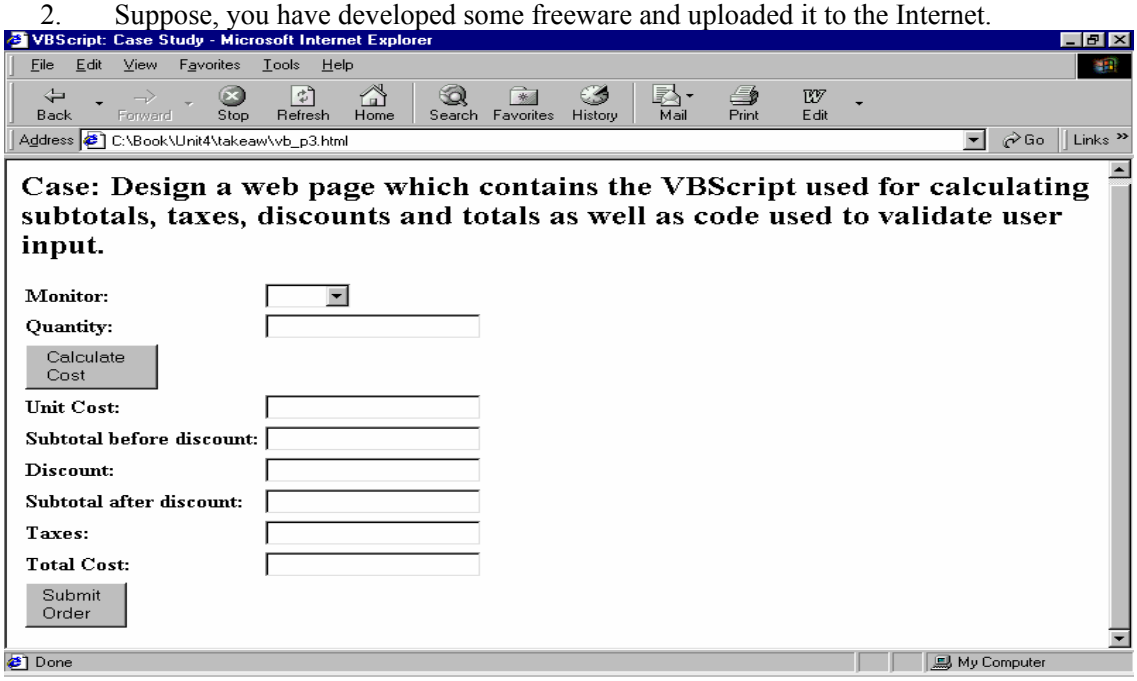
```
<INPUT TYPE="CHECKBOX" NAME="RESULT_CheckBox-4"
VALUE="CheckBox-2">Mutual Funds<BR>
<INPUT TYPE="CHECKBOX" NAME="RESULT_CheckBox-4"
VALUE="CheckBox-3">Real Estate
</FONT>
</TD>
</TR>
</TABLE>
</TD>
<A NAME="ItemAnchor5"></A>
</TR>
</TABLE>
<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0>
<TR>
<TD>
<TABLE BORDER=0>
<TR>
<TD  VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
<B>How do you buy your stocks?</B>
</FONT>
</TD>
</TR>
</TABLE>
<TABLE BORDER=0><TR><TD VALIGN="TOP">
<FONT  SIZE="2"  FACE="Arial" >
<!--DROP_DOWN_TYPE -->
<SELECT NAME="RESULT_RadioButton-5">
<OPTION>
</OPTION>
<!--DROP_DOWN_TYPE NAME="RESULT_RadioButton-5" VALUE="Radio-0" -
-><OPTION>1) On-Line</OPTION>
<!--DROP_DOWN_TYPE NAME="RESULT_RadioButton-5" VALUE="Radio-1" -
-><OPTION>2) Touch Tone Trading
</OPTION>
<!--DROP_DOWN_TYPE NAME="RESULT_RadioButton-5" VALUE="Radio-2" -
-><OPTION>3) Broker Assisted
</OPTION>
<!--DROP_DOWN_TYPE NAME="RESULT_RadioButton-5" VALUE="Radio-3" -
-><OPTION>4) Other</OPTION>
</SELECT>
</FONT>
</TD>
</TR>
</TABLE>
</TD>
<A NAME="ItemAnchor6"></A>
</TR>
</TABLE>
<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0>
<TR><TD>
<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0>
<TR><TD>
<FONT  SIZE="2"  FACE="Arial" >
<B>What is your hot stock pick for this year?</B>
</FONT></TD>
</TR><TR>
<TD>
```

```
<FONT  SIZE="2"  FACE="Arial" >
<INPUT TYPE="TEXT" NAME="RESULT_TextField-6" SIZE="30"
MAXLENGTH="30">
</FONT>
</TD></TR>
</TABLE>
</TD>
<A NAME="ItemAnchor7"></A>
<TD>
<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0><TR><TD>
<FONT  SIZE="2"  FACE="Arial" >
<B>Stock Symbol if you know it</B>
</FONT>
</TD>
</TR><TR>
<TD>
<FONT  SIZE="2"  FACE="Arial" >
<INPUT TYPE="TEXT" NAME="RESULT_TextField-7" SIZE="4"
MAXLENGTH="4">
</FONT>
</TD></TR>
</TABLE>
</TD>
<A NAME="ItemAnchor8"></A>
</TR>
</TABLE>
<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0>
<TR><TD>
<TABLE BORDER=0 CELLSPACING=1 CELLPADDING=0>
<TR><TD VALIGN="BOTTOM">
<FONT  SIZE="2"  FACE="Arial" >
<B>Any Investment Advice for others?</B>
</FONT>
</TD>
</TR><TR>
<TD>
<FONT  SIZE="2"  FACE="Arial" >
<TEXTAREA NAME="RESULT_TextArea-8" ROWS="7" COLS="35"
WRAP="SOFT"></TEXTAREA>
</FONT></TD></TR>
</TABLE>
</TD>
<A NAME="ItemAnchor9"></A>
</TD></TR><TR><TD>


</CENTER>
</TD></TR>
</TABLE>
</TD></TR>
</TABLE>
</CENTER>
<BR>
<CENTER>
<INPUT TYPE="SUBMIT" NAME="cmd_submit" VALUE="Submit">
</CENTER>
</FORM>
</BODY>
```

</HTML>

Check Your Progress 3

1. Design the following web page with all Validations and Runtime error handling:
2. Suppose, you have developed some freeware and uploaded it to the Internet.



Before downloading the software, the user has to fill up and submit an introductory form. Design the form as shown with validation of the inputs **(Case Study).**

5.11 SUMMARY

In this unit you have learned how to create Web pages that use VBScript. You have learned to add logic to your Web pages as with any other programming application. You have also learned Object support in VBScript. One of the very common and often used objects, the Dictionary Object has been discussed in detail. Validations form an important part of forms, which have been shown in many of the examples in this unit. You have also learned how to use the Err object for handling Runtime errors. You will now be able to develop Web pages like Enquiry forms, admission forms and so on. You have learned about the different operators and datatypes supported by VBScript and how to use them. You have learnt about functions and procedures, and how to make your programs modular using them. Coding conventions should be followed in order to make your programs readable and easily understandable by others. These include indentation of the code, naming the variables and constants and other formatting conventions such as those for comments.

5.12 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. The following code will do the job:

```
<SCRIPT LANGUAGE=vbscript>
<!--
```

```
Function IsPrime(str)
str=trim(str)
Dim bln,ctr
bln=true
ctr=1

if(cint(str)<1)then
IsPrime=false
exit function
end if

if(cint(str)=1 or cint(str)=2 or cint(str)=3)then
IsPrime=true
exit function
end if

for ctr=cint(str)-1 to 2 step -1
if(cint(str)mod(ctr)=0)then
bln=false
end if
next

IsPrime=bln

End Function

Function IsEven(str)
str=trim(str)
if(str="0")then
IsEven=true
exit function
end if
if(cint(str)mod(2)=0)then
IsEven=true
else
IsEven=false
end if
End Function

Sub main()
str=text1.value
Dim blnEven,blnPrime

if not(isnumeric(str))then
msgbox "Not numeric"
exit sub
end if

if(IsEven(str))then
msgbox "Even"
else
msgbox "Odd"
end if

if(IsPrime(str))then
msgbox "prime"
else
msgbox "not prime"
```

```
end if
End Sub
}

//-->
</SCRIPT>
```

Comment: It would be better to give driver code so that the script can be checked out. The same holds for other answers as well.

2. Use the following function to generate the fibonnacci series.

```
FUNCTION fib (n)
if (N=0) or (N=1) then
Fib = 1
  ElseFib = Fib (N- 1) + Fib (N-2)
END If
END FUNCTION
```

3. Use the following function to find the factorial

```
Function Factorial(n)
 Dim i
 If n < 0 Then
  Factorial = 0
  Exit Function
 End If
 Factorial = 1
 For i = 2 To n
  Factorial = Factorial * i
 Next
End Function
```

Check Your Progress 2

1.
```
<HTML>
   <HEAD>
   <META NAME="GENERATOR" Content="Microsoft Visual Studio
   6.0">
   <TITLE></TITLE>
   <SCRIPT LANGUAGE=vbscript>
   <!--

   Function IsPrime(str)
   str=trim(str)
   Dim bln,ctr
   bln=true
   ctr=1

   if(cint(str)<1)then
   IsPrime=false
   exit function
   end if

   if(cint(str)=1 or cint(str)=2 or cint(str)=3)then
   IsPrime=true
   exit function
   end if

   for ctr=cint(str)-1 to 2 step -1
   if(cint(str)mod(ctr)=0)then
```

```
bln=false
end if
next

IsPrime=bln

End Function

Function IsEven(str)
str=trim(str)
if(str="0")then
IsEven=true
exit function
end if
if(cint(str)mod(2)=0)then
IsEven=true
else
IsEven=false
end if
End Function

Sub mysub()
str=text1.value

if not(isnumeric(str))then
msgbox "Not numeric"
exit sub
end if

if(rad(0).checked)then
if(IsEven(str))then
msgbox "Yes it is even"
else
msgbox "No it is not even"
end if
Exit sub
end if

if(rad(1).checked)then
if(IsEven(str))then
msgbox "No it is not odd"
else
msgbox "Yes it is odd"
end if
Exit sub
end if

if(rad(2).checked)then
if(IsPrime(str))then
msgbox "Yes it is a prime number"
else
msgbox "No it is not a prime number"
end if
Exit sub
end if

End Sub
```

```
//-->
</SCRIPT>
</HEAD>
<BODY>
<P>

Even

<INPUT type=radio name="rad" checked value="even">
Odd
<INPUT type=radio name="rad" value="odd">
Prime
<INPUT type=radio name="rad" value="prime">

</P>
<P><INPUT id=text1 name=text1></P>
<P><INPUT id=button1 type=button value=Button name=button1
LANGUAGE=javascript onclick="mysub()"></P>

</BODY>
</HTML>
```

2. Following is the code:

```
Function IsPalindrome(str)
Dim iStart,iEnd,ctr,blnPalin
str=trim(str)
blnPalin=true
iEnd=len(str)
iCnt=round(iEnd / 2)
iStart=1
for ctr=1 to cint(iCnt)
if(mid(str,iEnd,1)<>mid(str,iStart,1))then
IsPalindrome=false
Exit Function
end if
iStart=cint(iStart)+1
iEnd=cint(iEnd)-1
next
if(blnPalin=true)then
IsPalindrome=true
else
IsPalindrome=false
end if
End Function

Sub blankStr()
str=text1.value
if(trim(str)="")then
msgbox "Pls. enter a number"
exit sub
end if

if(IsPalindrome(str))then
msgbox "Yes"
else
msgbox "No"
```

```
end if

End Sub
```

Check Your Progress 3

1.

```
<HTML>

<H2>  Case: Design a web page which contains the VBScript used for
calculating subtotals,
  taxes, discounts and totals as well as code used to validate user
input.</H2>

<HEAD>
<TITLE>VBScript: Case Study</TITLE>

<SCRIPT LANGUAGE="VBScript">
<!-- Add this to instruct non-IE browsers to skip over VBScript modules.
Option Explicit

Sub cmdCalculate_OnClick
  Dim AmountofDiscount
  Dim AmountofTax
  Dim DISCOUNT_LIMIT
  Dim DISCOUNT_RATE
  Dim SubtotalBefore
  Dim SubtotalAfter
  Dim TAX_RATE
  Dim TotalCost

  If (Len(Document.frmCaseStudy.txtQuantity.Value) = 0) Then
    MsgBox "You must enter a quantity."
    Exit Sub
  End If

  If (Not IsNumeric(Document.frmCaseStudy.txtQuantity.Value)) Then
    MsgBox "Quantity must be a numeric value."
    Exit Sub
  End If

  If (Len(Document.frmCaseStudy.cmbProducts.Value) = 0) Then
    MsgBox "You must select a product."
    Exit Sub
  End If


  DISCOUNT_LIMIT = 1000
  DISCOUNT_RATE = .10
  TAX_RATE = 0.06
' Calculate the subtotal for the order.
 SubtotalBefore = Document.frmCaseStudy.txtQuantity.Value *
Document.frmCaseStudy.lblUnitCost.value


  If (SubtotalBefore > DISCOUNT_LIMIT) Then
    AmountofDiscount = SubtotalBefore * DISCOUNT_RATE
  Else
    AmountofDiscount = 0
```

```
  End If
  SubtotalAfter = SubtotalBefore - AmountofDiscount

' Calculate taxes and total cost.
  AmountofTax = SubtotalAfter * TAX_RATE
  TotalCost = SubtotalAfter + AmountofTax

' Display the results.
  Document.frmCaseStudy.lblSubtotalBefore.value = SubtotalBefore
  Document.frmCaseStudy.lblDiscount.value = AmountofDiscount
  Document.frmCaseStudy.lblSubtotalAfter.value = SubtotalAfter
  Document.frmCaseStudy.lblTaxes.value = AmountofTax
  Document.frmCaseStudy.lblTotalCost.value = TotalCost

End Sub

Sub cmdSubmit_onClick
' Submit this order for processing.
  MsgBox "Your order has been submitted."
  Document.frmCaseStudy.Submit
End Sub

Sub cmbProducts_onchange()

   Select Case Document.frmCaseStudy.cmbProducts.SelectedIndex
    Case 1
      Document.frmCaseStudy.lblUnitCost.value = 1590
    Case 2
      Document.frmCaseStudy.lblUnitCost.value = 880
    Case 3
      Document.frmCaseStudy.lblUnitCost.value = 1940
    Case Else
      Document.frmCaseStudy.lblUnitCost.value = 0
  End Select
End Sub

-->
</SCRIPT>

</HEAD>

<BODY>
<FORM NAME="frmCaseStudy">
  <TABLE>
    <TR>
      <TD><B>Monitor:</B></TD>
      <TD>
  <SELECT NAME = "cmbProducts">
            <OPTION VALUE ="0" ></OPTION>
            <OPTION VALUE ="1" >Item 1</OPTION>
            <OPTION VALUE ="2">Item 2</OPTION>
            <OPTION VALUE ="3">Item 3</OPTION>
  </SELECT>
      </TD>
    </TR>
    <TR>
      <TD><B>Quantity:</B></TD>
      <TD>
```

```
  <INPUT TYPE = "TEXT" NAME ="txtQuantity" >
    </TD>
   </TR>
   <TR>
    <TD><INPUT TYPE="Button" NAME="cmdCalculate"
VALUE="Calculate
Cost"></TD>
    <TD></TD>
   </TR>
   <TR>
    <TD><B>Unit Cost:</B></TD>
    <TD>
  <INPUT TYPE = "TEXT" NAME ="lblUnitCost" >
    </TD>
   </TR>
   <TR>
    <TD><B>Subtotal before discount:</B></TD>
    <TD>
  <INPUT TYPE = "TEXT" NAME ="lblSubtotalBefore" >
    </TD>
   </TR>
   <TR>
    <TD><B>Discount:</B></TD>
    <TD>
  <INPUT TYPE = "TEXT" NAME = "lblDiscount" >
    </TD>
   </TR>
   <TR>
    <TD><B>Subtotal after discount:</B></TD>
    <TD>
  <INPUT TYPE = "TEXT" NAME ="lblSubtotalAfter" >
    </TD>
   </TR>
   <TR>
    <TD><B>Taxes:</B></TD>
    <TD>
  <INPUT TYPE = "TEXT" NAME ="lblTaxes" >
    </TD>
   </TR>
   <TR>
    <TD><B>Total Cost:</B></TD>
    <TD>
  <INPUT TYPE = "TEXT" NAME ="lblTotalCost" >
    </TD>
   </TR>
   <TR>
    <TD><INPUT TYPE="Button" NAME="cmdSubmit"
VALUE="Submit
Order"></TD>
    <TD></TD>
   </TR>
  </TABLE>
</FORM>

</BODY>

</HTML>
```

2.

```
<HTML>
<HEAD>
<TITLE>ABC Software's Registration Page</TITLE>
</HEAD>
<BODY BGCOLOR=WHITE> <CENTER>
<FONT SIZE=7 FACE="Arial" COLOR=BLUE>ABC</FONT>
<FONT SIZE=5 COLOR=BLACK>Registration</FONT>
</CENTER>
<HR COLOR="BLUE">
<FONT FACE="COMIC Sans MS"> Thank you for taking the time to
download and test our new product. In order to serve you well, we ask that you
submit the following information before downloading
<FONT FACE=ARIAL COLOR=BLUE>ABC</FONT>software.
<HR COLOR=RED WIDTH=75%> <CENTER>
<TABLE BORDER BORDERCOLOR="BLUE">
<TR>
<TD ALIGN=RIGHT>*Name:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtName">
</TD>
</TR>
 <TR>
<TD ALIGN=RIGHT>*E-Mail:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtEMail"></TD>
</TR>
<TR><TD ALIGN=RIGHT>Address:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtAddress"></TD>
</TR>
<TR>
<TD ALIGN=RIGHT>City:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtCity"></TD>
</TR>
 <TR>
<TD ALIGN=RIGHT>State:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtState"></TD>
</TR>
<TR>
<TD ALIGN=RIGHT>Zip:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtZip"></TD>
</TR>
 <TR>
<TD ALIGN=RIGHT>Country:</TD>
<TD><INPUT TYPE="TEXT" NAME="TxtCountry"></TD>
</TR>
<TR>
<TD COLSPAN=2 ALIGN=CENTER><INPUT TYPE="SUBMIT"
NAME="Btn1" VALUE="Send It!"><BR> <FONT SIZE=2>* Required
Field</FONT></TD>
</TR>
</TABLE>
<HR COLOR=RED WIDTH=75%>
<FONT SIZE=2>Thanks!</FONT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Btn1_OnClick
If TxtName.Value="" Or TxtEMail.Value="" Then
Dim MyMessage
MyMessage = "Please enter your name and e-mail address."
```

Comment: This displays the form but does not show error messages. Please correct.

```
MsgBox MyMessage, 0, "Incomplete Form Error"
End If

End Sub  -->
</SCRIPT>
</BODY>
</HTML>
```

5.13 FURTHER READINGS

1. *Learning VBScript* by Paul Lomax. Publisher: O'Reilly. Year: 2001
2. *Instant VBScript* by Alex Homer, Darren Gill. Publisher: Wrox Press Inc Pub. Year: 2003
3. *VBscript for the World Wide Web* by Paul Thurrott, Nolan Hester. Publisher: Peachpit Press. Year: 1999
4. *Teach Yourself Vbscript in 21 Days* by Keith Brophy, Timothy Koets. Publisher: Sams Publishing. year: 1996

Page 148: [1] Comment **milind**

Why repeat this question? The menu driven part is not the primary focus of the section gone by. The question does not significantly differ from question 1 of CYP 1.

Page 148: [2] Comment **milind**

This is fine if the students have done PERL earlier. Otherwise the reference to PERL can be deleted.

SOCIAS-IGNOU/P.O.10.5T/MAY, 2004