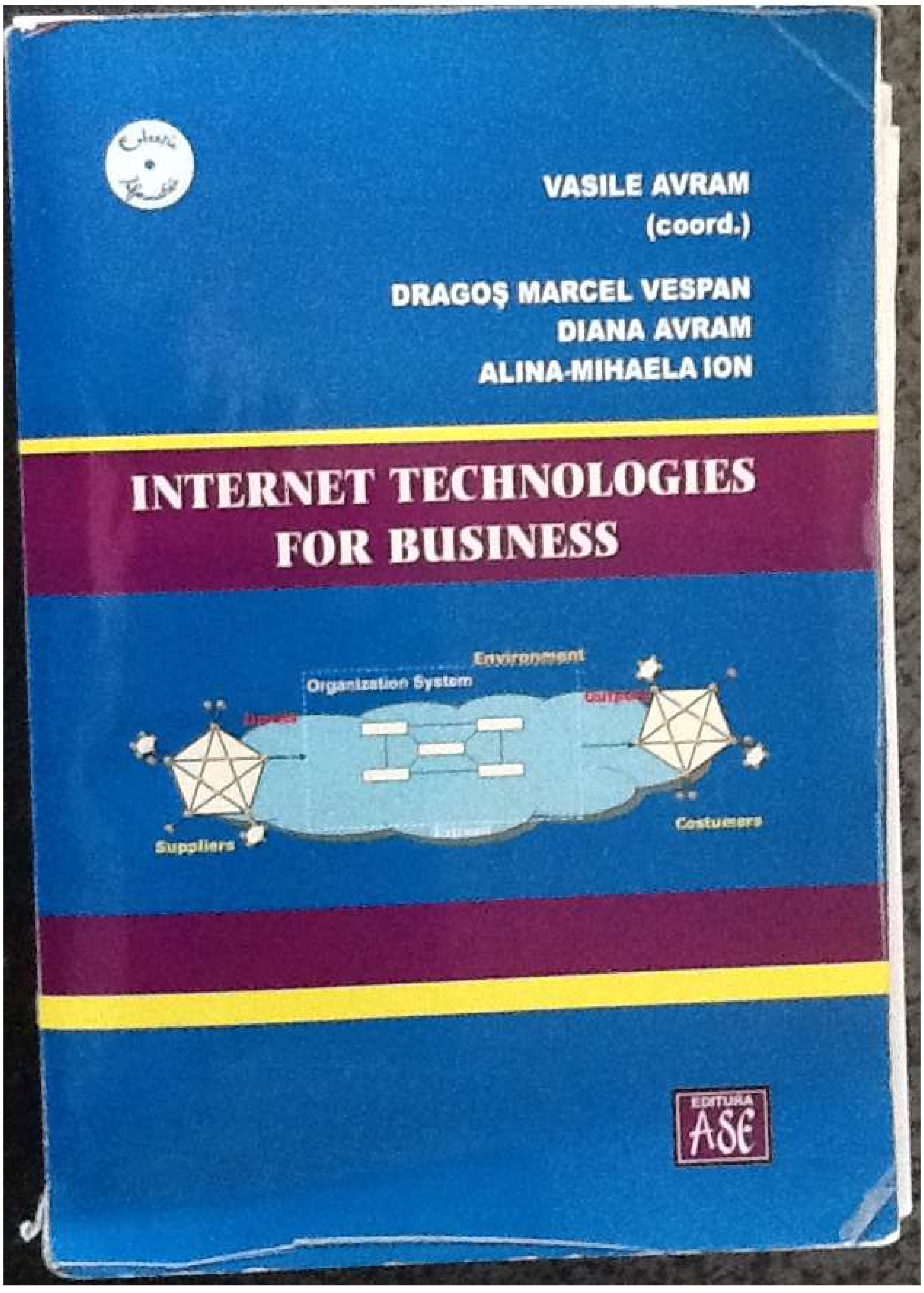


VASILE AVRAM
(coord.)

DRAGOŞ MARCEL VESPAN
DIANA AVRAM
ALINA-MIHAELA ION

INTERNET TECHNOLOGIES FOR BUSINESS

EDITURA ASE

INTERNET TECHNOLOGIES FOR BUSINESS

Contribuția autorilor la realizarea lucrării a fost următoarea / The authors contributions:
Prof. dr. Vasile Avram: capitol 1, 2, 3, capitol 4 paragrafele 4.1,4.2, 4.3.1, 4.3.2, 4.3.4, 4.5, 4.6.1, 4.7 capitol 6
Lect. dr. Dragoș-Marcel Vespan: capitol 5 paragrafele 5.1, 5.2, 5.4, 5.5, 5.6
Diana Avram: capitol 4 paragrafele 4.3.2.2, 4.3.3, 4.4, 4.6.2, 4.6.3, capitol 7
Prep. drd. Alina-Mihaela Ion: capitol 5 paragrafele 5.3, 5.7

ACADEMIA DE STUDII ECONOMICE BUCUREȘTI

Prof. dr. Vasile AVRAM
coordonator

Lect. dr. Dragoș-Marcel Vespan **Diana Avram**
Prep. drd. Mihaela-Alina Ion

INTERNET TECHNOLOGIES FOR BUSINESS

Editura ASE
București
2009

Editura ASE
Piața Romană nr. 6, sector 1, București, România
cod 010374
www.ase.ro
editura@ase.ro

Descrierea CIP a Bibliotecii Naționale a României
Internet technologies for business / Vasile Avram (coord.), Dragoș-Marcel Vespan, Diana Avram, Alina-Mihaela Ion. - București : Editura ASE, 2009
ISBN 978-606-505-274-1

I. Avram, Vasile
II. Vespan, Dragoș Marcel
III. Avram, Diana
IV. Ion, Alina-Mihaela

004.738.5:336

ISBN 978-606-505-274-1

Coperta: Livia Radu

Preface

Internet Technologies for Business has the goal of introducing and educating students in basic aspects of web organization and technologies. To achieve this goal, the text emphasizes subject matter organization and an interactive approach to generating student interest.

The text views web system as multilevel entity encompassing hardware, software systems, procedures, human elements, and organizational/societal impacts. The beginner student in internet technologies needs to gain knowledge of each of these levels trough a balanced approach, in which no level is receiving too little or too much emphasis. This text is carefully organized to achieve such a balance.

Chapter I is an introduction of various aspects of computer networks as foundations for the network of networks that is Internet. It introduces aspects regarding hardware components, topologies, protocols and standards, and interconnecting aspects.

Chapter II gives a closer look to Internet architecture, communication protocols and their position at web server side and/or client level together with an introduction in web pages, browsers and search engines.

Chapter III investigate the concepts and definitions used by Internet in relationship with business together with a description of business models and different point of views of classification of these.

Chapter IV is an investigation and classification of sites architectures together with and introduction to the required web technologies.

Chapter V is an extended presentation of the HTML language since this is the basic language designated to describe the web resources at client level (for their browser) and the almost technologies used to manipulate and generate answers to user requests uses this language as final end.
Chapter VI is an extended presentation of VBScript, as proprietary and platform dependent scripting language, for his client side scripting. It contains commented samples for the introduced notions the samples being those given for the Visual Basic to the course General Informatics and realized as VBScript solution to easy the understanding.
Chapter VII is an extended introduction to JavaScript as a platform independent scripting language together with commented samples for the introduced notions.

The student needs are not only to learn facts but also to work with them in an – describing, scripting, answering, evaluating, comparing – continuously process. Information should not just flow from instructor and text to student, but in other directions as well – from student to student, and from student to instructor.

The Authors

Contents

1 INTRODUCTION TO COMPUTER NETWORKS

1.1 LAN's & WAN's

Any system that requires computers to communicate with each other will need specialized hardware and software such as:

- modems;
- communication programs: contain standard protocols ensure that systems can signal to each other the start and finish of transmission and reception and any other problems experienced with data;
- network cards, network circuits and software.

"A network is nothing more than two or more computers connected to each other so that they can exchange information, such as e-mail messages or documents, or share resources, such as disk storage or printers [Lowe-05]".

A network is composed from a hardware part (servers, workstations, cables, printers and so on) and a software part (operating systems and applications).

A Local Area Network (LAN) links personal computers (Workstations) together so they can communicate and share resource, such as hard drives, printers, application software and data files.

All networks, even the most complex ones, include the same three fundamental blocks:

- devices that supplies services to the network;
- devices that uses that offered services;
- "something" that allow to communicate that devices.

The networks run under an operating system called Network Operating System (NOS) that must offer at least the following services:

- electronic mail (e-mail);
- sharing application software packages and files;
- central backup of files and application software;
- security;
- communication to remote Workstations (e.g. data can be transferred between networks; files can be created at a corporate national office and transferred electronically to each location and printing out), LAN's, and mini/mainframe connectivity.

Definition of LANs: LANs are networks of computers and peripherals linked together on a single geographical site. They located generally in one single building or in a group of buildings positioned on an array whose surface is up to some square kilometers.

LANs enabled multiple users in a relatively small geographical area to exchange files and messages, as well as access shared resources such as file servers and printers.

In a network context:

♦ a doctor's office can easily access the records on a poison control database maintained on a network as close as next door, or halfway around the world;

♦ a salesman can phone in to the network utilizing a microcomputer equipped with a modem from a remote location to place orders, for immediate access to inventory information;

♦ data can be transferred between networks;

♦ files can be created at a corporate national office, and transferred electronically to each location and printed out, especially important when timing is a factor.

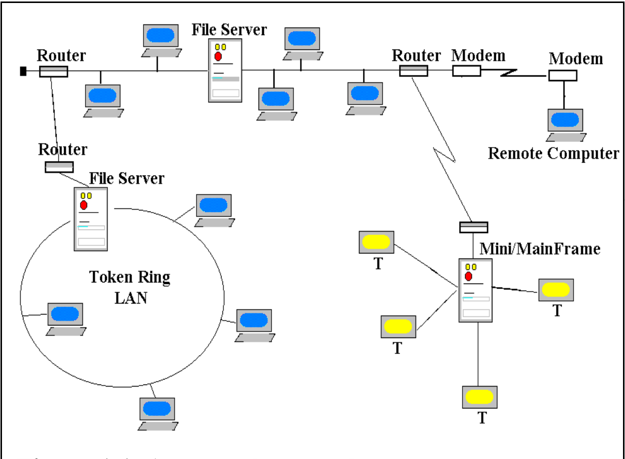


Figure 1.1 A network example

Network components consist of files, print or communication server, Workstations, Network Interface Cards (NIC), connectors, cables, wiring boxes and any other required hardware for the chosen topology.

All Workstations are connected to a File Server by intermediate of a network card and a transmission media (channel). All computer networks may consist of one or more File Servers. Data communication capabilities are available to allow you to connect to a remote PC, to another LAN or to a mainframe (figure 1.1)

The technology has produced a variety of tools to create inter-network solution. Such tools for connecting networks together include repeaters, bridges, routers, gateways, and so on.

Advantages of local area network:

- data can be shared by all users;
- the network can gradually extended as organization grows;
- users can share expensive resources such high quality printers;
- if one machine breaks down, the others can continue working;
- cost effective for large numbers of users;
- members of the network can send electronic mail to one another reducing the amount of paperwork.

Definition of WANs: A series of LANs joined together is often referred as WAN. WAN is a collection of LANs joined together over a large geographical area (county, country, region etc). Devices on each LAN communicate over the WAN using direct network connections or modems where direct connections are not available.

Wide-area networks (WANs) interconnect LANs with geographically dispersed users to create connectivity. Some of the technologies used for connecting LANs include T1, T3, ATM (Asynchronous Transfer Mode), ISDN (Integrated Services Digital Network), ADSL (Asymmetric Digital Subscriber Line), Frame Relay, radio links, etc.

A metropolitan area network (MAN) is a network that's smaller than a typical WAN but larger than a LAN, typically connecting the LANs within a same city.

A collection of interconnected networks is called an internetwork or internet. The individual networks are connected by intermediate of networking devices and the resulting assembly functions as a single large network. Internetworking refers to the industry, products, and procedures that meet the challenge of creating and administering internetworks.

1.2 Some network and internetwork components

File Server. A network starts with a server. For PCs networks is usually a high performance PC which can be installed to function as a file server only (dedicated) or as a combination File Server or Workstation (nondedicated). File Server means "serving" files to user’s request. It also handles the sharing of resources (e.g. disk drivers and printers) and security. Servers uses a Network Operating System (NOS) that must be enough capable to offer simultaneously services to most clients. The file server, utilizing NOS acts the same as a network traffic police which controls the Workstation file requests (reads and writes to network drives), printer output and communications between users and file servers attached to the network. The Operating System of the Network can be UNIX (and anyone of his clones, Linux for instance), MacOS, OS/2, Novell NetWare, IBM LAN Server, Banyan Vines, Windows NT xx Server, Windows 2003 Server, Windows 2008 Server etc. Network servers require large storage capacity, fast processors, and ample memory. Server tower cases also are larger than client computers and provide much more space for additional hard drives and other peripherals. Generally today medium-sized and large companies, uses many specialized servers that are organized, for efficiency reasons, as server blades. Another concept related to servers and used today is server virtualization or the ability to create a logical abstraction of physical assets (it allows for multiple “virtual” servers to run on one physical server, thereby consolidating many physical servers onto one).

A blade server is a server chassis housing multiple thin, modular electronic circuit boards, known as server blades (see the figures below). Each blade is a server in its own right, often dedicated to a single application. The blades are literally servers on a card, containing processors, memory, integrated network controllers, an optional fiber channel host bus adaptor (HBA) and other input/output (IO) ports.

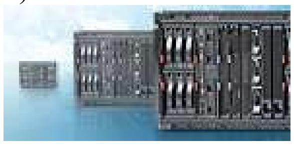
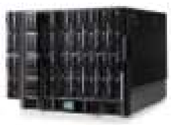
A blade server is sometimes referred to as a high-density server and is typically used in a clustering of servers that are dedicated to a single task, such as:

- File sharing;
- Web page serving and caching;
- SSL encrypting of Web communication;
- The transcoding of Web page content for smaller displays;

- Streaming audio and video content.

Blade servers allow more processing power in less rack space, simplifying cabling and reducing power consumption. According to a SearchWinSystems.com article on server technology, enterprises moving to blade servers can experience as much as an 85% reduction in cabling for blade installations over conventional 1U or tower servers. With so much less cabling, IT administrators can spend less time managing the infrastructure and more time ensuring high availability. (http://itknowledgeexchange.techtarget.com/)

Examples blade servers from main manufacturers (external views):

IBM **HP** **Dell**

Blade and virtualization technologies together provide critical building blocks for the today's enterprise data centers.

Workstation. All microcomputers with at least one disk drive have the possibility to (the ability to) function in "stand-alone" mode (the operating system and applications ca be stored and loaded from the disk drive). When they are connected to network, they become a (*Work)station* or a *Client*. A (Work)station can be diskless, or it can have all CD/DVD, floppy and hard drives. The boot files can be on the floppy, on the CD/DVD drive, or on the hard disk drive etc. For diskless Workstations can be on the Programmable Read Only Memory (PROM) on the network interface card. Any component of a network that uses services offered by a server is called generally *client*. The clients use operating systems for workstations. Network clients don't typically need the processing power and storage capacity required by a server computer. Network clients do need, however, to be able to properly run the client operating system they have been configured with.

Network Interface Cards (NICs). A network interface card is a card installed in the computer that enables it to communicate over a network. The network card can be installed in an extension slot, can be included in the motherboard design (as a built-in network port), it can be connected externally to an USB port or by intermediate of a PCMCIA card. Almost all NICs implement the free patent networking standard called Ethernet. Every client computer and every server computer must have at least a network interface card (or a built-in network port) in order to be able became a part of a network.

Examples network cards:

1 port PCI network card 10/100

4 ports PCI network card 10/100/1000

PCMCIA network adapter

1 port USB network adapter

Topologies and Protocol. Computers in a network are usually physically connected to each other using a communication medium (cable, radio waves, infrared waves, satellite waves etc). The devices connected into the network are referred by the term *node*. In order to allow the communication between servers and clients it is needed "something" to make a link between them. This "something" is called network environment (or medium), and is represented by topology and protocol. Topology is the cabling schematic which includes components that make up the design of the LAN. Just as people speak different languages, so do computers. The languages that allow computers to talk to another are called protocols. The protocol is the method in which the network interface cards (NICs) communicate over the topology. From the NIC point of view the protocols are essentially electronic rules of behavior that allow them to initiate and maintain communication. These rules are controlled by the protocol engine that:

- accepts raw data from the sending source;
- assembles and addresses packets;
- attaches any necessary information such as internet routing;
- places the packets onto the cable.

We can define *protocol* as "a set of rules that enables effective communications to occur" or, in other words, "an agreement between communicating parties on how communication is to proceed". The message is sent over the network, from one node to another node, as a sequence of one or many *packets*. Generally a packet includes the address of the node that sent (*source*) the packet, the address of the node the packet is being sent to (*destination*), data (chunks from the original message), and error detection and correction information. The communication take place by following a communication model generally more or less compliant with the ISO standard reference model called OSI (Open System Interconnection).

Network topologies fall more or less naturally into two categories based on what communication strategy is fallowed when transfer messages between parties:

- **Broadcast**. Broadcasting means to send a message to every other computer in the network. If a message arrives to a computer for which it was not intended, then it will be simply discarded. This strategy can be compared with the way radio programs are transmitted: anyone who has a radio can tune into stations but it is up to the receiver to decide whether to do so. This strategy is generally applied in local area network (LAN).
- **Unicast**. Unicasting means that the messages are sent specifically to a single receiver. Unicasting is typically adopted in WANs where the geographical distances prohibit the use of efficient transmission media that are used to exploit broadcast mechanisms effectively.

In order to increase the quality of offered services in a network can be included specialized servers as:

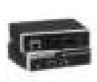
- mail servers – that allow the exchange of electronic messages;
- print servers – that allow sharing all connected printers in the network;
- modem servers – that allow to the user to share many modems at a time for calls in or out to the network;
- fax servers – that allow to users to receive and send fax in the network

environment;

- application servers – dedicated to a specific application and that allow users share the application and his data; - etc.

Repeaters. Repeaters provide the cheapest and "dumbest" interconnection between LAN's. A repeater provides a simple signal-regeneration service and operates at physical layer. As an electrical signal passes through a transmission media (for example, a length of coaxial cable), the signal degenerates in direct proportion to the distance traveled. This signal loss is called *attenuation*. A repeater links identical LANs, for instance two Ethernets, and protects against attenuation. A repeater simply amplifies the signal received on one cable segment and then retransmits (repeats) the same signal to another cable segment. The repeaters are specific to the transmission medium used. A repeater is required when the total length of a single span of network cable is larger than the maximum allowed for the cable type used. Connecting devices - such as switches, bridges, routers, and gateways - plays also a repeater role in the network.

Examples external view repeaters:



Hubs (concentrators). Network cable usually doesn't connect computers directly to each other. Instead, each computer is connected by cable to a device known as a hub (replaced now by switch). The hub, in turn, connects to the rest of the network and, possibly, to the server. Each hub contains a certain number of ports, such as 4, 8, 16, 24 etc, that allow connect as many computers as the number of ports they have. We need at least one hub/switch for our network. A hub is simply a box with circuitry inside and a bunch of jacks for RJ-45 plugs on the back. The circuitry inside the box links the cables together. A *hub* doesn't know anything about the computers that are connected to each of its ports: when a computer connected to the hub sends a packet to a computer that's connected to another port, the hub sends a duplicate copy of the packet to all its ports. Hubs are distinguished by the number of features they offer and most of those features that are designed to make the network administrator's job easier and are unnecessary in a small (say five or fewer PCs) peer-to-peer network. For such smaller systems, the minimal hub will be all you need. Hubs (and concentrators) are used to connect multiple users to a single physical device, which connects to the network. Hubs and concentrators act as repeaters by regenerating the signal as it passes through them.

The hubs operate at Layer 1 and 2 of OSI model.

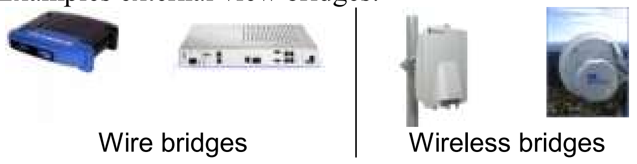
Examples external view hubs:

These images introduce a network hub external view. It is very difficult to see them at work or in a store since they replaced by switches.

This is a USB hub (not network!) and indeed you have the chance to see this at work

Bridges. Between the nodes of a network and between LAN's we can realize the communications by intermediate of bridges. Bridges provide a more intelligent connection service. A bridge can be thought of as a conscientious, but rather unimaginative, mail room clerk. From the point of view of a user, the bridges create an extended network providing access to previously unavailable devices and services. On a higher level a bridge effectively segment the network, keeping local traffic off the extended network while forwarding traffic intended for a remote device. Bridges are used to logically separate network segments within the same network. They operate at the OSI data link layer (Layer 2) and are independent of higher-layer protocols. The bridge is able to read MAC (Media Access Control) address and monitors packets as they move between segments, keeping track of the MAC addresses that are associated with various ports. As they gain more knowledge of the nodes connected to each network, they are better able to manage traffic flow.

Examples external view bridges:



Switches. Switches are similar to bridges but usually have more ports. Switches provide a unique network segment on each port, thereby separating collision domains. Today, network designers are replacing hubs in their wiring closets with switches to increase their network performance and bandwidth while protecting their existing wiring investments. A switch knows which computer is connected to each of its ports and when it receives a packet intended for a particular computer, it sends the packet only to the port that the recipient is connected to. A switch controls the flow of data by using the MAC address that is placed on each data packet (as address of source and destination). Switches divide networks into what are called *Virtual LANs* or *VLANs*. The VLAN is a logical grouping of computers on the network into a sort of communication group without requiring the computers to be in close proximity or even on the same floor. This allows to group computers that serve similar types of users into a VLAN (for example, even if your accountants are spread all over your company's office building, their computers can still be made part of the same VLAN, which would share bandwidth). Each computer on the network can be connected to its own port on the switch. By this direct connection the switch can supply each PC with a dedicated amount of bandwidth (for example, users on a 100Mbps Ethernet network can realize bandwidth of 100Mbps), so that the computers don't compete for the bandwidth. This is one reason why switches are rapidly replacing hubs. Another reason is that some switch hardware can take advantage of full-duplex access to the network media (which allows for the sending and receiving of data simultaneously on the network) and a computer on a Fast Ethernet network, which runs at 100Mbps, would actually realize a net total of 200Mbps throughput (from sending and receiving simultaneously on the full-duplex media).

Examples external view switches:

Routers. The function of a router is to direct data along the most efficient and economical route to the destination device. The routers, like bridges, can effectively extend the size of a network. The routers manage the exchange of data packets between network cabling systems, and they still to be "intelligent". From the point of view of a network user, the routers create a network of networks providing access to previously unavailable devices and services. They divide large networks into logical segments called sub-networks (subnets), division based on the addressing scheme used (such as IP, for example). Data traffic related to a particular subnet is kept local, the router only forwards data that is meant for other subnets on the extended network helping in that way to conserving network bandwidth. The Internet routers serve as intermediate "store-and-forward" devices which relay messages from source to destination. Routers decide how to forward data packets to their destinations based on a *routing table* (build, for example, by detecting the neighbor routers addresses). Routers separate broadcast domains and are used to connect different networks. Routers direct network traffic based on the destination network layer address (they operate at Layer 3 – Network of OSI model) rather than the workstation data link layer or MAC address. Routers are protocol dependent. Routers use protocols built in to their operating system to identify neighboring routers and their network addresses (such as IP addresses).

Examples routers:

1 WAN port 8 LAN ports

1 WAN port 4 LAN ports

2 WAN ports 8 LAN ports

Wireless dual

Today, in data communications, all switching and routing equipment perform two basic operations:

- *Switching data frames* - this is generally a store-and-forward operation in which a frame arrives on an input medium and is transmitted to an output medium. Switching is the process of taking an incoming frame from one interface and delivering it out through another interface;
- *Maintenance of switching operations* - in this operation, switches build and maintains switching tables and search for loops. Routers build and maintain both routing tables and service tables.

Gateways. The gateways provide the most intelligent, but slowest, connection service. A gateway is a combination of hardware and software that translates between two different protocols and acts as connection point to the network. Gateways provide translation services between different computer protocols. They allow devices on a network to communicate, and not merely connect, with devices on a different network (they acts as an entrance point to another network). They operate at Layer 4 - Transport of OSI model.

The new concept related to gateways refers to access mediation gateway, a gateway between the telephony network and other networks, such as the Internet. Access mediation supports the arbitration of call control and signaling between individual networks, resources, users, and services. Access mediation is the next evolutionary step for the advanced intelligent network (AIN). With the growing importance of the Internet, access gateways are a critical component of access mediation (International Engineering Consortium, www.iec.org).

1.3 The communication process

If two or more processes access same data, in the same time, we call that data shared data. We consider now a simple communication scheme (figure 1.2) in which a single **source process** hands over messages (e.g. notes) to other process designated as **target process**. The actual content of each message is not important and is assumed to take the form of a series of characters, suppose *n* characters. The simplest situation occurs when we are dealing with a single note. In that case, we merely need to ensure that if the target process wants to read the content of the note it will somehow be delayed until the source process has finished writing it. We can say that the processes use in common a *Shared_Note*. In order to synchronize the *Source_Process* with the *Target_Process*, that means to signal the target process the moment in which the source process writes the note (finish the writing), the message contains a field *isWritten* that acts as a semaphore: it starts with a False value (meaning the note not written yet) and ends in a True state in the moment the note is written by the sender. The target process must wait between the moment *isWritten* field switches from the False state to True state. The processes *Source_Process* and *Target_Process* communicate by using just a single note, described as a variable length note (that means the time needed to read/write operation vary). The data field for this variable length note is modified by the *Source_Process* (it modifies also the *isWritten* semaphore), and subsequently read by the *Target_Process*.

In a real communication scheme the time to wait the *Source_Process* to write the note is limited in order to eliminate the shutting off (non-blocking communication). If the *Target_Process* and *Source_Process* uses the same note repeatedly in order to exchange information the note itself should be rewritable, that means the *Target_Process* must signal to the source process the ending of read operation. We can add a new semaphore, for example called *isRead* that starts with a True value,

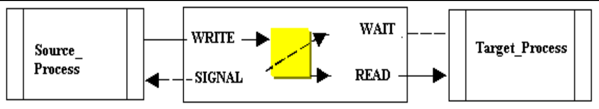


Figure 1.2 Two processes communicating via a single, shared note

change his state to False value in the moment in which the *isWritten* take the value True, and changes to True in the moment the *Target_Process* reads the note (figure 1.2). This communication schemes allow exchange no more than one piece of information at a time.

In practice a source process can pass a series of notes to target process using a queuing mechanism. A *Source_Process* writes data to a stream (the note, the synchronization information, the target address and so on) that subsequently flows to a target process where it is removed (figure 1.3).

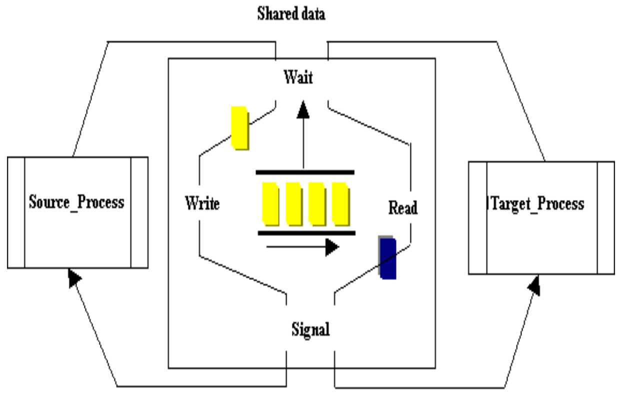


Figure 1.3 The principle of communicating by means of shared stream

The communication mechanism between many source processes and many target processes can follow the rules:

- any source process can pass a note to one target process (it will not know exactly which target process has picked up its note);
- any target process can pick up a note from the stream (it will not know exactly which source process has written the note);
- there is no restriction with respect to the number of times the stream can be accessed (it is possible that communication could not take place).

In this implementation scheme we never delay a process if communication cannot proceed because the stream has either too full to write another note or because there was simply no note to read (this is a form of non-blocking communication). A blocking mechanism can be defined by:

- delaying the source process after a certain number of notes have been written but not yet read;
- delaying the target process if there are no more written notes in a stream.

We can distinguish between two kinds of communication:

- **asynchronous** communication in which any data that is being communicating between source process and target process may be in "transit" without either of two processes waiting until that transmission is completed (it still in queue, for example). In *asynchronous communications* the transmitter and receiver use separate clocks. Although the two clocks are supposed to be running at the same speed, they don't necessarily tell the same time. An asynchronous communications system also relies on the timing of pulses to define the digital code. But they cannot look to their clocks for infallible guidance. A small error in timing can shift a bit a few numbers of positions, say from the least significant place to the most significant, which can drastically affect the meaning of the digital message;

- **synchronous** communication in which a source process and a target process has read the note written by the source process. Synchronous communications require the sending and receiving system to synchronize their actions. They share a common time base, a serial *clock*. This clock signal is passed between the two systems either as a separate signal or by using the pulses of data in the data stream to define it. The serial transmitter and receiver can unambiguously identify each bit in the data stream by its relationship to the shared clock. Because each uses exactly the same clock, they can make the match based on timing alone.

The message based communication, suppose that the two involved processes can play one of the roles: of a **sender** or of a **receiver**. From the moment that a message has been sent and until the moment it is received the message is said to be **in transmission**. Both sender and receiver can be explicitly identified through a unique address. If we send a message, to some specific address, we know exactly who's the other communicating party. This communication in which each process has precisely one address where it can be contacted is known as **point-to-point** communication. If we need communicate a message to the members of a group the point-to-point communication is very restrictive (and can be found to be impractical). In that case of groups we can have the situations:

- passing notes from several source process to a target process (e.g. all members of a group to the server of the Internet Service Provider – figure 1.4);
- passing notes from one source process to one or

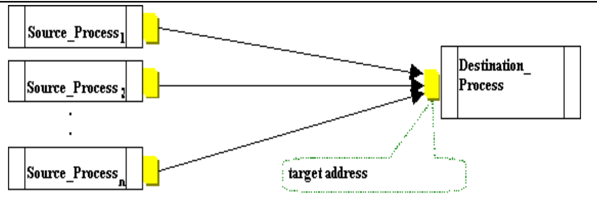


Figure 1.4 Any_to_One communication

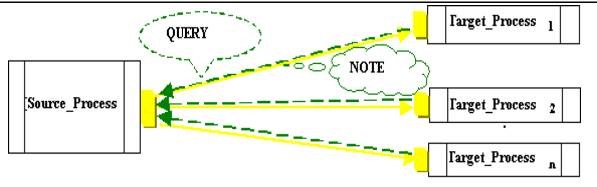


Figure 1.5 One sender to multiple receivers

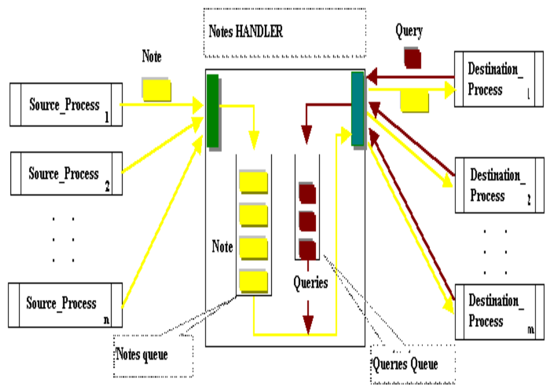


Figure 1.6 Many senders to many receivers

many target processes at the target process request (e. g. the server answers to the user request – figure 1.5);

- multiple source processes and target processes (figure 1.6).

A source process simply sends a note to the note handler which must then locate the target process to which it can forward the note. The notes are temporarily stored in an incoming queue. In order to know which target (destination) process is willing to a read note, a target process must first pass a request (Query) to the handler (the requests are also temporarily stored in its queue, Query Queue). In this scheme each sending process uses a communication buffer and has an associated **output queue** in which the messages are first appended before being transmitted to the receiver.

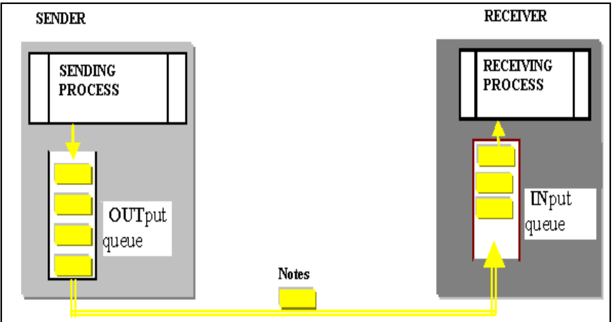


Figure 1.7 The organization of buffered message-passing

After transmission a message is appended to the receiving process **input queue**, from which it will be removed later. When a message is removed from the receiver's queue, the message is said to be **delivered** to the receiving process (figure 1.7).

1.4 Communication medium

In order to realize the communication and message exchange the computers attached to a network needs a communication medium. The communication medium is characterized by a measure called **bandwidth** that refers the transmission rate and whose unit of measurement is bits per second (bps or baud) and the highest units of this. Generally, for a communication medium, the bandwidth is defined as "the range of frequencies transmitted without being strongly attenuated". The bandwidth is a property of physical medium and usually depends on the construction, thickness, and length of the medium. The primary limit of any communications channel is its bandwidth. For an analog channel (such as a telephone line used by modems) the bandwidth specifies a range of frequencies, from the lowest to the highest, that the channel can carry or that are present in the signal. It is one way of describing the maximum amount of information that the channel can carry (the measure of analog bandwidth is in kilohertz or megahertz). In a digital circuit, the bandwidth is the amount of information that can pass through the channel (the measure of digital bandwidth in bits, kilobits, or megabits per second and so on). The kilohertz of an analog bandwidth and the kilobits per second of digital bandwidth for the same circuit are not necessarily the same and often differ greatly. The bandwidth of a communications channel defines the frequency limits of the signals that they can carry. This channel bandwidth may be physically limited by the medium used by the channel or artificially limited by communications standards. The bandwidths of radio transmissions, for example, are limited artificially, by law, to allow more different modulated carriers to share the air waves while preventing interference between them.

A network medium can use the bandwidth in two modes:

- **baseband**, when the entire bandwidth is allowed to a single signal;
- **broadband**, when the bandwidth is used to transport simultaneously two or more independent signals (similarly with the TV transmission in witch on the same cable have many programs simultaneously).

A general communication medium between a set of senders and a set of receivers is called **channel**. The medium used for transmission can be one or a combination of the following:

- copper-based medium, such as those in which a copper wire is coated with an insulating material, coaxial cables (up to 10Mbps) as used for TV sets, or so called twisted pair connection that are used in most LANs (UTP- unshielded twisted pair with a speed up to 100Mbps; STP – shielded twisted pair up to 100Mbps and allowing full-duplex communication) and 10 Gb with the new 10GbE Ethernet (the research is directed to define and realize the 100GbE over copper wires);
- fiber-optic medium, by which information is transported in the form of light, with a transmission rate > 1,000 GBps;
- satellite medium, which use radio wave to transmit data at distances over 30,000 Km;
- terrestrial microwave medium, by which radio waves are sent from one dish (antenna, aerial) to another;
- infrared waves, as those used by remote control for TV and video/audio devices.

One of the biggest problems faced by network system designers is keeping radiation and interference under control. All wires act as antenna, sending and receiving signals. As frequencies increase and wire lengths increase, the radiation increases. The pressure is on network designers to increase both the speed (with higher frequencies) and reach of networks (with longer cables) to keep up with the increasing demands of industry. Since the physical circuits are not perfect another problem of communication is error control (error-detecting and error-correcting).

Depending on how the transmission take place and how many actors are implied the channel have a specific name as:

- **mailboxes,** a channel that allow multiple senders and receivers and that are provided with message queue;
- **ports,** the channel that have only one receiver;
- **link,** is a special type of channel with only a single sender and a single receiver (the link refers to physical medium);
- A bidirectional link between a pair of processes which preserves the order of message transmission is called **connection**.

Because each individual computer can be attached at long distance to another by intermediate of a modem and this last are connected, if not internal, to a port, there data is serialized or, in other words, prepared for transmission. The transmission over a network channel uses the same principles. The basic element of digital information in a serial communication system is the data frame. A frame corresponds to a single character. Taken alone, that's not a whole lot of information. A single character rarely suffices for anything except answering multiple choice tests. To make something meaningful, you combine a sequence of characters to form words and sentences that means a data packet in terms of transmission process.
The structure of a packet and frame is dependent on the used protocol and topology. As a convention if one packet built according to the rules of a specific protocol/topology must passes over another topology the packet is first encapsulated in a packet having the heading and trailing information as required by this. Below is the structure of a TCP (Transmission Control Protocol) packet:

SP	*DP*	*SN*	*AckN*	*DO*	*Res*	*Flags*	*Window*	*Checksum*	*UP*	*Options*	*Data*

Where:

Source Port (SP) and *Destination Port (DP)* — Identifies points at which upper-layer source and destination processes receive TCP services.
Sequence Number (SN) —Usually specifies the number assigned to the first byte of data in the current message. In the connection-establishment phase, this field also can be used to identify an initial sequence number to be used in an upcoming transmission.
Acknowledgment Number (AckN) — Contains the sequence number of the next byte of data the sender of the packet expects to receive.
Data Offset (DO) — Indicates the number of 32-bit words in the TCP header.
Reserved (Res) — Remains reserved for future use.
Flag s— Carries a variety of control information, including the SYN and ACK bits used for connection establishment, and the FIN bit used for connection termination.
Window — Specifies the size of the sender's receive window (that is, the buffer space available for incoming data).
Checksum — Indicates whether the header was damaged in transit.
Urgent Pointer (UP) — Points to the first urgent data byte in the packet.
Options — Specifies various TCP options.
Data — Contains upper-layer information.

1.5 Topologies and networks

Topology is the cabling schematic which includes components that make up the design of the network. The networks, or parts of these, fits in one or more topologies from the categories described in this paragraph.
Linear (bus) topology (Ethernet, Arcnet, G-net). This topology (Figure 1.10) consists of several Workstations (nodes, sites) and a file server, which are attached to a common cable (like a TV cable).

This cable is referred to as a bus or trunk. The nodes are attached to the cable using either T-connectors or taps and drop cables. The cable ends cannot be left open, they must be terminated with a terminating resistor device matching the impedance of the cable. This terminator absorbs the signal preventing it from echoing back on the line which will cause serious signal scrambling and bring the network down.

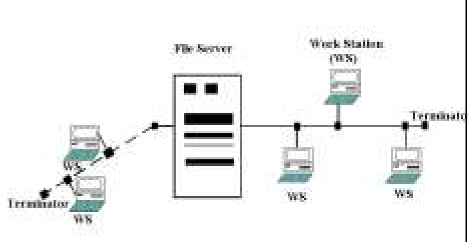


Figure 1.10 Linear Bus Topology

With a linear topology, each node is constantly monitoring (listening) to the cable. The node can only transmit data when the cable is idle, so it waits for the first opportunity. The linear topology uses CSMA/CD (Carrier Sense for Multiple Access/Carrier Detect) technology.

The terminator can be removed and the cable extended so that it is possible to attach new workstation (WS) to this line. The added line must be terminated with a terminator resistor device. It is possible also to remove WS's from this topology without eliminating the corresponding cable (Figure 1.10).

The network with linear cabling has a single backbone, one main cable that runs from one end of the system to the other. Along the way, PCs tap into this backbone to send and receive signals. The PCs link to the backbone with a single cable through which they both send and receive. In effect, the network backbone functions as a data bus, and this configuration is often called a bus topology. In the typical installation, a wire leads from the PC to the backbone, and a T-connector links the two. The network backbone has a definite beginning and end.

Distributed Star Topology (S-Net, Ethernet, Arcnet...). In the most popular network systems based on the star topology, each cable is actually twofold. Each has two distinct connections, one for sending data from the hub to an individual PC and one for the PC to send data back to the hub. These paired connections are typically packaged into a single cable (figure 1.11). Star-style networks have become popular because their topology matches that of other office wiring. In the typical office building, the most common wiring is used by telephones, and telephone wiring converges at the wiring closet in which is the PBX (Private Branch Exchange, the telephone switching equipment for a business). Star-style topologies require only a single cable and connection for each device to link to the central location where all cables converge into the network hub.

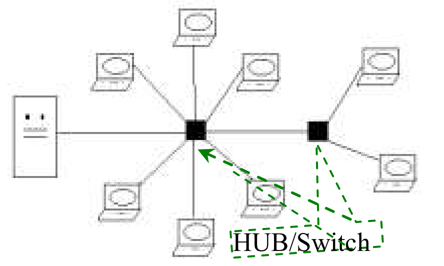


Figure 1.11. Distributed Star Topology

With this topology (figure 1.11.), the nodes are connected to a central device called a hub, multi-port repeater, wire center, and so one. Hubs are used to split or amplify network transmission signals. Star-style

topologies require only a single cable and connection for each device to link to the central location where all cables converge into the network hub. The HUBs used in networks with star topology can be of active or passive types:
- Active hubs regenerate the signals and are connected to an electrical wall outlet;
- Passive hubs have no power; they merely split the signal.

Each combination of nodes has its own hub. A new node or hub can be added to the network very easily. This makes the system extremely flexible for future growth considerations. A network with a star physical topology can be set up to function as a bus one (figure 1.12). In that scheme the hub acts as a data concentrator that mix the signals received from all PCs and resend that data to all computers connected to.

The active hubs that include a switching service, called switching hubs, can switch quickly the signals between two attached devices. Each device attached to the hub has its own private connection and can access all bandwidth of his connection and this increases the performances of the network (figure 1.13).

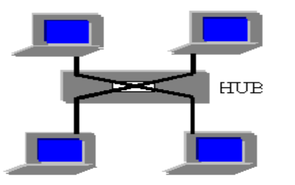


Figure 1.12 The logical transformation of a star network in a bus (functionally)

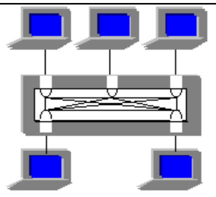


Figure 1.13 The principle of switching HUB

Logical Ring Topology. The ring topology looks like a linear network that's biting its own tail. The backbone is a continuous loop, a ring, with no end. But the ring is not a single, continuous wire. Instead it is made of short segments daisy chained from one PC to the next, the last connected, in turn, to the first. Each PC thus has two connections. One wire connects a PC to the PC before it in the ring, and a second wire leads to the next PC in the ring. Signals must traverse through one PC to get to the next, and the signals typically are listened to and analyzed along the way.

In a physical ring topology (figure 1.14.), the cable runs from node to node with the last node connecting back to the first. Here the central controlling device (file server) pools each node in a predefined sequence querying for a request for network access. If a node makes a request, the message is transmitted. If no request is made, then the server moves to the next node.

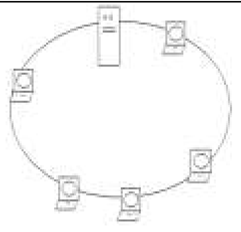


Figure 1.14 Physical Ring Topology

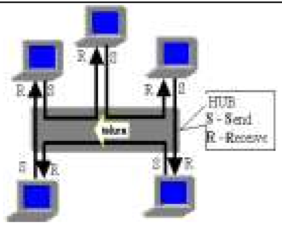


Figure 1.15 Logical Ring Topology

In a logical ring (figure 1.15) the nodes are actually cabled in a distributed star (star-wired ring) or linear (token-bus) topology. A token is passed from node to node in a logical ring. A token is a combination of special recognizable bits which grant permission to the network interface card in possession of the token to transmit data. When a token is lost or a node is added to the network, all nodes stop and count down from their respective addresses until the ring is re-established. Each one recognizes the address of the node it receives from, and the address of the node is sends to, as well its own address creating the ring.

FDDI networks. FDDI – Fiber Distributed Data Interface - works as a token ring network at 100Mbps. A FDDI network can include a primary and secondary counter-rotating rings configured to transmit data in opposite directions (figure 1.16 a). In these configurations a station can be attached to:

- A single ring – singularly attached station – SS (Single Attached Station – SAS) ;
- Both rings – dually attached station – SD (Dual Attached Station - DAS);

he connecting devices can be attached, for example:

- Single Attached Concentrator (SAC);
- Dual Attached Concentrator (DAC).

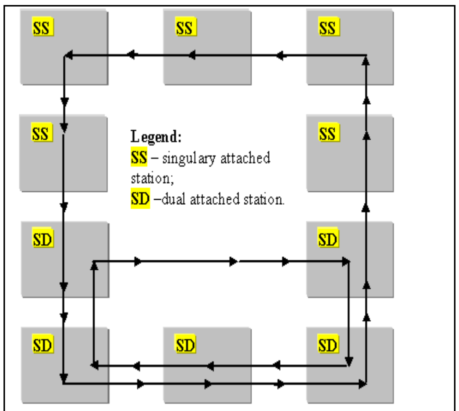


Figure 1.16 a) The principle of making rings in FDDI networks

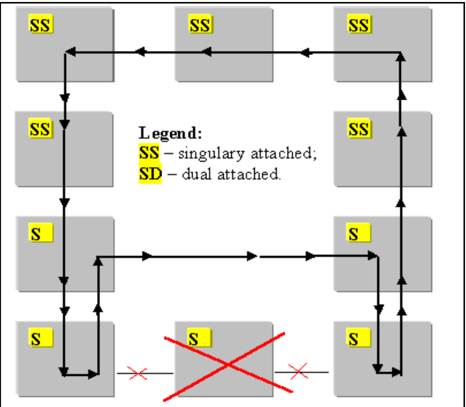


Figure 1.16 b) The principle of ring recovery in FDDI networks

Due to his configuration mode a FDDI network can be dynamically rebuild at physical failure of a cable or node (figure 1.16 b). The rings of FDDI networks can be extended to 100Km in diameter. FDDI is not so good for workstations because in that case is necessary to be reconfigured frequently. Even they offer an attractive speed because of the high costs of devices they used generally as transport networks for Internet.

Complex LANs

A saturated bus network can be extended by connecting a supplemental bus network by intermediate of a bridge as shown in figure 1.17.

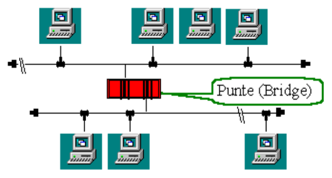


Figure 1.17 Extending bus networks

In the case in which the used links are point-to-point (P-P) the devices are added simply by adding new connections (figure 1.18).

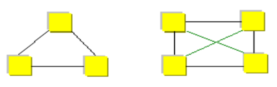


Figure 1.18 Extending P-P networks

If each device is connected to all others the obtained network is called mesh and his appearance is similarly to the example in figure 1.19. Such network has the advantage of the usage of all bandwidth and the possibility to find a route in the moment a direct connection fails.

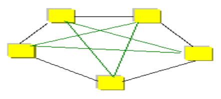


Figure 1.19 Mesh networks

The large LANs uses a version of the mesh structure called hybrid mesh his topology having a form similarly to the figure 1.20.

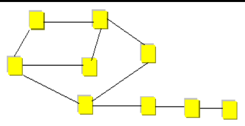


Figure 1.20 Hybrid mesh

Wide Area Network topologies. A Wide Area Network (WAN) spans a large geographical area, often a country or a continent. In the WAN the hosts are connected by intermediate of communication subnetworks (figure 1.21, 1.22). Subnets are composed by two distinct elements:

- *transmission lines*, used to move messages between machines;
- *switching elements*, represented by specialized computers and/or routers that connect three or more lines.

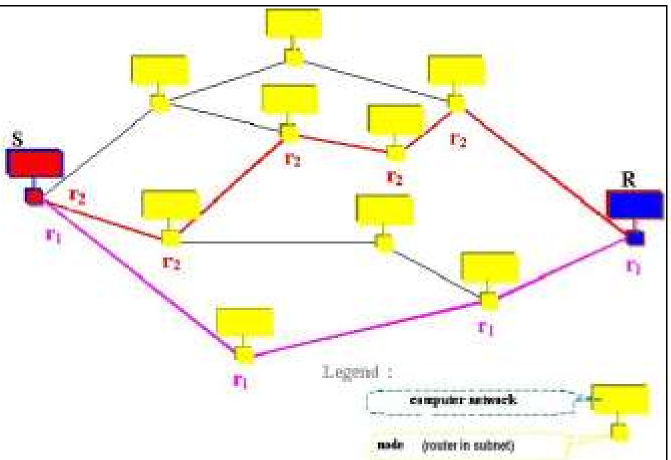


Figure 1.21 WAN topology

In most WANs, the network contains numerous transmission lines, each one connecting a pair of routers. If two routers that not connected directly whish communicate they must do this indirectly, via other routers.

WAN's are organized completely differently from local area network, in a shape of a graph (figure 1.21) in which point-to-point connections, established between routers, are now taken as they are: messages are forwarded according to some routing message and routing is based on a technology referred as switching technology (circuit-switched or packet-switched).

Circuit-switched technology is a connection-oriented service modeled after the telephone system. Packet-switching technology is a connectionless-oriented service modeled after the postal system.

The cabling schematic used for the communication subnetworks of the WAN gives his topology, a graph.

In the figure 1.21 are shown two alternative routes, labeled with r_1 and r_2, for getting message from node S (sender) to node R (receiver) in a WAN. The act of moving information across a WAN (generally an internetwork) from a source to a destination is called routing and take place at Layer 3 (network layer) of OSI reference model (see §1.10 Routers, in this chapter).

In a circuit-switched technology whenever a message is to be sent it is first necessary to set up a complete physical connection from sender S to the receiver R. An essential part of the WAN is represented by the routers, which acts as true switches in a circuit-switched network. A limitation of that kind of WAN is that as long as neither the sender nor the receiver has indicated that communication has ended, the connection should be maintained.

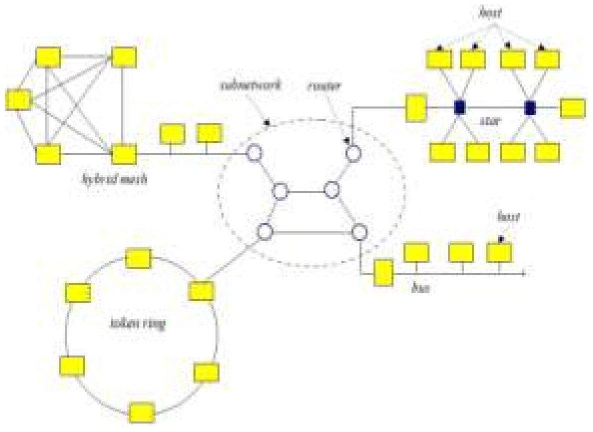


Figure 1.22 WAN - communication subnetworks, networks, and hosts

Alternatively to circuit-switched technology the packet-switched technology can be employed: a message is disassembled into a number of packets (and then decomposed into a number of frames formed by series of bits) which subsequently sent across the network (no full connection between sender and receiver is required). Each packet carries the address of the destination and nodes in the WAN will forward a packet in the right direction. When a packet is sent from one router to another via one or more intermediate routers, the packet is received at each intermediate router in its entirety, stored there until the required output line is free, and then forwarded.

In a WAN:

- **Point-to-point link** provides a single, pre-established WAN communications path from the customer premises through a carrier network, such as a telephone company, to a remote network. Point-to-point lines are usually leased from a carrier and thus are often called leased lines.
- **Switched circuits** allow data connections that can be initiated when needed and terminated when communication is complete (Integrated Services Digital Network - ISDN, for example). When a router has data for a remote site, the

switched circuit is initiated with the circuit number of the remote network. In the case of ISDN circuits, the device actually places a call to the telephone number of the remote ISDN circuit. When the two networks are connected and authenticated, they can transfer data.

- **Packet switching** is a technology in which users share common carrier resources (many customers share the carrier's network). The carrier can create virtual circuits between customers' sites by which packets of data are delivered from one to the other through the network. The section of the carrier's network that is shared is often referred to as a *cloud*. Some examples of packet-switching networks include Asynchronous Transfer Mode (ATM), Frame Relay, Switched Multimegabit Data Services (SMDS), and X.25.
- A **virtual circuit** is a logical circuit created within a shared network between two network devices. Two types of virtual circuits exist: switched virtual circuits (SVCs) and permanent virtual circuits (PVCs).
    - SVCs are virtual circuits that are dynamically established on demand and terminated when transmission is complete (are used in situations in which data transmission between devices is sporadic). Communication over an SVC consists of three phases: circuit establishment, data transfer, and circuit termination.
    - PVC is a permanently established virtual circuit that consists of one mode: data transfer. PVCs are used in situations in which data transfer between devices is constant.
- **Dialup services** offer cost-effective methods for connectivity across WANs. Two popular dialup implementations are **dial-on-demand routing** (DDR) and **dial backup**:
    - **DDR** is a technique whereby a router can dynamically initiate a call on a switched circuit when it needs to send data.
    - In **dial backup**, the switched circuit is used to provide backup service for another type of circuit, such as point-to-point or packet switching. The router is configured so that when a failure is detected on the primary circuit, the dial backup line is initiated. The dial backup line then supports the WAN connection until the primary circuit is restored. When this occurs, the dial backup connection is terminated.

WANs uses numerous types of devices that are specific to network environments such as routers, ATM switches, multiplexers, or specific to WAN environments WAN switches, access servers, modems, CSU/DSUs, and ISDN terminal adapters:

- WAN switches - a multiport internetworking device used in carrier networks such as Frame Relay, X.25, and SMDS, and operate at the data link layer of the OSI reference model;

- Access servers - a concentration point for dial-in and dial-out connections;

- Modems - a device that interprets digital and analog signals, enabling data to be transmitted over voice-grade telephone lines;

- CSU/DSUs - channel service unit/digital service unit is a digital-interface device used to connect a router to a digital circuit like a T1. The CSU/DSU also provides signal timing for communication between these devices;

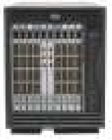
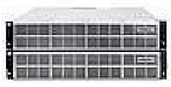
- ISDN terminal adapter - is a device used to connect ISDN Basic Rate Interface (BRI) connections to other interfaces, such as EIA/TIA-232 on a router.

Storage Area Network (SAN). A SAN is a dedicated high-speed sub-network of interconnected shared storage devices. The storage devices are available to all servers on LAN or WAN so that each server acts as pathway between the end user and the stored data.

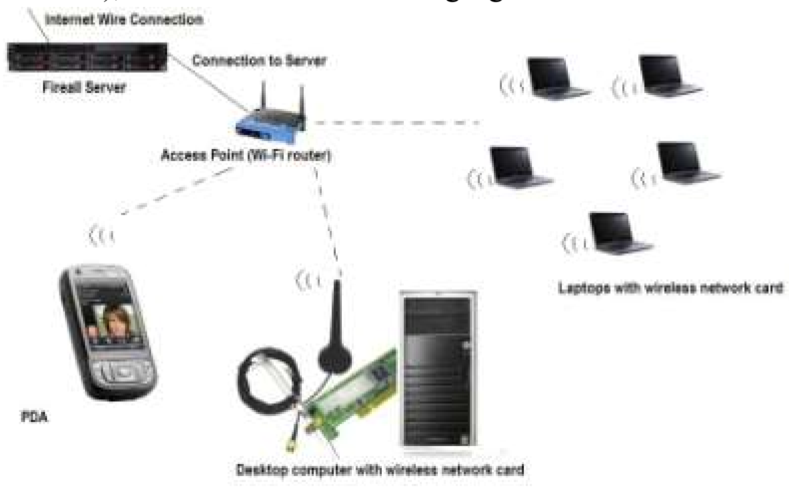
Being a network SAN has two basic components:

- SAN hardware, composed by high speed communication media (such as fiber channel, for example), storage devices, and switches
- SAN software, used to manage, monitor and configure the SAN.

Examples external view storage and connecting components for SAN:

			a) b)
IBM TotalStorage SAN256B	HP LeftHand SANs	SUN Brocade DCX-4S Backbone	a) Switch and b) disk storage

Wireless LAN (WLAN). Wireless LAN's uses short distance radio waves for communication among station devices (generally computers) provided with a wireless network card, or among station devices and access point devices (Wi-Fi routers), as shown in the following figure:


Internet Wire Connection
Fireall Server
Connection to Server
Access Point (Wi-Fi router)
Laptops with wireless network card
PDA
Desktop computer with wireless network card

The basic building block of IEEE 802.11 LAN [IEEE-802.11] is the Basic Service Set (BSS) that allow to two or more stations (STA), provided with a wireless network card, to communicate directly.

The connected stations are called members and the area where these members of WLAN remain in communication (coverage area) is called Basic Service Area (BSA). If a communicating station goes away of that area it cannot communicate directly with the other members of BSS. The IBSS (Independent Basic Service Set) is the most basic type of an IEEE 802.11 LAN. A minimal IBSS may consist of two STAs that are able to communicate directly and the network obtained is called ad hoc network (a network that is not preplanned and that exists as long as the connection maintained, as is needed. Even this type of network is dynamic it has a strong limitation due to the shortness of the distance covered. If the distance should be enlarged many BSSs can be interconnected to a Distribution System (DS). The IEEE 802.11 standard specification realizes a logical separation of the Wireless Medium (WM) from the Distribution System Medium (DSM). The access via the wireless medium for associated stations is realized by intermediate of Access Point (AP) devices (Wi-Fi routers). A wireless network of arbitrary size and complexity (such as a network of networks), an extended service set (ESS), can be created, as union of all BSSs connected to a DS. The WLAN uniquely identified by a 32 character string that is attached to the header of packets sent over the WLAN. All access points and all devices attempting to connect to a specific WLAN must use the same SSID.

1.6 Cooperative processing

Topology describes only one physical aspect of a network. The connections between the various PCs in a network also can fit one of two logical hierarchies: client-server and peer-to-peer. The alternatives form a class system among PCs. Some networks treat all PCs the same; others elevate particular computers to a special, more important role. Although the network serves the same role in either case, these two hierarchical systems enforce a few differences in how the network is used.

Client-Server. The server in a client-server network runs special software (the network operating system). The server need not be a PC. Sometimes a mainframe still slaves away at the center of a network. For PC networks, typically, the server is a special PC more powerful than the rest in the network (not with standing that the server's work is less computationally intense than that of the clients it serves). Its most important feature is storage. Because its file space is shared by many—perhaps hundreds—of PCs, it requires huge amounts of mass storage. In addition, the server is designed to be more reliable because all the PCs in the network depend on its proper functioning. If it fails, the entire network suffers. Most modern servers are designed to be fault-tolerant. That is, they will continue to run without interruption despite a fault, such as the failure of a hardware subsystem.

Most servers also use the most powerful available microprocessors, not from need but because the price difference is tiny once the additional ruggedness and storage are factored in—and because most managers think that the single most important PC in a network should be the most powerful. The corresponding term for the desktop PC workstations is client. This form of network link is, consequently, called a client-server hierarchy. Note that the special role of the server gives it more importance but also relegates it to the role of a slave that serves the need of many masters, the clients.

Peer-to-Peer. In this hierarchy every PC is equal. PCs share files and other resources (such as printers) among one another. They share equally, each as the peer of the others, so this scheme is called peer-to-peer networking. Peer-to-peer means that there is no dedicated file server as you would find in big, complex networks. All PCs can have their own, local storage, and each PC is (or can be) granted access to the disk drives and printers connected to the others. Even in peer-to-peer networks, some PCs are likely to be more powerful than others or have larger disk drives or some such distinction. Some PCs may have only floppy disks and depend on the network to supply the equivalent of hard disk storage. In other words, some PCs are created more equal than others. In fact, it's not unusual for a peer-to-peer network to have a single dominant PC that serves most of the needs of the others. In a peer-to-peer network, no one PC needs to be particularly endowed with overwhelming mass storage or an incomprehensible network operating system. Each computer connects to the network using simple driver software that makes the resources of the other PCs appear as extra disk drives and printers. The failure of a network peer only eliminates that peer; the rest of the network continues to operate. And if you duplicate vital files on at least two peers, you'll never have to fear losing data from the crash of a single system.

Peer-to-peer networking (or P2P) is the utilization of the relatively powerful computers (personal computers) that exist at the edge of the Internet for more than just client-based computing tasks **[MSTcN]**. The modern personal computer (PC) has a very fast processor, vast memory, and a large hard disk, none of which are being fully utilized when performing common computing tasks such as e-mail and Web browsing. The modern PC can easily act as both a client and server (a peer) for many types of applications. P2P networks are typically used for connecting nodes via largely *ad hoc* connections for many purposes such as file sharing (files containing audio, video, data or anything in digital format, and real-time data, such as telephony traffic).

Peer-to-peer networking has the following advantages over client/server networking:

- Content and resources can be shared from both the center and the edge of the network. In client/server networking, content and resources are typically shared from only the center of the network.
- A network of peers is easily scaled and more reliable than a single server. A single server is subject to a single point of failure or can be a bottleneck in times of high network utilization.

- A network of peers can share its processor, consolidating computing resources for distributed computing tasks, rather than relying on a single computer, such as a supercomputer.
- Shared resources of peer computers can be directly accessed. Rather than sharing a file stored on a central server, a peer can share the file directly from its local storage.

Peer-to-peer networking solves the following problems:

- Allows the processing resources of edge computers to be utilized for distributed computing tasks.
- Allows local resources to be shared directly, without the need for intermediate servers.
- Allows efficient multipoint communication without having to rely on IP multicast infrastructure.

Peer-to-peer networking enables or enhances the following scenarios:

- Real-time communications (RTC);
- Collaboration;
- Content distribution;
- Distributed processing;
- Improved Internet technologies.

Even though many organizations still process the balk of their information on large mainframe computers, the trend is migrating down to smaller, more cost-effective computers such as microcomputers and LAN servers. Cooperative processing environments provide the capability for parts of on application to execute on different computers, with data at various networked sites. The client/server architecture is a common architecture used to implement cooperative processing. More generally, trough cooperative processing architectures have two or more computers sharing application and/or file or database processing. This allows distribution of programs, files and databases across a network of interconnected computers. Cooperative processing provides transparent access to computing resources in a network so that application programs and users do not need to know where resources are located. Cooperative processing can provide a single user interface to a wide variety of remote computing resources.

This cooperative processing may be one of host-based, peer-to-peer or workstation-based processing.

1^0 Host-Based processing (or front-end). Front-end processing involves host applications sending user interface information to client workstations as if these workstations where simple host terminals.

A workstation application reads host computer transmissions by calling an application program resident on the workstation figure 1.21. Host user interface data are mapped to fields on the workstation user interface screen. Then the workstation application may proceed to edit and process the input from the user once a transaction has been completed, the workstation user interface back into transaction fields and send the data back to the host using host terminal emulation. This type of processing is more advanced than

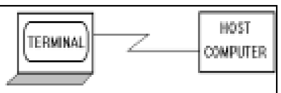


Figure 1.21 Host-based Processing

simple dumb terminal processing. No much is gained from this type of processing, since most of the work is still done on the host. Still, the workstation interface can be tailored to the user in ways that the host terminal cannot be.

2^0 Peer-to-peer processing. With this type of processing, two computers share the work load as if the two were equivalent computers (even though one could be a workstation computer and another a powerful mainframe computer). One computer typically handles the user interface processing while the other computer handles file or database processing (figure 1.22.).

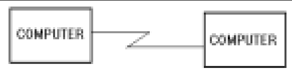


Figure 1.22 Peer-to-Peer Processing

3^0 Workstation-Based Processing. With this type of cooperative processing, a workstation program captures all user interface information, processes the information such as the translating of some or all of database or file information into a format that can be used on a host computer (the translation into a SQL query, for example). This transaction or query can then be shipped to one or most host computers using ordinary communications lines but acting as if the host computers are the local databases and file managers (figure 1.23.). The workstation environment can then act as a control center to process data from multiple host computers.

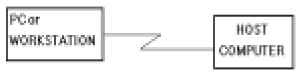


Figure 1.23 Workstation Processing

This approach loads most of the application processing on the client workstations, so it can save processing costs avoided on the host. It typically requires more powerful workstation if multiple complex applications are to operate simultaneously. If only one application is used by the workstation user at a time, then a less costly workstation may be used.

1.7 Communication models

Getting a message from here to there, especially when "here" and "there" are computers, is not a trivial task, even for designers of computer systems. In order to realize communication some level (or layers) of abstraction can be considered:

L1: Physical connection between two computers, particularly transmission medium;

L2: Transmission models (i.e. how a receiver can pick up the bits that have been put on a line by a sender);

L3: Multiplexing, which is concerned with sending several signals at the same time over a single medium in such a way each signal can be recognized separately by various receivers;

L4: Switching technology, in which we can make distinction between absence of switches (LANs) and circuit or packet-switching network (WANs);

L5: Frame transmission, particularly the means to send and receive a series of bits, and means to detect that something went wrong during the transmission;

L6: Data transmission, involving error-free transmission of large amounts of data from one computer to another by splitting the data into a series of frames.

In order to realize each level an agreement must be reached between Sender and Receiver concerning how communications is going to take place at that level. The agreements can be:

- **intra-level** agreements or horizontal that deals with agreements between Sender and Receiver with respect to a single level;
- **inter-level** agreements or vertical that describe how function at a higher level – N_k, for example – can be realized by means of the functions available at one level lower – N_{k-1}, for example. This inter-level agreements form the interface between two adjacent levels (figure 1.24).

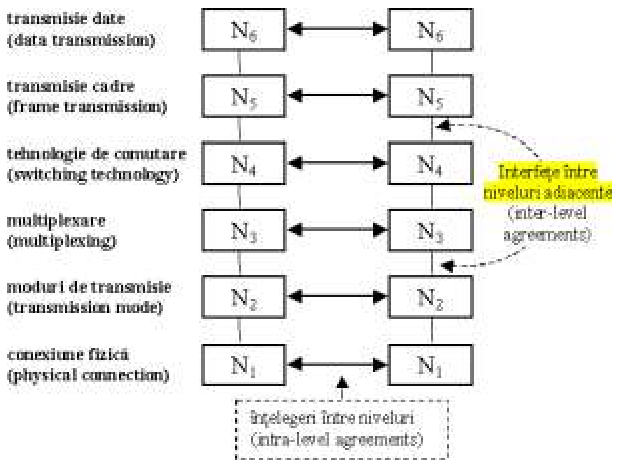


Figure 1.24 The principle of layered agreements

Many organizations are involved in setting standards for networking from which the five most important organizations are:

- **American National Standards Institute (ANSI,** pronounced *An-See***):** The official standards organization in the United States (http://www.ansi.org);
- **Institute of Electrical and Electronics Engineers (IEEE,** pronounced *Eye-triple-E***):** An international organization that publishes several key networking standards; in particular the IEEE 802.3 is the official standard for the Ethernet networking system (http://www.ieee.org);
- **International Organization for Standardization (ISO):** A federation of more than 100 standards organizations from throughout the world (http://www.iso.org).
- **Internet Engineering Task Force (IETF):** The organization responsible for the protocols that drive the Internet (http://www.ietf.org);
- **World Wide Web Consortium (W3C):** An international organization that handles the development of standards for the World Wide Web (http://www.w3c.org).

OSI MODEL

In an attempt to simplify inter-device communication, the International Organization for Standardization, commonly referred to as ISO, proposed a seven-layer model of communication. Known as the Open System Interconnection (OSI) reference model, and shown on the figure 1.25, this model has provided the impetus for practically all recent network design activities.

Each layer in the model deals with specific computer-communication functions. For our purposes, you should be familiar with at least the first four layers:

Layer 7		Application
Layer 6		Presentation
Layer 5		Session
Layer 4		Transport
Layer 3		Network
Layer 2		Data Link Control
Layer 1		PHYSICAL

Figure 1.25 The OSI reference model

1) **Physical layer** - The lowest layer of the OSI model, the *physical* layer, describes the transmission of signals across a medium (raw bits over a communication channel) that connects communicating devices. The media may be wire (as in the case of coaxial cable, twisted pair wire or fiber optic cable) or wireless (as in the case of microwave, infrared waves or satellite communications). A repeater which amplifies and repeats a signal is an example of a device that operates at the physical layer. In other words, the first layer of the OSI Reference Model defines the basic hardware of the network, which is the “cable” that conducts the flow of information between the devices linked by the network. This layer defines, for example, not only the type of wire (for example, coaxial cable or twisted pair wire) but the possible lengths and connections of the wire, the signals on the wire, and the interfaces of the cabling system. This is the level at which the device that connects a PC to the network (the network host adapter) is defined. This layer must ensure that if a 1 bit is sent by one side a 1 bit (not 0) will be received by the other side.
2) **Data Link layer** - The next OSI layer, the *data link* layer, deals with the transmission of data between devices on the same network. In addition to describing how a device accesses the transmission media, the data link layer provides some level of error detection and control. LAN technologies such as Ethernet, Token Ring, and FDDI operate at this layer. “The main task of that layer is to transform a raw transmission facility into a line that appears free of transmission errors to the network layer”. This task is accomplished at sender side by breaking the raw input data into data frames and by transmitting sequentially these frames. The data link layer also introduces addressing. Data link addresses, usually called machine or physical addresses (Media Access Control - MAC address), provide a unique identifier for each device. Bridges operate at the data link layer. It defines how information gains access to the wiring system. The Data Link layer defines the basic protocol used in the local network. This is the method used for deciding which PC can send a message over the cable at any given time, the form of the messages, and the transmission method of those messages. This level defines the structure of the data that is transferred across the network. All data transmitted under a given protocol takes a common form called the packet, or network data frame, each of which is a block of data that is strictly formatted and may include destination and source identification as well as error correction information. All network data transfers are divided

into one or more packets, the length of which is carefully controlled. Breaking network messages into multiple packets enables the network to be shared without interference and interminable waits for access. If you transferred a large file, say a bitmap, across the network in one piece, you might monopolize the entire network for the duration of the transfer. Everyone would have to wait. By breaking all transfers into manageable pieces, everyone gets access in a relatively brief period, making the network more responsive.

3) **Network layer** - The network layer, unlike the physical and data link layers, deals with the transfer of data between devices on different networks. For example, the network layer might manage the transfer of data from a user on the Engineering LAN to a user on the Manufacturing LAN through the intermediate Administration LAN. The network layer adds the concept of a network address, a specific identifier for each intermediate network between the data source and destination. Routers operate at the network layer. This layer defines how the network moves information from one device to another. This layer corresponds to the hardware interface function BIOS of an individual PC, because it provides a common software interface that hides differences in underlying hardware. Software of higher layers can run on any lower layer hardware because of the compatibility this layer affords. Protocols that enable the exchange of packets between different networks operate at this level.
4) **Transport layer** - This level is for the control of data movement across the network. It defines how messages are handled, particularly how the network reacts to packets that become lost or to other errors that may occur. The transport layer is probably of most interest to network administrators or designers in that it manages the transfer of data from a source program to a destination program. End-to-end management is facilitated by yet another type of address, the process address, which identifies a specific computer program. Gateways operate at this layer and higher OSI layers. The basic function at transport layer is to accept data from above, split it up into smaller units if needed be, pass these to the network layer, and ensure that the pieces all arrives correctly at the other end.
5) **Session layer** - This layer defines the interaction between applications and hardware much as a PC BIOS provides function calls for programs. By using functions defined at this Session layer, programmers can create software that will operate on any of a wide variety of hardware. In other words, the Session layer provides the interface for applications and the network. Among PCs, the most common of these application interfaces is IBM's Network Basic Input/Output System or NetBIOS. This layer allows users on different machines to establish sessions between them.
6) **Presentation layer** - This layer provides the file interface between network devices and the PC software. This layer defines the code and format conversions that must take place so that applications running under

a PC operating system, such as DOS, Windows, OS/2 or Linux, can understand files stored under the network's native format. This layer is concerned with the syntax and semantics of the information transmitted.

7) **Application** - The application layer includes the basic services that users expect from any network including the capability to deal with files, send messages to other network users through the mail system, and to control print jobs. The layer contains a variety of protocols that are commonly needed by users. One widely-used application protocol is HTTP, which is the basis for World Wide Web (www), http://www...

Within the OSI model, a user presents data to the application layer. Data is passed downward through the hierarchy with each layer adding addressing and/or control information (figure 1.26) in a process called encapsulating (by adding specific header and/or termination data). When data reaches the lowest layer (the physical layer), it is actually sent to a device (over the transmission channel). At the receiving end, the process is reversed. Data is passed upward through the layer hierarchy with each layer stripping address or control information in a process called de-encapsulating.

In Internet the information is no sent in one large message. For efficiency reasons the messages are broken up into separate parts called packets (usually from 1 to 1500 characters long). The transmission control protocol (TCP) performs the task of splitting up the original message into packets on dispatch and reassembling it on receipt. Figure 1.26 shows how each layer in the OSI stack add his header (similarly to a destination address) to parts of the message (datagram, frames, packets and message itself) in an encapsulating process. The datagram, frame and packet notions are used by OSI protocols (and TCP/IP also) in the message transmission/reception process:

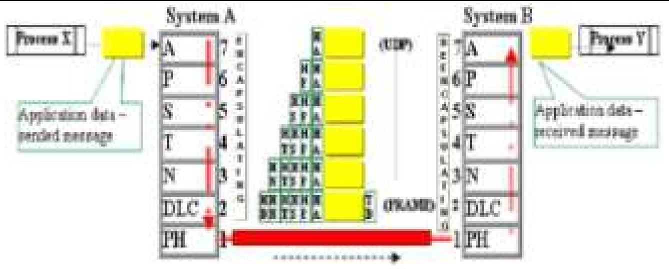

System A
System B
Process X
Process Y
Application data - sended message
Application data - received message
A
P
S
T
N
DLC
PH
7
6
5
4
3
2
1
(UDP)
(FRAME)

Figure 1.26 Data transmission in OSI model

- A **datagram** is the unit of transmission in the network layer (such as IP). A datagram may be encapsulated in one or more packets passed to the data link layer;
- A **frame** is the unit of transmission at the data link layer (DLC). A frame may include a header and/or a trailer, along with some number of units of data;
- A **packet** is the basic unit of encapsulation, which is passed across the interface between the network layer and the data link layer. A packet is usually mapped to a frame; the exceptions are when data link layer

fragmentation is being performed, or when multiple packets are incorporated into a single frame.

The Internet is a packet-switched network that uses TCP/IP as its protocol. This means, as messages and packets are sent, there is no part of the network that is dedicated to them. This is like the fact yours letters and parcels are sent by post they mixed with letters and parcels from other people. The transmission media for Internet such telephone lines, satellite links and optical cables are equivalent of the vans, trains and planes that are used to carry post.

The OSI model, particularly its conceptualization of the layered structure of the communication process, has served as an architectural blueprint for network designers. It also provides a hierarchical family or suite of protocols which designers may pick and choose from.

The earlier defined protocols were as the "languages" which enabled computers to communicate with each other. We can now refine this definition and describe a protocol as a set of rules that governs the transfer of data between identical OSI layers.

The one standout exception to the OSI design is the Internet, which was already growing, evolving, and flourishing with its own architecture. While at first the Internet firewall separated the two approaches to network design, the quick adoption of intranet concepts by businesses for internal publishing has spread the Internet philosophy inside organizations.

An intranet is a private business network generally used for the distribution of corporate information and e-mail that uses the same protocols as the Internet. For example, a business might publish its health services or benefits manual in HTML on its intranet.

IEEE MODEL

The Institute of Electrical and Electronics Engineers (IEEE) has proposed a widely used variation of the OSI model. This approach (see the figure 1.27) has provided the basis for much LAN engineering and design effort. The IEEE model segments the logic contained within the OSI data link layer.

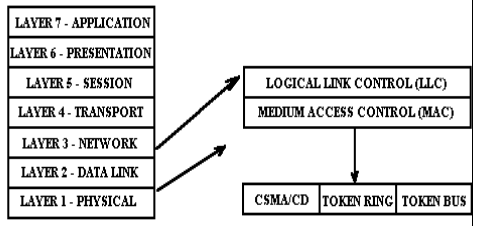


Figure 1.27 The IEEE model

1) The *medium access control (MAC)* layer provides a medium-specific access technique that describes how a device gains control of the transmission medium. Specific MAC-layer standards exist for Ethernet, Token Ring, and FDDI.
2) The *logical link control (LLC)* layer provides connection establishment, data transfer, and connection termination services. LLC provides three types of service:
   - *Unacknowledged-connectionless service* - minimizes overhead and complexity. It does not guarantee message delivery. Connectionless

service is analogous to the postal system in that no connection between message source and destination is established prior to transmission. This service is often to as "best-effort" or datagram service. It is most often used in applications where higher-layer protocols can provide functions for error detection, error recovery, and message sequencing, or in applications where the loss of scattered messages may be tolerable.

- *Connection-mode service* - provides reliable exchange of messages. It provides a connection-oriented service for ordered delivery and error detection, but at the price of some increased overhead and complexity. Connection-oriented service is analogous to the telephone system in that a connection between message source and destination is established prior to transmission. This service is best used in applications that involve lengthy data exchanges, or in applications where higher-layer protocols can be relieved of the connection management task.
- *Acknowledged-connectionless service* - is most often used in automated factory environments where a central processor may communicate with a large number of programmable devices (many of which possess limited processing capabilities). This service relieves these devices of the additional burden of connection management.

While OSI has provided a conceptual framework for network designers, its presence in terms of product designed in accordance with the conceptual model, until recently, has been minimal in the commercial marketplace. Within marketplace, other communication protocols and products (many of which use the OSI and IEEE layer structures) have achieved greater dominance.

1.8 Communications protocols

The communications protocols which have achieved greater dominance include TCP/IP, XNS, IPX, Apple Talk, DECnet, VINES, and SNA. These protocols are very briefly described on the next several pages.

TCP/IP. TCP/IP (named after two of its major components, the *Transmission Control Protocol* and the *Internet Protocol*) is the fruit of government-sponsored research faced with an exponential growth in the number of defense-related computers and faced with a need to satisfy immediate operational requirements, the Department of Defense (DoD) mandated that all its computer equipment conform to a series of military standard protocols, collectively referred to as TCP/IP. These protocols were the result of extensive research and experimentation performed by the Defense Advanced Research Projects Agency (DARPA).

Recognizing that the TCP/IP protocol suite did not conform to the OSI model, the Department of Defense committed to an eventual migration to OSI architecture. Given the current enormous installed base of TCP/IP equipment, however, such migration has proceeded at a careful pace.

TCP/IP has proven popular in the commercial marketplace as well. Vendors have introduced TCP/IP-based products and usage of such products has steadily risen.

Part of this growth can be traced to the inclusion of the TCP/IP protocols in the University of California's UNIX Berkeley Software Distribution (BSD). Many commercial users have found in TCP/IP a mature and reliable means of achieving multi-vendor connectivity. Consequently, such users have adopted TCP/IP as an interim step while awaiting the availability of OSI products. The major differences between the OSI and TCP/IP:

- The application layer in TCP/IP handles the responsibilities of layers 5, 6, and 7 in the OSI model.
- The transport layer in TCP/IP does not always guarantee reliable delivery of packets as the transport layer in the OSI model does. TCP/IP offers an option called UDP (User Datagram Protocol) that does not guarantee reliable packet delivery.

Like the OSI model, TCP/IP (whose structure is shown in figure 1.28) consists of a hierarchical series of layers. In contrast with the OSI model, TCP/IP collapses networking functionality into only four layers:

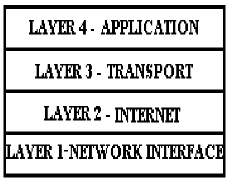


Figure 1.28 TCP/IP protocol stack

1. *Network interface layer* - managers the exchange of data between a device and the network to which it is attached. It also routes data between devices on the same network. The TCP/IP network interface layer corresponds to the OSI physical and data link layers.
   The network layer is concerned with packet routing and used low level protocols such as ICMP (Internet Control Message Protocol), IP (Internet Protocol), and IGMP (Internet Group Management Protocol). IP is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world. ICMP - provides low level support for IP, including, error messages, routing assistance, and echo requests.
2. *Internet layer* - manages the exchange of data between devices on different networks. In IEEE terms, the internet layer provides connectionless, datagram service. Like the OSI network layer to which it corresponds, the TCP/IP internet layer adds a network addressing function.
3. *Transport layer* - provides end-to-end connectivity between data source and destination. The TCP/IP transport layer corresponds to the OSI transport layer. The transport uses two protocols, UDP (User Datagram Protocol) and TCP. TCP is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received. TCP does guarantee delivery of packets to the applications which use it. UDP does not guarantee packet

delivery and applications which use this must provide their own means of verifying delivery. UDP provides a simpler protocol which as lower overhead for single messages or for software which wishes to do its own error checking.

4. *Application layer* - manages the functions (for example, remote login or file transfer) required by user programs; it corresponds to the upper layers of the OSI model. Some of the applications are SMTP (mail), Telnet, FTP, Rlogin, NFS, NIS, and LPD.

TCP/IP groups together the communication protocols used to manage the data transmission over Internet. TCP/IP is made up of two acronyms, TCP, for Transmission Control Protocol, and IP, for Internet Protocol. Very simply explaining, TCP handles packet flow between systems and IP handles the routing of packets. The functioning principle is simple: TCP divides data into packets provided with an envelope containing the required data for identification and validation; IP provide for each packet the destination address. The packets are placed onto the network and the routers define for each one the pathway to fallow. To the destination (receiver) the IP address checked together with the TCP content and if success the packets are reassembled.

The today networks are designed using a layered approach. Each layer presents a predefined interface to the layer above it. By doing so, a modular design can be developed so as to minimize problems in the development of new applications or in adding new interfaces.

In the table that follows are given some details about each layer in the TCP/IP protocol stack.

TCP/IP Protocol Stack and some software components[*)]

5	**Application**	Authentication, compression, and end user services. The application layer in TCP/IP handles the responsibilities of layers 5, 6, and 7 in the OSI model.	Application programs such as HTTP, Telnet, FTP, RLOGIN (based on TCP), SMTP (mail), NFS, DNS, NTP (based on UDP), PING and TRACEROUTE (based on ICMP and UDP). **SMTP** (Simple Mail Transfer Protocol) is an electronic mail protocol; **FTP** (File Transfer Protocol) and **TFTP** (Trivial File Transfer Protocol) allows file transfers with and, respectively without authentication; **Telnet** emulates the terminal of a TCP/IP machine; The utilities with an "**R**" prefix (REXEC, RLOGIN, RSH, RCP etc) execute remote commands.
4	**Transport**	Handles the flow of data between systems and provides access to	The transport level uses two protocols, UDP and TCP. **UDP** which stands for User

		the network for applications via the (BSD socket library). The transport layer in TCP/IP does not always guarantee reliable delivery of packets as the transport layer in the OSI model does. TCP/IP offers an option called UDP that does not guarantee reliable packet delivery.	Datagram Protocol does not guarantee packet delivery and applications which use this must provide their own means of verifying delivery. UDP - provides a simpler protocol which as lower overhead for single messages or for software which wishes to do its own error checking. **TCP** - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received. TCP does guarantee delivery of packets to the applications which use it.
3	**Network**	Packet routing	**IP** - is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world. **ICMP** - which provides low level support for IP, including, error messages, routing assistance, and echo requests.
2	**Physical**	Kernel OS/device driver interface to the network interface on the computer	The link layer is concerned with the actual transmittal of packets as well as IP to Ethernet address translation. This layer is concerned with **ARP**, the device driver, and **RARP**.

***) The number designates the corresponding starting number level in the OSI stack**

Below introduced two protocols very used in data transfers:

- **SLIP** – Serial Line Transfer Protocol - is a protocol for IP frames transport over a serial link. This protocol and, his compact version CSLIP, cannot negotiate the IP address they only transmit the information.

- **PPP** – Point-Point-Protocol – is a transfer protocol that transfers data packets from one point to another one. It transports any kind of packets (does not mother which protocol used to make the packets) by encapsulating this in PPP packet format. This protocol is used by almost dial-up connections.

The TCP/IP protocol suite includes:

- Internet Protocol (**IP**) which is the low level protocol, which transports raw data over networks;
- Internet Control Message Protocol (**ICMP**), which provides low level support for IP, including, error messages, routing assistance, and echo requests;
- Address Resolution Protocol (**ARP**), which translates logical network addresses (internet addresses) to hardware addresses (Ethernet addresses) and **RARP** (Reverse Address Resolution Protocol) is used to translate hardware addresses into internet addresses.

The **ARP** protocol operates at the link layer; it receives a destination IP address and sends out a broadcast request for all machines to see. The request asks the question "If you are the IP address N, please respond with your Ethernet address". Each host on the network monitors the network for these requests, when its address is requested it sends an ARP reply. The ARP reply specifies the 48 bit Ethernet address to use for that IP address. ARP assumes that every host knows the mapping between its own hardware address and protocol address(es). Information gathered about other hosts is accumulated in a small cache. All hosts are equal in status; there is no distinction between clients and servers.

Packet format: To communicate mappings from <protocol, address> pairs to 48bit Ethernet addresses, a packet format that embodies the Address Resolution Protocol is needed. The format of the packet has the following structure:

Ethernet transmission layer (not necessarily accessible to the user):

48.bit: Ethernet address of destination
48.bit: Ethernet address of sender
16.bit: Protocol type = ether_type$ADDRESS_RESOLUTION

Ethernet packet data:

ar$hrd	16.bit	Hardware address space (e.g., Ethernet, Packet Radio Net.)
ar$pro	16.bit	Protocol address space. For Ethernet hardware, this is from the set of type fields ether_typ$<protocol>.
ar$hln	8.bit	byte length of each hardware address
ar$pln	8.bit	byte length of each protocol address
ar$op	16.bit	opcode (ares_op$REQUEST \| ares_op$REPLY)
ar$sha	nbytes	Hardware address of sender of this packet, n from the ar$hln field
ar$spa	mbytes	Protocol address of sender of this packet, m from the ar$pln field.
ar$tha	nbytes	Hardware address of target of this packet (if known).
ar$tpa	mbytes	Protocol address of target.

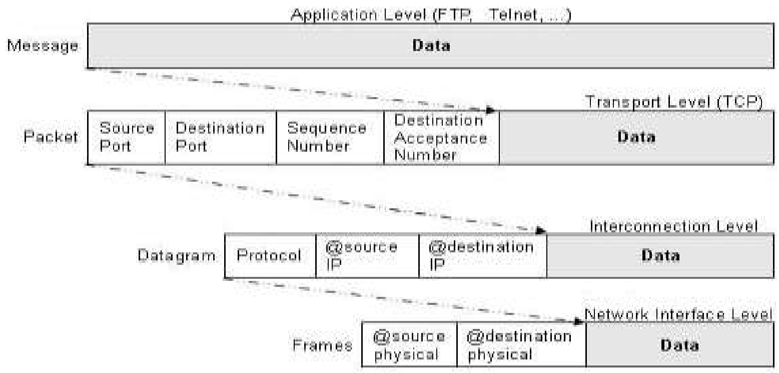
Packet Generation: As a packet is sent down through the network layers, routing determines the protocol address of the next hop for the packet and on which piece of hardware it expects to find the station with the immediate target protocol address.
Packet Reception: When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following (negative conditionals indicate an end of processing and a discarding of the packet):

```
?Do I have the hardware type in ar$hrd?
Yes: (almost definitely)
  [optionally check the hardware length ar$hln]
  ?Do I speak the protocol in ar$pro?
  Yes:
    [optionally check the protocol length ar$pln]
    Merge_flag := false
          If the pair <protocol type, sender protocol address> is already in my
          translation table, update the sender hardware address field of the entry
          with the new information in the packet and set Merge_flag to true.
    ?Am I the target protocol address?
    Yes:
          If Merge_flag is false, add the triplet <protocol type, sender protocol
          address, sender hardware address> to the translation table.
      ?Is the opcode ares_op$REQUEST?  (NOW look at the opcode!!)
      Yes:
          Swap hardware and protocol fields, putting the local hardware and
          protocol addresses in the sender fields.
          Set the ar$op field to ares_op$REPLY
          Send the packet to the (new) target hardware address on the same
          hardware on which the request was received.
```

Notice that the <protocol type, sender protocol address, sender hardware address> triplet is merged into the table before the opcode (operation code) is looked at. This is on the assumption that communication is bidirectional; if A has some reason to talk to B, then B will probably have some reason to talk to A. Notice also that if an entry already exists for the <protocol type, sender protocol address> pair, then the new hardware address supersedes the old one.

The **RARP** protocol was developed to answer the problem: Given an Ethernet address what is the IP address assigned? The RARP was developed first for diskless systems. When a diskless system boots it sends out a broadcast RARP request. The server then examines the request and determines if that host is one it is serving, if so it sends back a RARP reply containing the IP address. The host booting then uses that address to continue booting. RARP requires one or more server hosts to maintain a database of mappings from hardware address to protocol address and respond to requests from client hosts. RARP uses the same packet format used by ARP.

In Internet the data transfer unit is called datagram. Similarly to a packet the datagram composed from a heading part and a body part (data). Data can be of any size but is transported in physical frames build using encapsulation similarly to OSI model and follows a decomposition model as shown in the following image:



XNS. The five-layered Xerox Network Services Internet Transport Protocol (XNS), shown in figure 1.29, was developed by the Xerox Corporation to connect Ethernets.

1. Transmission media layer - manages the exchange of data between a device and its attached network. The XNS transmission media layer corresponds to the OSI physical and data link layers and to the TCP/IP network interface layer.
2. Internet layer - provides for the exchange of data between devices on different networks. This level defines how data is delivered across the network. The XNS internet layer corresponds to the OSI network layer and to the TCP/IP internet layer.
3. Transport layer - provides end-to-end connectivity between communicating devices. The XNS transport layer corresponds to the OSI and TCP/IP transport layers.
4. Control layer - manages data presentation and the control of device resources. The XNS control layer corresponds to the OSI session and presentation layers.
5. Application layer - manages data semantics or meaning. The XNS application layer corresponds to the OSI and TCP/IP application layers.

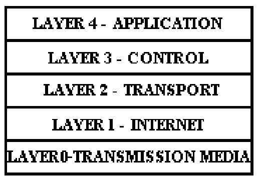


Figure 1.29 The XNS protocol

XNS spawned several proprietary derivatives. The most widespread of these derivatives is Novell's Internet Packet Exchange Protocol (IPX).

IPX

IPX, which is defined by Novell as a "service" that provides applications with the ability to send and receive messages across a network, is usually found in PC or workstation environments, supporting a wide variety of LAN topologies and media. In many ways, IPX is virtually identical to XNS - both are based on the same five-layer hierarchy. These protocol families differ primarily in IPX's provision of value-added features such as the *Service Advertising Protocol*, which enables IPX servers to broadcast their identity and offered services across a network.

The most common workstation protocols, used by Novell, are IPX (Internetwork Packet eXchange) and SPX (Sequenced Packet eXchange). The File Server protocol can use the guidelines of international standards. For example, the File Server protocol NetWare v3.11 has six layers of communication protocols between an application and the computer hardware (figure 1.30).

These protocols interacts with previous one and with the next one, except for the Application layer that interacts with the end user using an application program and except for the Hardware layer who interacts with the hardware under the specification of this last one.

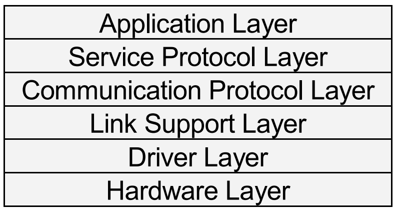


Figure 1.30 Novell NetWare v3.11 protocols

The communication protocols allow the Service Protocol Layer to communicate with the Link Support Layer.

The signal route for transmitting data between various parts of the network is called bus. Several devices can be connected to a single bus, allowing them to share the same data pathway. We use to connect the parts of a network interface cards. The network interface cards are connected using linear (bus), logical ring, and distributed star topologies. Between the sites of a network and between LAN's we realize the communications by intermediate of routers. The router manages the exchange of data packets between network cabling systems, and they still to be "intelligent". Between different types of networks the communication and adaptation of messages is ensured by bridges. The router can include the bridge functions, such that the terms can be used to denote the same functionality. Bridges can be local or remote. NetWare routers do more than transfer data packets between networks that use the same communications protocol. NetWare routers not only pass data packets, but they route the packets by way of the most efficient path. NetWare routers can also connect cabling systems that use different kinds of transmission media and different addressing systems. For example, a NetWare router can connect a network using the Ethernet addressing structure and RG/58 coaxial cable to another network using the ARCNET addressing structure and RG/62 coaxial cable.

Two main types of bridges exist in NetWare (the exemplifications and applications names are from Novell v3.11 NetWare):

- **Internal** - that runs as part of a file server. It connects separate network cabling topologies or separate networks by way of the server's NetWare operating system.

- **External** - that runs in a networked computer that is not a file server. It manages packet routing with ROUTEGEN.EXE. External routers can be dedicated or nondedicated:

  - *Dedicated* - A computer that works only as a router. It router can't function as a workstation. Since no workstation applications run on this kind of router, such applications can't hang and cause the router to stop operating. Router failure stops data sharing between networks, and also brings down workstations connected to the server via the router.

- *Nondedicated* - Can function simultaneously as a router and router as a workstation. In a nondedicated router, the workstation's NetWare shell runs "on top of" the router software.

The NetWare routers can be:

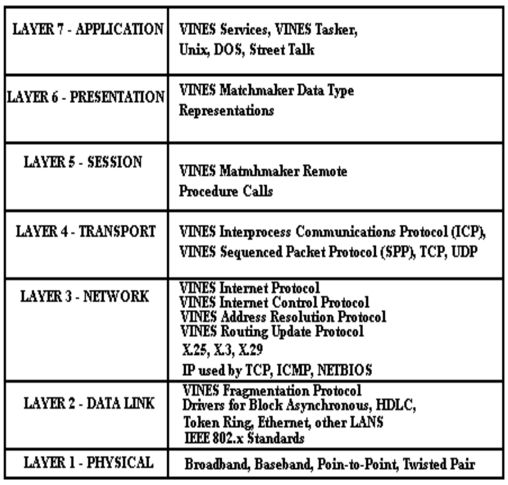
- *Local* - A bridge used within the cable-length limitations for its line drivers.
- *Remote* - A router connected beyond its driver limitations. You can find the cable-length limits for your line drivers in the NetWare installation supplements.

Not only these protocols are developed and used but they still dedicated to a computer platform, such as AppleTalk and DECNET. Other protocols such as SNA (Systems Network Architecture) are very powerful and reflects the OSI model but they are no such used in Romania. Because this protocol is OSI oriented it works as this one. The Vines (Virtual Networking System) based on UNIX System 5.3 reflects the OSI model and use IEEE standards, and industry standard protocols such as TCP/IP, AppleTalk and X.25 (figure 1.31).

1.9 Standards

A network is a collection of ideas, of hardware and software. The software comprises both the programs that make it work and the protocols that let everything work together. The hardware involves the network adapters, the wires, hubs, concentrators, routers, and even more exotic fauna. Getting it all to work together requires standardization. Because of the layered design of most networks, these standards can appear at any level in the hierarchy, and they do. Some cover a single layer and others span them all to create a cohesive system.

Ethernet. The progenitor of all of today's networks was the Ethernet system originally developed in the 1970s at the Xerox Corporation's Palo Alto Research Center for linking its Alto workstations to laser printers. The invention of Ethernet is usually credited to Robert Metcalf, who later went on to found 3Com Corporation, an early major supplier of PC networking hardware and software. During its first years, Ethernet was proprietary to Xerox, a technology without a purpose, in a world in which the PC had not yet been invented. In September 1980, however, Xerox joined with minicomputer



LAYER 7 - APPLICATION	VINES Services, VINES Tasker, Unix, DOS, Street Talk
LAYER 6 - PRESENTATION	VINES Matchmaker Data Type Representations
LAYER 5 - SESSION	VINES Matmhmaker Remote Procedure Calls
LAYER 4 - TRANSPORT	VINES Interprocess Communications Protocol (ICP), VINES Sequenced Packet Protocol (SPP), TCP, UDP
LAYER 3 - NETWORK	VINES Internet Protocol VINES Internet Control Protocol VINES Address Resolution Protocol VINES Routing Update Protocol X.25, X.3, X.29 IP used by TCP, ICMP, NETBIOS
LAYER 2 - DATA LINK	VINES Fragmentation Protocol Drivers for Block Asynchronous, HDLC, Token Ring, Ethernet, other LANS IEEE 802.x Standards
LAYER 1 - PHYSICAL	Broadband, Baseband, Poin-to-Point, Twisted Pair

Figure 1.31 Vines protocols

maker Digital Equipment Corporation (DEC) and semiconductor manufacturer Intel Corporation to publish the first Ethernet specification, which later became known as E.SPEC VER.1. The original specification was followed in November 1982 by a revision that has become today's widely used standard, E.SPEC VER.2. This specification is not what most people call Ethernet, however. In January 1985, the Institute of Electrical and Electronic Engineers (IEEE) published a networking system derived from Ethernet but not identical with it. The result was the IEEE 802.3 specification. Ethernet and IEEE 802.3 share many characteristics - physically, they use the same wiring and connection schemes - but each uses its own packet structure. Consequently, although you can plug host adapters for true Ethernet and IEEE 802.3 together in the same cabling system, the two standards will not be able to talk to one another. Some PC host adapters, however, know how to speak both languages and can exchange packets with either standard. The basis of Ethernet is a clever scheme for arbitrating access to the central bus of the system. The protocol, formally described as Carrier Sensing, Multiple Access with Collision Detection (CSMA/CD) is often described as being like a party line. It's not. It's much more like polite conversation. All the PCs in the network patiently listen to everything that's going on across the network backbone. Only when there is a pause in the conversation will a new PC begin to speak. And if two or more PCs start to talk at the same time, all become quiet. They will wait for a random interval (and because it is random, each will wait a different interval) and, after the wait, attempt to begin speaking again. One will be lucky and win access to the network. The other, unlucky PCs will hear the first PC blabbing away and wait for another pause. Access to the network line is not guaranteed in any period by the Ethernet protocol. The laws of probability guide the system, and they dictate that eventually every device that desires access will get it. Consequently, Ethernet is described as a probabilistic access system. As a practical matter, when few devices (compared to the bandwidth of the system) attempt to use the Ethernet system, delays are minimal because all of them trying to talk at one time is unlikely. As demand approaches the capacity of the system, however, the efficiency of probability-based protocol plummets. The size limit of an Ethernet system is not set by the number of PCs but by the amount of traffic; the more packets PCs send, the more contention, and the more frustrated attempts. The Ethernet protocol has many physical embodiments. These can embrace any topology, type of cable, or speed. The IEEE 802.3 specification defines several of these, and assigns a code name to each. Today's most popular Ethernet implementations operate at a raw speed of 10 MHz. That is, the clock frequency of the signals on the Ethernet (or IEEE 802.3) wire is 10 MHz. Actual throughput is lower because packets cannot occupy the full bandwidth of the Ethernet system. Moreover, every packet contains formatting and address data that steals space that could be used for data. Today's four most popular IEEE 802.3 implementations are 10Base-5, 10Base-2, 10Base-T, and 100Base-T. Although daunting at first look, you can remember the names as codes: The first number indicates the operating speed of the system in megahertz; the central word "Base" indicates that Ethernet protocol is the basis of the system;

and the final character designates the wire used for the system. The final digit (when numerical) refers to the distance in hundreds of feet the network can stretch, but, as a practical matter, also specifies the type of cable used. Coincidentally, the number also describes the diameter of the cable; under the 10 MHz 802.3 standard, the "5" stands for a thick coaxial cable that's about one-half (.5) inch in diameter; the "2" refers to a thinner coaxial cable about .2 inch in diameter; the "T" indicates twisted pair wiring like that used by telephone systems. Other differences besides cable type separate these Ethernet schemes. The 10Base-5 and 10Base-2 use a linear topology; 10Base-T and 100Base-T are built in a star configuration. The three IEEE 802.3 systems with the "10" prefix operate at the same 10 MHz speed using the same Ethernet protocol, so a single network can tie together all three technologies without the need for such complications as protocol converters (or gateways). In typical complex installations, thick coaxial cable links far-flung workgroups, each of which is tied together locally with a 10Base-T hub. This flexibility makes IEEE 802.3 today's leading networking choice. The 100Base-T system operates at 100 MHz, yielding higher performance consistent with transferring multimedia and other data intensive applications across the network. Its speed has made it the system of choice in most new installations. Actually 100Base-T isn't a single system but a family of siblings each designed for different wiring environments. 100Base-TX is the purest implementation - and the most demanding. It requires Class 5 wiring, shielded twisted pair designed for data applications. In return for the cost of the high class wiring, it permits full duplex operation so any network node can both send and receive data simultaneously. 100Base-T4 works with shielded or unshielded voice-grade wiring, Classes 3 and 4, but only allows for half-duplex operations. 100Base-FX uses the same timing and protocol as the 100Base-T systems but operates across fiber optic cables instead of copper twisted pair wiring. It also allows full duplex operation. StarLAN is the Ethernet derivative developed by AT&T and sanctioned by the IEEE as 1Base-5 in the 802.3 specification. As you would expect from a networking system designed by a telephone company, it was designed to use unshielded twisted pair wiring with a star configuration (although nodes can also be daisy chained) that can take advantage of standard office telephone wiring (where all the wires from a given office or floor converge in a wiring closet). The speed of StarLAN was set at 1 MHz to assure reliable operation over the inexpensive wiring the system used. Because 10Base-T effectively fills the same wiring niche with 10 times the speed, StarLAN has fallen out of favor.

The new standard developments are concentrated to increase speed to Gigabits and is called Gigabit Ethernet. The Gigabit speed is obtained including on copper media and the Gigabit Ethernet 10/100/1000 Mbps are in current use (fiber optic channels). The 10 Gigabit Ethernet (10GbE) is the most recent (2006) and fastest Ethernet standard, including over twisted pair copper wires 10GBase-T, with 10 gigabit/second on unshielded twisted pairs and maximum 100-150 meters per segment. The new target is 100 gigabit Ethernet (100GbE) and the IEEE teams start study and develop in November 2006.

Token Ring. Another way to handle packets across a network is a concept called token passing. In this scheme, the token is a coded electronic signal used to control network access. The token is a small frame whose structure contains three fields (one byte each): - *start delimiter*, that alerts each station the arrival of a token (or data/command frame); - *access control*, contains information about priority (3 bits), reservation (3 bits), type (1 bit) and monitor (1 bit); - *end delimiter*, signals the end of the token (or data/command frame).

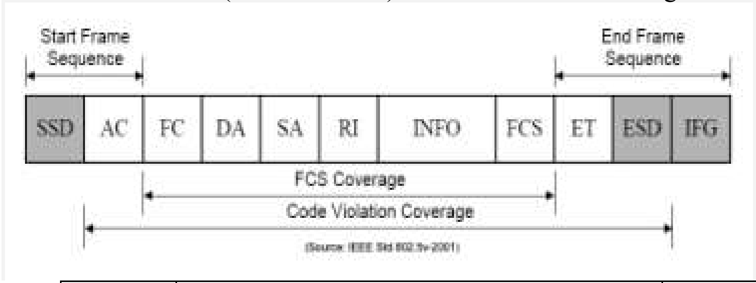
IBM originated the most popular form of this protocol, which after further development, was sanctioned by the IEEE as its 802.5 standard. Because this standard requires a ring topology, it is commonly called Token Ring networking. Although once thought the most formidable competitor to Ethernet, it is now chiefly used only in large corporations.

Other networking systems such as FDDI (see the "FDDI" section that follows) use a similar token passing protocol.

In a token passing system, all PCs remain silent until given permission to talk on the network line. They get permission by receiving the token (a small frame of information). A single token circulates around the entire network, passed from PC to PC in a closed loop that forms a ring topology. If a PC receives the token and has no packets to give to the network to deliver, it simply passes along the token to the next PC in the ring. If, however, the PC has a packet to send, it links the packet to the token along with the address of the destination PC (or server). All the PCs around the ring then pass this token and packet along until it reaches its destination. The receiving PC strips off the data and puts the token back on the network, tagged to indicate that the target PC has received its packet. The remaining PCs in the network pass the token around until it reaches the original sending PC. The originating PC removes the tag and passes the token along the network to enable another PC to send a packet. This token passing method offers two chief benefits: reliability and guaranteed access. Because the token circulates back to the sending PC, it gives a confirmation that the packet was properly received by the recipient. The protocol also assures that the PC next in line after the sending PC will always be the next one to get the token to enable communication. As the token circulates, it allows each PC to use the network so that the network operates in a deterministic way, which means that it is possible to calculate the maximum time that will pass before any end station will be capable of transmitting (not as CSMA/CD based networks that are undetermined). The token must go all the way around the ring—and give every other PC a chance to use the network—before it returns to any given PC to enable it to use the network again. Access to the network is guaranteed even when network traffic is heavy. No PC can get locked out of the network because of a run of bad luck in trying to gain access.

The original Token Ring specification called for operation at 4 MHz. A revision to the standard allows for operation at 16 MHz. The specification originally required the use of a special four-wire shield twisted pair cabling, but current standards enable several types of cabling, including unshielded twisted pair wires.

The frame structure (token and data) is similar to the following:



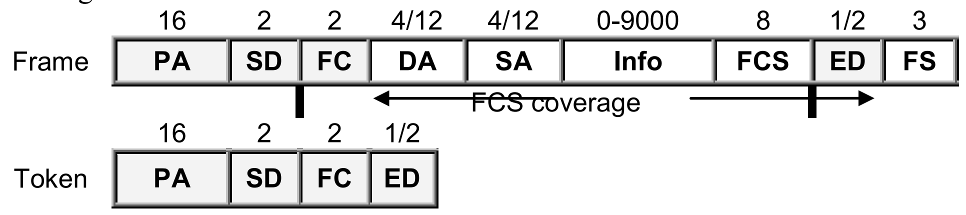
Field Abbr.	Field Name	Field Length (octets)
SSD	Start-of-Sequence Delimiter	Media dependant
AC	Access Control	1
FC	Frame Control	1
DA	Destination Address	6
SA	Source Address	6
RI	Routing Information	0 to 30
INFO	Information	0 or more
FCS	Frame Check Sequence	4
ET/ED	End Transit/Ending Delimiter	1
ESD/FS	End-of-Sequence Delimiter/Frame Status	Media dependant
IFG	Interframe Gap	Media dependant
Source: IEEE 802.5v-2001 (Amendment to IEEE Std 802.5, 1998 Edition and IEEE Stds 802.5r and 802.5j, 1998 Edition)		

Asynchronous Transfer Mode (ATM). One of the darling technologies of new networking, Asynchronous Transfer Mode or ATM is fundamentally different from other networking systems. It is a switched technology rather than a shared bus. Instead of broadcasting down a wire, a sending PC sets up a requested path to the destination specifying various attributes of the connection, including its speed. The switch need not be physical. In fact, ATM is independent of the underlying physical wiring and works with almost any physical network architecture from twisted pair to fiber optical. Its performance depends on the underlying physical implementation, but its switched design assures the full bandwidth of the medium for the duration of each connection. Instead of packets, ATM data takes the form of cells. The length of each cell is fixed at 53 bytes. The first five (header) serve as an address. The remaining 48 are the payload, the data the packet transfers. The payload can be any kind of data—database entries, audio, video, or whatever. The small, fixed-length cells are well suited to transferring voice and video traffic because such traffic is intolerant of delays that result from having to wait for a large data packet to download, among other things. ATM is independent of data types and carries any and all bytes with exactly the same dispatch. ATM is built from a layered structure. It takes the form of three layers at the bottom of the network implementation—the physical layer, the ATM layer, and the adaptation layer. The physical layer controls how ATM connects with the overall network wiring. It defines both the electrical characteristic of the connection and the actual network interface. The ATM layer takes care of addressing and routing. It adds the five-byte address header to each data cell to assure that the payload travels to the right destination. The adaptation layer takes the data supplied from higher up the network hierarchy and divides it into the 48-byte payload that will fit into each cell. ATM is part of a network. By itself it does not make a network. Because of its high speed potential and versatility, it is becoming popular in large businesses where it neatly sandwiches between other network standards.

An *ATM network* is made up of an *ATM switch* and *ATM endpoints*. An ATM switch is responsible for cell transit through an ATM network. The job of an ATM switch is well defined: It accepts the incoming cell from an ATM endpoint or another ATM switch. It then reads and updates the cell header information and quickly switches the cell to an output interface toward its destination. An ATM endpoint (or end system) contains an ATM network interface adapter. Examples of ATM endpoints are workstations, routers, digital service units (DSUs), LAN switches, and video coder-decoders (CODECs).

FDDI. Although many publications use the acronym FDDI to refer to any network using optical fibers as the transmission medium, it actually refers to an international networking standard sanctioned by the American National Standards Institute and the International Standards Organization. The initials stand for Fiber Distributed Data Interface, a logical ring standard for data transmission in LAN's that can extend in range up to 200 Km (124 miles). The standard is based on a dual counter rotating fiber optic ring topology (see paragraph 1.4 FDDI, figure 1.16 a and b), one for normal data transmission and another one for possible backup in case the primary ring fails. If the secondary ring is not used for backup purposes it can be used for data transmission too extending in that way the bandwidth to 200 MHz. The FDDI standard permits the connection of up to 1000 of PCs or other nodes with a distance up to 2 to 3 kilometers between PCs and an entire spread up to 200 kilometers. FDDI is frequently used as high-speed backbone technology.

The logical structure of the frame and token for FDDI is similar to those of token ring and contains the fields:



Field	Name	Description
PA	Preamble	16 (or more) IDLE symbols. Causes line signal changes every bit to ensure receiver clock synchronization at the beginning of a frame.
SD	Start Delimiter	the 2 symbols J and K are used to show the start of the frame and also to allow interpretation of correct symbol boundaries.

FC	Frame Control	symbols indicating whether or not this is an information frame or a MAC frame (e.g. the token), with some additional control information for the station identified by the DA.
DA	Destination address	4 or 12 symbols identifying the destination station. 16 symbols are used for a full 48-bit MAC address, 4 symbols for a 16-bit local addressing mechanism. If the first bit of the (decoded) address is a 1 then this identifies a group address.
SA	Source Address	4 or 12 symbols identifying the source station.
Info	Information	his is usually set to about 9000 symbols (4500 decoded octets) in length and is determined by the maximum length of time that a station can hold the token.
FCS	Frame Check Sequence	8 symbols containing a 32-bit CRC. The FCS covers the fields FC, DA, SA, information and FCS.
ED	End Delimiter	1 or 2 T control symbols.
FS	Frame Status	3 symbols which are a combination of R and S symbols indicating if the frame has been seen by the destination station and if it has been copied by the destination station.

Frame Relay. Is a high-performance WAN protocol that operates at the physical and data link layers of the OSI reference model. Designed initially for use across Integrated Services Digital Network (ISDN) interfaces it is used today over a variety of other network interfaces. Frame Relay is a packet-switched technology that enable end stations to dynamically share the network medium and the available bandwidth. The following two techniques are used in packet-switching technology:

- Variable-length packets - that are used for more efficient and flexible data transfers. These packets are switched between the various segments in the network until the destination is reached;
- Statistical multiplexing techniques - control network access in the packet-switched network. The advantage of this technique is that it accommodates more flexibility and more efficient use of bandwidth. Frame Relay is strictly a Layer 2 protocol suite and this enables it to offer higher performance and greater transmission efficiency making it suitable for current WAN applications, such as LAN interconnection.

Devices attached to a Frame Relay WAN fall into the following two general categories:

- Data terminal equipment (DTE) - generally are considered to be terminating equipment for a specific network and typically are located to a customer such as personal computers, routers, and bridges.
- Data circuit-terminating equipment (DCE) - are carrier-owned internetworking devices with the purpose to provide clocking and

switching services in a network, and now they are the devices that transmit data through the WAN.

Frame Relay provides connection-oriented data link layer communication. Frame Relay virtual circuit service allows creating logical connection between two data terminal equipment (DTE) devices across a Frame Relay packet-switched network (PSN). Virtual circuits provide a bidirectional communication path from one DTE device to another and are uniquely identified by a data-link connection identifier (DLCI). A number of virtual circuits can be multiplexed into a single physical circuit for transmission across the network. Frame Relay virtual circuits fall into two categories:

- **switched virtual circuits** (SVCs) - temporary connections used in situations requiring only sporadic data transfer between DTE devices across the Frame Relay network. SVCs operate in one of the four operational states: call setup, data transfer, idle, and call termination;
- **permanent virtual circuits** (PVCs) - permanently established connections that are used for frequent and consistent data transfers between DTE devices across the Frame Relay network. PVCs operates with only two states data transfer and idle.

AppleTalk. Apple Computer developed its own networking scheme for its Macintosh computers. Called AppleTalk, the network is built around an Apple-developed hardware implementation that Apple called LocalTalk. A LocalTalk can have maximum 32 active nodes and maximum 300 m a cable segment. Multiple LocalTalk can be connected together by intermediate of routers or other interconnecting devices . In operation, LocalTalk is similar to Ethernet in that it uses probabilistic access with Carrier Sensing, Multiple Access (CSMA) technology. Instead of after the fact collision detection, however, LocalTalk uses collision avoidance. Originally designed for shielded twisted pair cable, many LocalTalk networks use unshielded twisted pair telephone wiring. The LocalTalk system is slow, however, with a communication speed of 230.4 KHz (that's about one quarter megahertz).

Arcnet. Another token passing network system, Arcnet, pre-dates IEEE 802.5 Token Ring. Arcnet was developed in 1977 by Datapoint Corporation. In an Arcnet system, each PC is assigned an eight-bit address from 1 to 255. The token is passed from one PC to the next in numerical order. Each PC codes the token signal with the value of the next address in the network, the network automatically configuring itself so that only active address numbers are used. The number is broadcast on the network so that all PCs receive every token, but only the one with the right address can use it. If the PC receiving the token has a packet to send, it is then allowed to send out the packet. When the packet is received, an acknowledgment is sent back to the originating PC. The PC then passes the token to the next highest address. If the PC that receives the token has no packets to send, it simply changes the address in the token to the next higher value and broadcasts the token. Because the token is broadcast, the Arcnet system does not require a ring. Instead it uses a simple bus topology that includes star-like hubs. Arcnet hubs are either active or passive.

Active hubs amplify the Arcnet signal and act as distribution amplifiers to any number of ports (typically eight). Passive hubs act like simple signal splitters and typically connect up to four PCs. The basic Arcnet system uses coaxial cable. Compared to today's Ethernet systems, it is slow, operating at 2.5 megahertz.
Zero-Slot LANs. When you need to connect only a few PCs and you don't care about speed, you have an alternative in several proprietary systems that are lumped together as Zero-Slot LANs. These earn their name from their capability to give you a network connection without requiring you to fill an expansion slot in your PC with a network host adapter. Instead of a host adapter, most Zero-Slot LANs use a port already built into most PCs, the serial port. Protocols and topologies of Zero-Slot LANs vary with each manufacturer's implementation. Some are built as star-like systems with centralized hubs; others are connected as buses. Nearly all use twisted pair wiring, although some need only three connections and others use up to eight. The former take advantage of a protocol derived from Ethernet; the latter use the handshaking signals in the serial port for hardware arbitration of access to the network. The one factor shared by all Zero-Slot LANs is low speed. All are constrained by the maximum speed of the basic PC serial port, which is 115,200 bits per second (or about one-tenth megahertz). Lower speeds are often necessary with long reaches of cable because Zero-Slot LAN signals are particularly prone to interference. Serial ports provide only single-ended signals, which are not able to cancel induced noise and interference, as is possible with balanced signals.

1.10 Understanding Internetwork Tools

Now that you conceptually understand what internetwork devices do, we're going to explain what bridges and routers do and how they work from a technical perspective.

How bridges and routers work

With our discussion of communications models and protocols as a background, let's take a second (and more technical) look at bridges and routers to examine how they provide network connection services.

Bridges come in three basic types. Regardless of type, however, all bridges provide network connection at the data link layer, as shown in figure 1.32.

Transparent bridge

The first type of bridge, a *transparent* bridge, provides network connection to LANs that employ identical protocols at the data link and physical layers. Transparent bridges are so named because their presence and operation are transparent to network hosts. When transparent bridges are powered on, they learn the workstation locations by analyzing the source address of incoming frames from all attached networks. For example, if a bridge sees a frame arrive on port 1 from Host A, the bridge concludes that Host A can be reached through the segment connected to port 1. Transparent bridges place no burden on devices. Devices take no part in the route discovery or selection process. From the device's point of view, it appears that all devices reside on a single extended network, with each device identified by a unique address.

Using figure 1.32 as an example, transparent bridge processing can be summarized as follows:

1°. The bridge reads the data link layer destination addresses of all messages transmitted by devices on **LAN A**;

2°. The bridge ignores all messages addressed to devices on **LAN A**;

3°. The bridge accepts all messages addressed to devices on **LAN B** and, using the physical and data link protocols common to both networks, relays these messages to **LAN B**;

4°. The bridge performs identical functions for all messages transmitted on **LAN B**.

Obviously, such processing requires that the bridge acquire some knowledge of the location of devices. While this information could be manually configured, most transparent bridges provide a learning function that acquires device addresses. The bridge learns addresses by reading the data link source address of each message that it receives. As the bridge receives messages, it builds and updates a database (called the "forwarding table") that lists each data link source address, the bridge connection on which the address was seen, and a timer value that indicates the age of the observation.

The bridge relays messages on the basis of entries in its forwarding table. When the bridge reads a message, it compares the message's data link destination address with addresses found in the forwarding table. If the bridge fails to find a

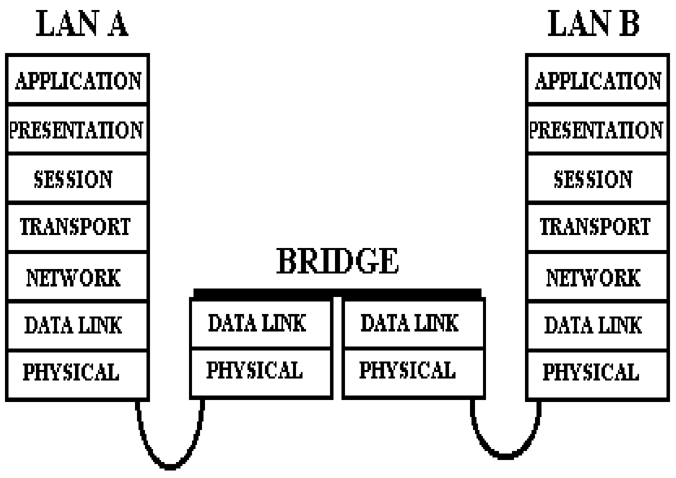


Figure 1.32 The Data-Link connection over a bridge

match, it relays the message on all bridge connections (except the connection on which the message was received). This action of relaying a message on multiple connections is called "flooding".

If the bridge finds a match between the destination address and a forwarding table entry, it compares the bridge connection on which the message was seen with the bridge connection associated with the table entry. Identical connection values indicate that the source and destination devices are located on the same network. Because relay is not necessary in this case, the bridge ignores the message.

Different connections indicate that the source and destination devices are not located on the same physical network. In this instance, the bridge forwards the message based on the connection found in the forwarding table.

Translating bridge

A *translating* bridge is a specialized from of transparent bridge. It provides network connection services to LANs that employ different protocols at the physical and data link layers. *Translational bridging* provides translation between the formats and transit principles of different media types (usually Ethernet and Token Ring). Figure 1.32 shows a translating bridge that connects adjacent Ethernet and Token Ring LANs. A translating bridge provides connection services by manipulating the "envelopes" associated with each type of LAN. Processing performed by a translating bridge is relatively straightforward, because the Ethernet, Token Ring and FDDI envelopes are somewhat similar. Each LAN type, however, sends messages of different lengths. Because a translating bridge cannot fragment messages, each LAN device must be configured to transmit messages of the supportable length.

Using figure 1.33 as an example, translating bridge processing can be summarized as follows:

1°. The translating bridge, using the physical and data link layer protocols employed by **LAN A** (the Token Ring), reads the data link layer destination addresses of all messages transmitted by devices on **LAN A**;

2°. The translating bridge ignores all messages addressed to devices on **LAN A**;

3°. The translating bridge accepts all messages addressed to devices on **LAN B** (the Ethernet) and, using the physical and data link protocols employed by **LAN B**, relays these messages to **LAN B**;

4°. The translating bridge performs identical functions for all messages transmitted on **LAN B**.

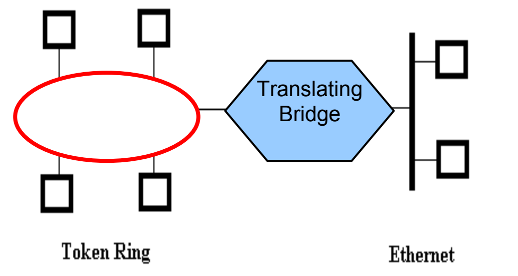


Figure 1.33 Translating bridge principle

Encapsulating bridge

An *encapsulating* bridge is generally associated with so-called "backbone" topologies. Figure 1.34 depicts such a topology with four Ethernets linked by a high-speed FDDI backbone. As shown in the figure, an encapsulating bridge provides connections to LANs that use identical physical and data link layer protocols. The internetwork connection (the backbone) is provided by a network that uses different physical and data link layer protocols.

Unlike translating bridges, which manipulate the actual message envelope,

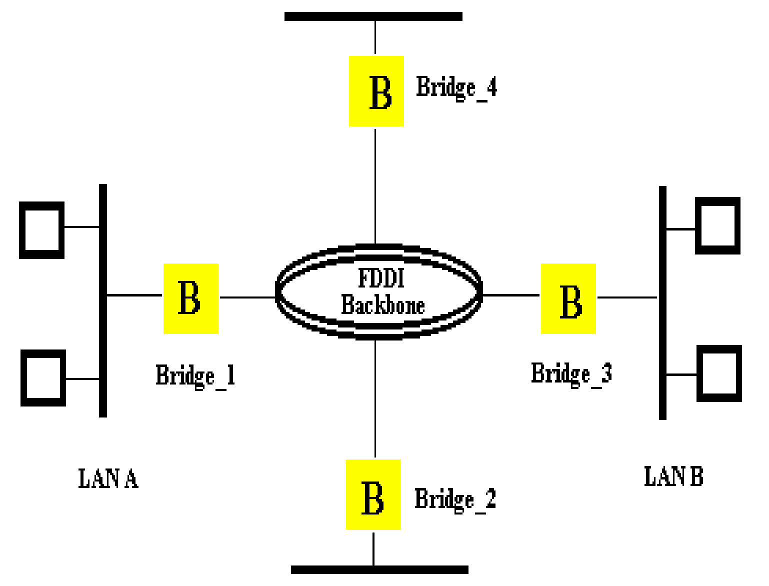


Figure 1.34 Encapsulating bridge

encapsulating bridges place received messages within a backbone-specific envelope (thus, the term "encapsulating") and forwards the encapsulated massage to other bridges for eventual delivery to the message recipient.

Using figure 1.34 as an example, a message from a device on **LAN A** to a device on **LAN B** is processed as follows:

1^{o}. **Bridge_1**, using the physical and data link layer protocols employed by **LAN A** (an Ethernet), reads the data link layer destination addresses of all messages transmitted by devices on **LAN A**;

2^{o}. **Bridge_1** ignores all messages addressed to devices on **LAN A**;

3^{o}. **Bridge_1** accepts all messages addressed to devices on other LANs, places these messages within an FDDI-specific envelope addressed to all bridges (such a collective address is called a "multicast address"), and sends this envelope across the FDDI backbone;

4^{o}. **Bridge_2** receives the message, removes the outer envelope, and checks the destination data link address. Since the address is not local, **Bridge_2** ignores the message;

5^{o}. **Bridge_3** receives the message, removes the outer envelope, and checks the destination data link address. Since the address is local, **Bridge_3** uses Ethernet physical and data link layer parameters to forward the message to the destination device;

6°. **Bridge_4** receives the message, removes the outer envelope, and checks the destination data link address. Since the address is not local, **Bridge_4** ignores the message;
7°. **Bridge_1** strips the encapsulated message from the FDDI backbone.

Source routing bridges

The term *source routing* was coined by IBM to describe a method of bridging frames across Token Ring networks. Source routing requires that the message source (not the bridge) supply the information needed to deliver a message to its intended recipient. Source routing bridges (SRBs) are so named because they assume that the complete source-to-destination route is placed in all inter-LAN frames sent by the source. SRBs store and forward the frames as indicated by the route appearing in the appropriate frame field. An SRB network contains LANs and bridges.

Within a source routing network, bridges need maintain forwarding tables. Rather, they make the decision to forward or to drop a message solely on the basis of data contained within the message envelope. To implement such a scheme, each source routing device determines the route to a destination through a process called "route discovery".

Route discovery can be accomplished in several ways. One way (somewhat simplified) goes something like this. Refer to figure 1.35, which shows a network topology within which five Token Rings are linked by three source routing bridges. To illustrate route discovery, assume that a device on **LAN_1** has a message to transmit to a device on **LAN_5**:

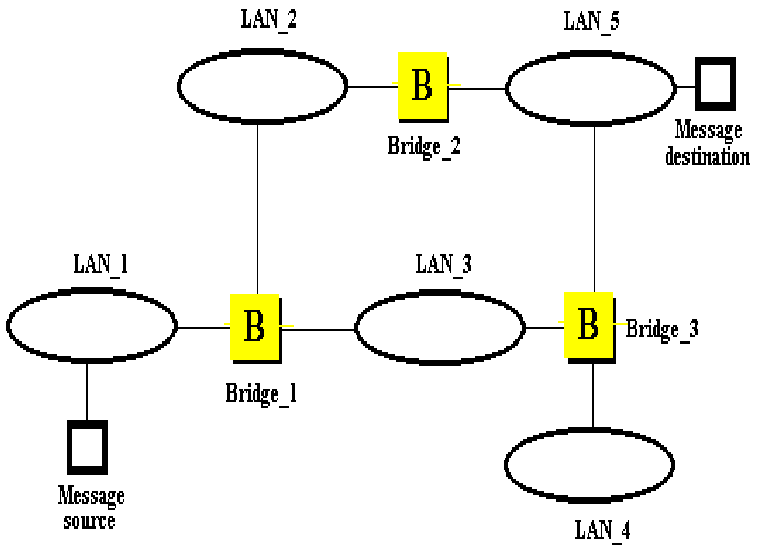


Figure 1.35 Source routing network

- The **LAN_1** device initiates route discovery by sending an "explorer" packet. Explorer packets use a unique envelope which is recognized by a source routing bridge;

- On receiving an explorer packet, each source routing bridge enters the connection on which the packet was received and its own name in a section of the envelope (called the "routing information field"). The bridge then floods the packet to all its connections except the one on which the packet was received. As a consequence, multiple copies of the same explorer message can appear on a LAN, and explorer recipient receives multiple copies of the message (one copy for each possible path from source to destination);
- Each received explorer message contains a sequenced list of connection/bridge designators, which traces the message's path through the source routed network. On receiving the explorer messages, the **LAN_5** recipient chooses one of the available routes (perhaps the fastest or the most direct) and sends a response to the **LAN_1** originator. This response lists a specific route (composed of intervening bridges and LAN connections) between source and destination;
- After "discovering" the route, the **LAN_1** device stores it in memory and uses it whenever it has messages to send to the **LAN_5** device. These messages are enclosed in a different type of envelope recognized by source routing bridges. Bridges receiving such an envelope simply scan the list of connections and bridges to obtain forwarding instructions.

Routers

Routing is the act of moving information across an internetwork from a source to a destination, and along the way, at least one intermediate node typically is encountered. Routing involves two basic activities: determining optimal routing paths and transporting information groups (typically called packets) through an internetwork.

Compared with bridges, which provide connection services at the data link layer (layer 2), routers provide connection services at the network layer (layer 3, shown in figure 1.36). This distinction provides routing and bridging with different information to use

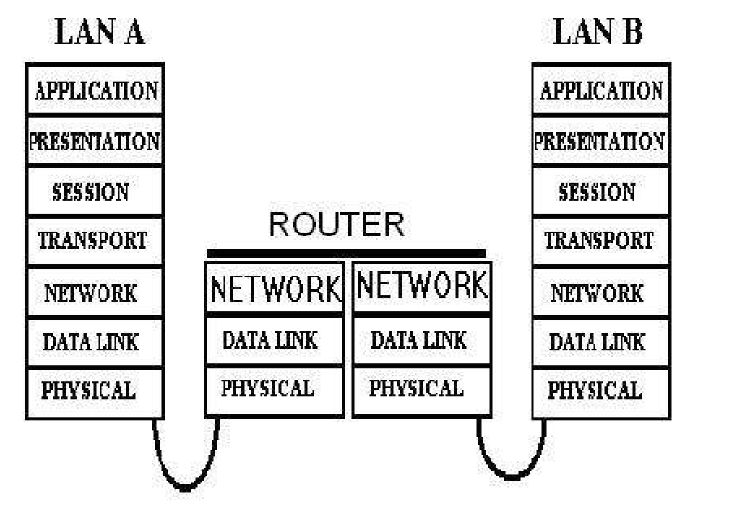


Figure 1.36 Network layer service in a router

in the process of moving information from source to destination, so the two functions accomplish their tasks in different ways. Connected networks may use different protocols at both the data link and physical layers.

In the case of two devices communicating through one a series of intervening networks, the network layer provides the information required to switch and route data to its intended destination. A router offers more sophisticated and more complex services than those offered by a bridge. It actively selects the path between source and destination nodes, basing its selection on factors such as transmission cost, transit delay, network congestion, or distance between message source and destination. Distance is usually measured in terms of "hop counts", the number of routes between a source and destination.

Unlike most bridges, whose services are transparent, a router's services must be explicitly requested by a device. A router processes only those messages that are addressed to it by other devices.

As an introduction to routing, let's build a network - one which uses no specific routing protocols, but which does demonstrate the "logic" of routing. To start, take a look at the internetwork in figure 1.1, which shows a Linear LAN and a Token Ring connected by a router. Each LAN is identified by a unique LAN address (in OSI terms, a "network layer address"), and each device on a LAN is identified by an address unique to the LAN (in OSI terms a "data link layer address").

Routing protocols use metrics to evaluate what path will be the best for a packet to travel. A metric is a standard of measurement, such as path bandwidth, that is used by routing algorithms to determine the optimal path to a destination. To aid the process of path determination, routing algorithms initialize and maintain routing tables, which contain route information. Route information varies depending on the routing algorithm used. Routing algorithms fill routing tables with a variety of information. Destination/next hop associations tell a router that a particular destination can be reached optimally by sending the packet to a particular router representing the "next hop" on the way to the final destination. When a router receives an incoming packet, it checks the destination address and attempts to associate this address with a next hop.

Routing tables also can contain other information, such as data about the desirability of a path. Routers compare metrics to determine optimal routes, and these metrics differ depending on the design of the routing algorithm used. A variety of common metrics used such as:

- path length, the most common;
- reliability, described as dependability of each network link;
- delay, length of time required to move a packet from source to destination through the internetwork;
- bandwidth, available traffic capacity to a link;
- load, the degree to which a network resource (such as a router) is busy;
- communication cost.

Routers communicate with one another and maintain their routing tables through the transmission of a variety of messages. The routing update message is one such message that generally consists of all or a portion of a routing table. By analyzing routing updates from all other routers, a router can build a detailed picture of network topology. A link-state advertisement, another example of a message sent between routers, informs other routers of the state of the sender's links. Link information also can be used to build a complete picture of network topology to enable routers to determine optimal routes to network destinations.

2 Internet – ARCHITECTURE, OFFERED SERVICES, COMMUNICATION AND NAVIGATION

2.1 How WANs (and Internet) are organized

The WANs (and the most general one, Internet) are composed from (figure 2.1):

- at the lower level LANs, MANs etc or in other words **sub-networks** ($\mathbf{S_{i,j}}$);
- at the next up the sub-networks are linked together, by using inter-network devices, in **areas** ($\mathbf{A_{k,l}}$);
- the areas are linked together, by means of routers, into **domains** ($\mathbf{D_m}$);
- all connected domains (by means of routers and using a packet or circuit switching transmission technology) form the **WAN**.

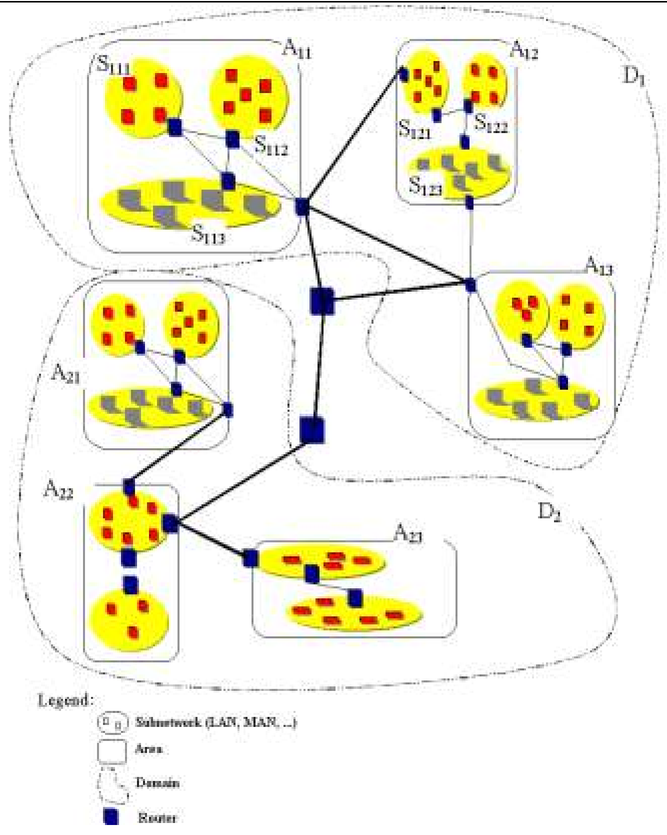


Figure 2.1 The WAN (Internet) architecture

The Internet is not a single network, but a worldwide collection of loosely connected networks that are accessible by individual computer hosts in a variety of ways, including gateways, routers, dial-up connections, and Internet Service Providers (ISP). The Internet is easily accessible to anyone with a computer and a network connection. Individuals and organizations worldwide can reach any point on the network without regard to national or geographic boundaries or time of day.

The main reason most people buy a modem—or an entire PC, for that matter—is to connect to the Internet.

The Internet is built using hardware and software. Both hardware and software serves as a means to access what you really want: the information that the Internet can bring to your personal computer. Without the right hardware and software, you could not connect to the Internet, but having the hardware alone won't get you to the World Wide Web.

The Internet has two aspects, physical and logical and it can be viewed as a collection of physical and logical pieces that are tied together physically and logically:

- The **physical aspect** is a collection of wires, optical fibers, and microwave radio links and other devices that carry digital signals between computers. The combination of connections forms a redundant network. Computers are linked to one another in a web that provides multiple signal paths between any two machines;
- The **logical aspect** is a set of standards for the signals that travel through that network. The Internet uses various protocols depending on what kind of data is being transferred. The languages that allow computers to talk to another are called protocols. The protocol is the method in which the network interface cards (NIC) communicate over the topology. Protocols are essentially electronic rules of behavior that allow the network interface cards to initiate and maintain communication. These rules are controlled by the protocol engine that:
  - accepts raw data from the sending source;
  - assembles and addresses packets;
  - attaches any necessary information such as internet routing;
  - places the packets onto the communication channel.

The Internet was not designed to link computers but to tie together computer networks and, consequently, to allow data to flow between networks. The chief protocol and the defining standard of the Internet is TCP/IP (Transmission Control Protocol/Internet Protocol). Even if you only have a single personal computer when you connect with the Internet you must run a network protocol that allows your computer logically communicate to others. The common way to make an individual personal computer a physical part of Internet is to use a modem.
Internet allows communication between millions of connected computers world-wide. Information is transmitted from client PCs (individuals or companies) whose users request services to server computers (figure 2.2) that hold information and host business applications that deliver the services in response to request.

The client PCs within homes or business are connected to Internet via local Internet Service Provider (ISP) which, in turn, are linked to larger ISPs with connection to the major national and international infrastructure or backbone (high-speed data transport channels).

The **W**orld **W**ide **W**eb (or web or www, for short) is a medium for publishing information on the Internet in an easy-to-use form. The medium is based on a standard document format known as HTML (Hypertext markup language). The www represented by all the interlinked documents on the Internet made up of pages containing text, graphics and other elements. The web is accessed using a web browser that enables user to navigate through the information available and display any page of interest

The transmission of information across the Internet is often described as being based around either pull or push technology:

- *Pull technology* describes information sent out as a result of receiving specific request, for example a page is delivered to a web browsers in response to a specific request from the user;
- *Push technology* describes information that is sent without a user's specifically requesting it, for example a customized news service received by subscribing to a channel or e-mail.

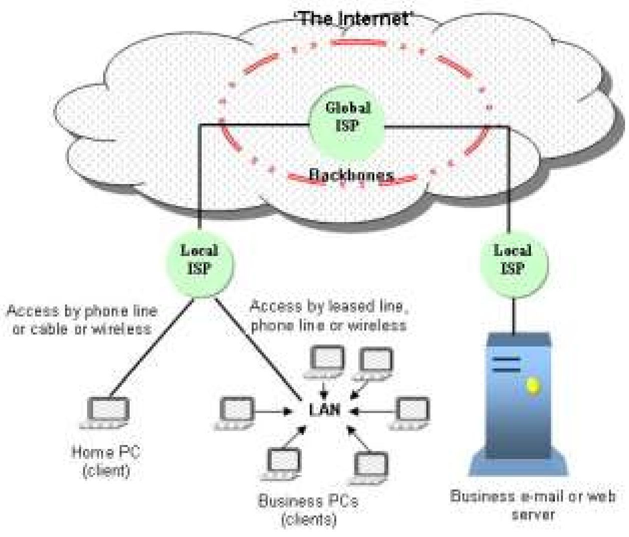

'The Internet'
Global ISP
Backbones
Local ISP
Local ISP
Access by phone line or cable or wireless
Access by leased line, phone line or wireless
LAN
Home PC (client)
Business PCs (clients)
Business e-mail or web server

Figure 2.2 Some infrastructure components of Internet

Client/Server Technology. The Internet is based on **client/server** technology (figure 2.3). All data, including e-mail messages and Web pages, are stored on server. The individuals access that resources and the net control through client applications, such as Web browser. A client uses the Internet to request information or services from a distant computer and the server sends the request information back to the client via Internet. The client platforms include a variety of devices and information appliance. An information appliance is a device (such as Internet-enabled cell phones, for example) that has been customized to perform, in a user friendly way, a few specialized computing tasks. In the following table are listed some common Internet platforms:

Device	Description
PC	General purpose computing platform that can perform many different tasks. The performed tasks can be complex to use
Net PC	Network computer with minimal local storage and processing capabilities and designed to use software and services delivered over the networks and the Internet
MID	Mobile Internet Device is a highly portable Internet-connected device both business and individual consumers designed as a pocket-size solution for access information on-the-go
Smart Phone	Provide voice communication and in addition has a small

	screen and keyboard for browsing the Web and exchanging e-mail
Game machine	Game machines provided with a modem, keyboard, and capabilities to function as Web access terminal
PDA	Wireless handheld personal digital assistant (PDA) with e-mail and Internet services. Typical functions for PDAs include address book, appointment scheduler, calculator, clock, expense tracking, currency conversions, alarm etc. Sophisticated PDA can include communications, spreadsheet and word processing applications
E-mail machine	Telnet with keyboard that provides textual e-mail capabilities (it requires linking to an e-mail service)
Set-top box	Is an important component of the Interactive digital TV system and is used to receive and decode message (from cable, satellite dish, aerial antenna etc) and then display on a conventional TV. It provide also surfing and e-mail capabilities using a television set and wireless keyboard (or remote control). The set-top box includes a modem that is used to pass back selections made on interactive channels (such as the interactive shopping channels, for example) .

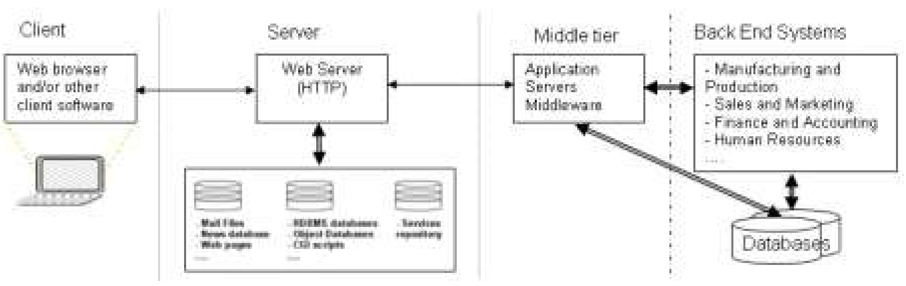


Figure 2.3 Client/server computing on the Internet

In the right side of the figure we consider the **back end systems** (or back office) that are in use by enterprises. The enterprise software consists of a set of interdependent modules for applications such as sales and distribution, financial and accounting, investment management, production planning, plant maintenance and human resources etc that allows data to be used by multiple functions and business processes for more precise coordination and control. The modules can communicate with each other directly or by sharing common repository data. Contemporary enterprise system uses client/server computing architecture.

In the companies in operation before PC and Internet appears we can found many existing legacy mainframe applications that are essential to daily operations and very risky to change. In general these ones are incompatible with the new applications developed for PC platforms. The legacy systems can be made more useful if their information and business logic can be integrated with other applications. One way to integrate various legacy systems is by using special software called middleware. **Middleware** is a special software which allows different software applications to communicate (it allows and assists data transfers between incompatible systems similarly to the way the network gateway operates in Internet).

Another way to integrate the existing systems is the use of an **enterprise application integration software (EAI)**. This kind of software is dedicated to tie together multiple applications to support enterprise integration. The software allows system builders to model their business process graphically and define the rules that applications should follow to make this process work. The software then generates the under-laying program instructions to link existing applications to each other to support those processes.

Cloud technologies. For Cloud computing we don't have yet a definition unanimously accepted. Some of these definitions are:
a) A massive network of servers or even individual PCs interconnected in a grid. The computers run in parallel, combining the resources of each to generate supercomputing-like power. [Google]
b) A cloud is a pool of virtualized computer resources that hosts a variety of different workloads and allow them to be deployed and scaled-out through the rapid provisioning of virtual machines or physical machines; supports redundant, self-recovering, highly scalable programming models and resource usage monitoring in real time to enable rebalancing of allocation when needed.
c) "Cloud computing is a style of computing where massively scalable IT-related capabilities are provided as a service across the Internet to multiple external customers [Gartner]."
d) According to [LCT] "Cloud computing is a paradigm that focuses on sharing data and computations over a scalable network of nodes." The computing cloud is a massive network of nodes having at least a two dimensional scalability:

- *horizontal* - as the ability to connect and integrate multiple clouds to work as a single logical cloud;
- *vertical* - as the ability to improve the capacity of a cloud by enhancing individual existing nodes in the cloud.

According to [IBM-09] Cloud computing is:

- a business delivery model by which hardware, software and network resources are optimally leveraged to provide innovative services over the Web, and servers are provisioned in accordance with the logical needs of the service using advanced, automated tools. The business model of a cloud facilitates more efficient use of existing resources.
- an infrastructure management methodology that enables IT organizations to manage large numbers of highly virtualized resources as a single large resource. It also allows IT organizations to massively increase their data center resources without significantly increasing the number of people traditionally required to maintain that increase.

The cloud enables the service creators, program administrators and others to use these services via a Web-based interface that abstracts away the complexity of the underlying dynamic infrastructure. The cloud also provides a user interface that allows both the user and the IT administrator to easily manage the provisioned resources through the life cycle of the service request. The cloud user disposes of self-service functions (that can be performed 24 hours a day and take only minutes to perform) to add/remove servers, change the installed software, increase/decrease the allocated processing power, memory or storage and even can start, stop and restart servers.

Figure 2.4 gives a image about how the cloud build and how user realizes the connection to cloud computing together with a closer look to the user (what is before connection) and to a layered approach of the cloud. The layers in the architecture are defined as categories of services:

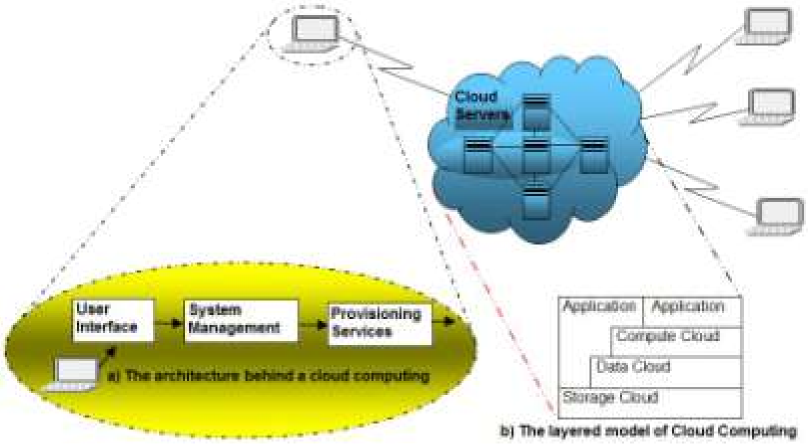


Figure 2.4 Connecting to cloud

- Storage Cloud - storage services;
- Data Cloud - data management services (record, column, or object-based);
- Compute Cloud - computational services;
- Application - generally SaaS.

The user access the cloud, for the services provisioned by the vendor from a browser application program running anywhere in the world, by intermediate of his user interface and by using the services of the system management.

The cloud computing is a logical corollary and consequence of many ancestors: grid computing, utility computing and Software-as-a-Service, as shown in figure 2.5.

Grid Computing	Utility Computing	Software-as-a-Service	Cloud Computing
- Solving large problems with parallel computing - Made mainstream by Globus Alliance (1980)	- Offering computing resources as a metered service - Introduced in late of 1990	- Network-based subscriptions to applications - Gained momentum in 2001	- Next-generation Internet computing - Next-generation data-centers
Usually a grid is a cluster of servers on which a large task could be divided into smaller tasks to run in parallel. The applications must conform to the grid software interfaces.	*On-demand computing*		*Computing and extended IT and business resources, such as servers, storage, network, applications and processes, can be dynamically shaped or carved out from the underlying hardware infrastructure and made available to a workload.*

Figure 2.5 Connecting to cloud (Adapted from IBM-09)

The cloud is easy to program than distributed or grid computing.

The clouds can be specialized such as cloud storage, cloud services, calculation cloud, etc. Related to cloud computing we have the following concepts:

- *Cloud storage* - data are stored on a virtual server having a dynamically location perceived by the user as a static one.
- *Cloud services* - any web application or service offered via “cloud computing” is called “Cloud service”. The user runs the application stored in cloud by intermediate of his web browser. If the user computer fails this fact do not affect both application and data. By storing the documents in the cloud is possible that all users granted to access and manipulate the document to work simultaneously as a team on this.
- *Software-as-a-Service* (SaaS) - a sole application is delivered to thousand of users by intermediate of vendor servers. Each organization deserved by vendor is called tenant, and the architecture of this arrangement is called

multi-tenant architecture. The clients do not pay for the software possession, as in a desktop licensed usage, instead they pay for usage based on a time scale and a subscription. The vendor servers are partitioned virtually so that each deserved organization works with an instance of the application virtually personalized (customized). The most known application offered by Cloud computing is Google MapReduce, that run on a cloud composed by 1,800 machines 2 GHz Intel Xenon, 4GB memory and 160 GB IDE disks. The estimation of Gartner is that SaaS will rise early at a rate of 22.1% until 2011.

From a Google point of view (one of the bigger supplier in Cloud resources) the Cloud computing is:

- *user-centric* - once connected a user can access the stored objects and share with others and any device accessing his data becomes as if is his object;
- *task-centric* - is focused on application result and not on the application itself;
- *powerful* - thousands of computers connected together;
- *accessible* - any computer having a connection to Internet (for efficiency considerations, a broadband connection) can use the cloud;
- *intelligent*;
- *programmable*.

The Cloud computing represents for giant IT companies a strategic field of investments in hardware, software and research:

- IBM and Dell ship cloud computing machines.
- Google have in 2008 1 million servers in 30 data-centers and realizes early investments of about $2 Billions in Datacenters.
- A new IBM-Google initiative aims to provide computer science students with a complete suite of open source-based development tools so they can gain the advanced programming skills necessary to innovate and address the challenges of this computing model which uses many computers networked together through open standards and thereby drive the Internet's next phase of growth [IBM].
- Microsoft enlarge their server farms at a rate of 20,000 new servers/month.

2.1.1 The Logical Structure of Web Servers

The base plate of a web server (figure 2.6) composed by three basic elements: the physical server, the server operating system (must include a network operating system - NOS) and the server called HTTP (HyperText Transport Protocol).

The physical structure, processor, network cards, connection and operating system are described in the books indicated by the references [AvDg03 and DgAv05] and will not be reintroduced here.

Very briefly, in the functional architecture from figure 2.6, the elements are:

- Administrative workstation (or the system console) is the workstation used to administer the operating system running on the server. In general, a Web server, that is part of Internet, is a dedicated server (it is possible to use non-dedicated servers in Intranet configuration, this means at local level and, in these situations the machine running the server operating system can be also used as a workstation and consequently can play the role of the administrative workstation).
- Servers uses a Network Operating System (NOS) that must enough capable to offer simultaneously services to most clients. The server, utilizing NOS acts the same as a network traffic police which controls the Workstation file requests (reads and writes to network drives), printer output and communications between users and file servers attached to the network. This is the system software necessary to control the access to and flow of information around the network. It is used to implement the different levels of the open system interconnection (OSI) model. It provides the following functions:
    - access control or security through providing user accounts with user names and passwords;
    - file and data sharing of data stored on a database server or file server;
    - communication between users via e-mail, diary systems or workgroup software;
    - sharing of devices.

  The Operating System of the Network can be UNIX (and anyone of his clones such as Linux), MacOS, OS/2, Novell NetWare, IBM LAN Manager (these last previous two are the most widely used), Banyan Vines, Windows NT xx Server, Windows 2000 or 2003 Server, etc;

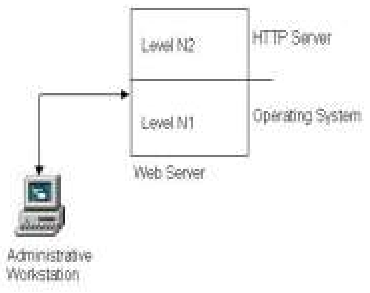


Figure 2.6 The functional architecture of the base plate of a web server

- HTTP is the protocol that governs how web browsers (clients) and web servers talk to each other. All messages sent between browsers and servers must be formatted according to the HTTP specification. The HTTP commands allow an application to interpret a page together with his HTML (HyperText Markup Language) links. The HTTP server manages, interprets and acts the HTTP commands.

    The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0, as defined by RFC1945, improved the protocol by allowing messages to be in the format of

MIME like messages, containing meta-information about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, or virtual hosts. In addition, the proliferation of incompletely implemented applications calling themselves "HTTP/1.0" has necessitated a protocol version change in order for two communicating applications to determine each other's true capabilities. This specification defines the protocol referred to as "HTTP/1.1". This protocol includes more stringent requirements than HTTP/1.0 in order to ensure reliable implementation of its features. Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation.

The system software manufacturers offer a lot of web servers that runs under different platforms. Table 2.1 shows the main pairs server-platform on the market.

Table 2.1 **The main pairs server-platform**

Manufacturer	**Server**	**Platform**
Apache	Apache*)	Linux, Unix, Windows (NT, 2k, 2003, 2008, …)
Netscape	Enterprise server	Linux, Unix, Windows (NT, 2k, 2003, …)
Microsoft	Internet Information Services	Windows (NT, 2k, 2003)
Lotus	Domino	Windows NT, OS/2
Novell	Intranetware	Netware, Windows (2k, 2003)
Sun	Sunserver	Solaris
Oracle	Webstart	Unix

*) is one of the pairs widely used in the domain of web servers

In the table below is shown the position of web server top developers:

Developer	**January 2006**	**Percent**	**February 2006**	**Percent**	**Change**
Apache	50,502,840	67.11	51,810,676	68.01	0.90
Microsoft	15,510,953	20.61	15,666,702	20.56	-0.05
Sun	1,879,856	2.50	1,880,313	2.47	-0.03
Zeus	561,524	0.75	579,198	0.76	0.01

Source: Web Server Survey news.Netcraft.com **February 2006** survey based on received responses from **76,184,000** sites

Developer	**July 2009**	**Percent**	**August 2009**	**Percent**	**Change**
Apache	113,019,868	47.17%	104,611,555	46.30%	-0.87
Microsoft	55,918,254	23.34%	49,579,507	21.94%	-1.39
qq.com	30,447,369	12.71%	30,278,988	13.40%	0.69

Google	14,226,904	5.94%	14,213,976	6.29%	0.35
nginx	10,174,573	4.25%	11,502,109	5.09%	0.84
lighttpd	1,326,240	0.55%	2,025,521	0.90%	0.34

Source: Web Server Survey news.Netcraft.com

The HTTP protocol is a request/response protocol. The HTTP protocol allow to clients and web servers to establish a connection based on TCP (Transmission Control Protocol) allowing data transfers (documents, images etc) from server to client or from client to server. A client sends a request to the server in the form of a request method, URI (Uniform Resource Identifiers), and protocol version, followed by a MIME-like (Multipurpose Internet Mail Extensions) message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta-information, and possible entity-body content. URI's have been known by many names: WWW addresses, Universal Document Identifiers, Universal Resource Identifiers, and finally the combination of Uniform Resource Locators (URL) and Names (URN). As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify - via name, location, or any other characteristic - a resource.

The data transferred have an associated data-type (a header describing the content text, image, HTML etc and how they coded) and the transfer uses the ASCII character set and the MIME standard.

The information using the MIME standard are converted as MIME standard requires and provided with a header having, for example, the following fields:

```
MIME-version: 1.0
Content-Type: type/specification parameter_name=parameter_value
Content-ID:
Content-Description:
```

In the Content-Type field is specified the type of the send message body. The specifications of document types in MIME standard are listed in the table 2.2.

Table 2.2 **The MIME standard document types**

Specification	Explanation
Text/plain	Unformatted text
Text/richtext	Text with simple formatting elements
Text/enriched	Text with complex formatting elements
Text/html	Text with HTML formatting elements
Image/jpeg	Image in JPEG format
Image/gif	Image in GIF format
Audio/basic	ISDN format on 8 bits, 1 channel and 8000 Hz
Video/mpeg	Movie in MPEG format

Message/external-body	Reference to an unformatted document stored on hard drive
Message/rfc822	Document in RFC822 (e-mail) format
Message/partial	The biggest part of a document in RFC822 (e-mail) format
Multipart/mixed	The content composed by many documents in MIME format
Multipart/alternative	The contents composed by many parts in MIME format each part containing the same information but represented in a different format
Multipart/parallel	The contents includes many parts in MIME format that can be processed simultaneously
Multipart/digest	The MIME body have many parts each of each in message/rfc822 format
Application/octet-stream	Can not be processed by the program and requires saving the MIME body in a
Application/postscript	Document or application in PostScript format
Application/x-www-form-url-encoded	Data from HTML forms

The field Content Transfer Encoding describes the method used for data coding in the MIME body as shown in table 2.3.

Table 2.3 **The types for content data coding**

Type	**Explanation**
7bit	The contents is in NVT ASCII format, un-coded
8bit	The content composed by rows containing characters represented on 8 bits, uncoded
Binary	The content composed by different characters but not divided in rows
Quoted-printable	The contents is coded in NVT ASCII format on 7 bits using the q method
Base64	The contents is coded using b method (base64)
x-user	The contents coded with a user defined method

The client application (a general browser or another web oriented application) contact the *http* server and then send his request in which it specifies the type of action the browser whishes the server to perform. The server applications, executes the client request and, send to this one an answer including the information corresponding to the query execution. The typical structure of a client query is: *method identifier*, *required object name*, *the client http protocol version number*. In the context of HTTP, a method is essentially the name of a command. The HTTP methods identifiers and the action requested are listed in table 2.4.

[Internet Society, RFC 2616] HTTP messages consist of requests from client to server and responses from server to client:

```
HTTP-message = Request | Response ; HTTP/1.1
messages
```

Request and Response messages use the generic message format of RFC 822 for transferring entities (the payload of the message). Both types of message consist of a start-line, zero or more header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and possibly a message-body.

```
generic-message = start-line
                  *(message-header CRLF)
                   CRLF
                   [ message-body ]
start-line = Request-Line | Status-Line
```

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation (Some of this are Accept, Accept-Charset, Authorization, Host, User-Agent etc; see the RFC 2616 for any details). After receiving and interpreting a request message, a server responds with an HTTP response message:

```
Response = Status-Line
*(( general-header
| response-header
| entity-header ) CRLF)
CRLF
[ message-body ]
```

The first line of a Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase
CRLF
```

The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. The first digit of the Status-Code defines the class of response:

· 1xx: Informational - Request received, continuing process
· 2xx: Success - The action was successfully received, understood, and accepted
· 3xx: Redirection - Further action must be taken in order to complete the request
· 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
· 5xx: Server Error - The server failed to fulfill an apparently valid request

Table 2.4 **Method Identifiers in HTTP protocol**

Method	**Explanation** (The hachured text in the Explanation column refers to HTTP/1.1 specification as defined in RFC 2616 Internet Society, June 1999)
GET	The client want download the content of the object specified in the query. The GET method means to retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.
HEAD	The client wants in advance the http header fields as if he receives from server when download completed. The HEAD method is identical to GET except that the server must not return a message-body in the response. The metainformation contained in the HTTP headers in response to a HEAD request should be identical to the information sent in response to a GET request. This method can be used for obtaining metainformation about the entity implied by the request without transferring the entity-body itself. This method is often used for testing hypertext links for validity, accessibility, and recent modification.
POST	The client wants modify the required object by sending in query the changed contents of the object. The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow a uniform method to cover the following functions: o Annotation of existing resources; o Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles; o Providing a block of data, such as the result of submitting a form, to a data-handling process; o Extending a database through an append operation.
PUT	The client want store transmitted data to the URL address specified in the query. The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity should be considered as a modified version of the one residing on the origin server. If the Request-URI

	does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI. If a new resource is created, the origin server must inform the user agent via the 201 (Created) response (status-code value 201). If an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes should be sent to indicate successful completion of the request. If the resource could not be created or modified with the Request-URI, an appropriate error response should be given that reflects the nature of the problem. The recipient of the entity must not ignore any Content-* (e.g. Content-Range) headers that it does not understand or implement and must return a 501 (Not Implemented) response in such cases.
PATCH	Similarly to PUT bat the body contains only the changes to be done on the object given by URL
COPY	The clients wants copy the resource specified by URL
MOVE	The client wants change the name of the resource given by URL
DELETE	The client wants delete the resource specified by URL. The DELETE method requests that the origin server delete the resource identified by the Request-URI. This method may be overridden by human intervention (or other means) on the origin server. The client cannot be guaranteed that the operation has been carried out, even if the status code returned from the origin server indicates that the action has been completed successfully. However, the server should not indicate success unless, at the time the response is given, it intends to delete the resource or move it to an inaccessible location.
LINK	The client want link the resources specified in the query
UNLINK	The client want unlink the resources specified in the query
TRACE	The client wants that the server include in the answer what this receive from his part. The TRACE method is used to invoke a remote, application-layer loop-back of the request message. The final recipient of the request should reflect the message received back to the client as the entity-body of a 200 (OK) response. The final recipient is either the origin server or the first proxy or gateway to receive a Max-Forwards value of zero (0) in the request. A TRACE request must not include an entity.
OPTIONS	The client wants supplemental information about the features offered by the specified resource. The OPTIONS method represents a request for

	information about the communication options available on the request/response chain identified by the Request-URI. This method allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval.
WRAPPED	Allows unifying sub-queries in one query.
CONNECT	This specification reserves the method name CONNECT for use with a proxy that can dynamically switch to being a tunnel (e.g. SSL tunneling).

*) The hachured text in the Explanation column refers to HTTP/1.1 specification as defined in RFC 2616 Internet Society, June 1999

The structure of the server answer is similarly with the structure of the query and contains:

- the protocol version that must be used to process the server answer (the result of the execution of the operation specified by the client);
- the header fields;
- the http body.

The best view of the Internet comes with following a packet from your personal computer:
when you log into a web site, you actually send a command to a distant server telling it to download a page of data to your personal computer (figure 2.7).

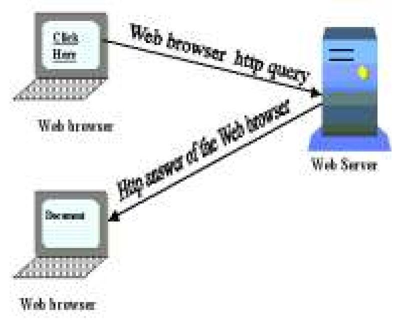


Figure 2.7 The principle of communication between a Web

Your web browser packages that command into a packet labeled with the address of the server storing the page that you want. Your personal computer sends the packet to your modem (or terminal adapter), which transmits it across your telephone or other connection to your Internet Service Provider (ISP).

The ISP actually operates as a message forwarder. At the ISP, your message gets combined with those from other PCs and sent through a higher speed connection (at least you should hope it is a high speed connection) to yet another concentrator that eventually sends your packet to one regional center. There the major Internet carriers exchange signals, routing the packets from your modem to the carrier that haul them to their destination based on their Internet address.

The World Wide Web is the most visually complicated and compelling aspect of the Internet. Despite its appearances, however, the web is nothing more than another file transfer protocol. When you call up a page from the web, the remote server simply downloads a file to your personal computer. Your web browser then decodes the page, executing commands embedded in it to alter the typeface and to display images at the appropriate place. Most browsers cache several file pages (or even megabytes of them) so that when you step back, you need not wait for the same page to download once again.

2.1.2 The “transport” protocols

TCP/IP groups together the communication protocols used to manage the data transmission over Internet. The description of TCP/IP was introduced in §1.8.

2.1.3 The IP addressing

Currently there are two types of Internet Protocol (IP) addresses in active use: IP version 4 (IPv4) and IP version 6 (IPv6). IPv4 was initially deployed on 1 January 1983 and is still the most commonly used version. IPv4 addresses are 32-bit numbers often expressed as 4 octets in "dotted decimal" notation (for example, 192.0.32.67). Deployment of the IPv6 protocol began in 1999. IPv6 addresses are 128-bit numbers and are conventionally expressed using hexadecimal strings (for example, 1080:0:0:0:8:800:200C:417A). The computers in TCP/IP based networks, even having only one computer, are called hosts. This name comes from the first deployment of TCP/IP – in the moment the standard defined the personal computers and workstation don’t exists yet – all existing computers are multi-user and for that reason they called host. In this paragraph we explain the structure and usage of IP addresses in IP version 4.

In Internet each station has a unique number expressed as a 32-bit number and all of the Internet addresses are global. From the address itself, neither you nor a computer can tell where that address is or, more importantly, how to connect to it. The routers in the Internet regional centers maintain tables to help quickly send packets to the proper address. Without such guidance, packets wander throughout the world looking for the right address.

The Internet addresses are coded on 4 bytes and are expressed in so called **dotted-decimal** notation: for example for the number 2188611658 the address can be written (in 256 base) as $130x256^3+115x256^2+144x256^1+69x256^0 \Rightarrow$ 130.115.144.69

The Internet addresses are organized in five classes from A to E. Each address belonging in the class A, B or C consists of two parts:
a) a network identifier (**netid** – network address; we denote this by letter N), distributed by the non-governmental organization InterNIC (Internet Network Information Center - www.internic.org; www.internic.net) or one of the regional centers. This address is used for routing over Internet (the choosing of the pathway from router to router);
b) a host identifier (**hostid** – the address of the machine in the network; we denote this by letter H). This address part can be also divided into two parts – sub-network address and the hardware address. The subnet address allows routing inside the private network that can provide routers or other interconnection devices that splits the network.

Both IPv4 and IPv6 addresses are assigned in a delegated manner. Users are assigned IP addresses by Internet service providers (ISPs). ISPs obtain allocations of IP addresses from a local Internet registry (LIR) or national Internet registry (NIR), or from their appropriate Regional Internet Registry (RIR):
AfriNIC (African Network Information Centre) - Africa Region
APNIC (Asia Pacific Network Information Centre) - Asia/Pacific Region
ARIN (American Registry for Internet Numbers) - North America Region
LACNIC (Regional Latin-American and Caribbean IP Address Registry) – Latin America and some Caribbean Islands
RIPE NCC (Réseaux IP Européens) - Europe, the Middle East, and Central Asia
The Internet Assigned Numbers Authority - IANA - has the role to allocate IP addresses from the pools of unallocated addresses to the RIRs according to their established needs. When an RIR requires more IP addresses for allocation or assignment within its region, the IANA makes an additional allocation to the RIR.

The class D addresses are reserved for broadcast groups (multicast addressing) and E for future use (experimental).
The range values for the first classes (A, B, and C) classes are:

Class	The maximal number of networks	The maximal number of hosts per network	Address structure*	Comments
A	128	16777216	N.H.H.H	Major networks
B	16384	65536	N.N.H.H	Large sites
C	2097152	256	N.N.N.H	Small cites, or groups of midsize

* N stand for network and H stand for host
The address range values for the five classes are:

Class	Address range		1st byte
A	0.0.0.0	121.255.255.255	1-127
B	128.0.0.0	191.255.255.255	128-191
C	192.0.0.0	223.255.255.255	192-223
D	224.0.0.0	239.255.255.255	224-239
E	240.0.0.0	241.255.255.255	240-254

IP partitions the routing problem into three parts:
- routing exchanges between end systems and routers (ARP),
- routing exchanges between routers in the same routing domain (interior routing), and,
- routing among routing domains (exterior routing).

The machines having standard IP addresses can change information over Internet. Two machines that are members of two different networks must passes through an interconnection system of the network, a router (it is possible to pass through many routers to go from one machine to another one). Each router is connected at least to two machines.

The distinction between the hardware address and network address is realized by intermediate of a so called subnet mask. The comparison between the destination address of an IP packet and the subnet-mask shows if the receiver (the destination) is a member of the same network or not. If the receiver is not a member of the same network with the sender the packet is transmitted (passed) to router that decides, according with his routing table and other reasons (the traffic, for example), to which other router will be send. If the receiver is in the same network an address resolution frame of the logic address with the hardware (physical) address is send over the network. The receiver that recognize that address resolution frame (ARP) respond by giving his hardware address and from that moment the communication between the two machines can really take place.

It is still possible for almost people to get assignment of a number for a small "Class C" network in which the first three bytes identify the network and the last byte identifies the individual computer. Larger organizations can get a "Class B" network where the first two bytes identify the network and the last two bytes identify each of up to 64 thousand individual workstations. There are only about 2 million class A, B and C addresses. Almost all the "B" class addresses are assigned. As a result there is a proposal to enlarge the address space to 128 bits, called IPNG (Internet Protocol Next Generation or IPv6). It also removes certain non-essential features of the IP protocol making it faster and easier to implement.

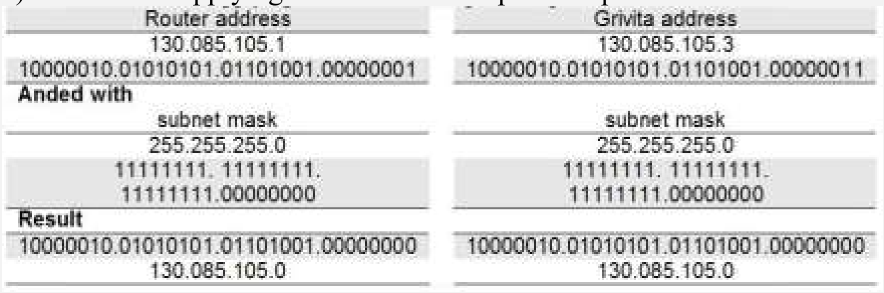
Certain addresses have special meanings. In particular 0, 127 and 255 are usually reserved for special use. The number 255 indicates a broadcast address (for example 131.123.2.255), which is listened for by all machines on the net or subnet. Note that some vendors use 0 as the broadcast address by default (e.g. Sun) whereas others use 255. All systems on a network must be configured to use the same broadcast address. This is set with the operating system specific command. The value 0 is not assigned to any machine or network. The network address with 127 as the first byte is the "loopback network", which is fictitious. The address 127.0.0.1 is called "localhost" and means the current host machine.

The organization connects to the Internet through one of a dozen regional or specialized network suppliers. The network vendor is given the subscriber network number and adds it to the routing configuration in its own machines and those of the other major network suppliers.

When designing networks we generally build a network of networks using some devices, such as routers and bridges, allowing us to extend a network beyond the limits imposed by the standard on a single network. It turns out that routers or occasionally hosts acting as gateways play a special part in this design. Routers and gateways understand different protocols, such as IP, and can look at the IP portion of a packet and from the destination address to determine the route it should take next. The IP uses a concept called subnets to determine individual networks. Each separate network is a separate subnet, the router needs to look at IP addresses and determine if they belong to that network or not. This is down by using a subnet mask. The subnet mask is a 32 bit value as IP address is and is logically and-ed with the IP address to see if the destination is on the same network as the router or gateway. For a class B address a normal subnet mask value would be 255.255.255.0 or sometimes displayed with the hexadecimal address of 0xffffff00. Those values produce the equivalent mask, one with the first 24 bits set to 1 and then the remaining bits are 0.

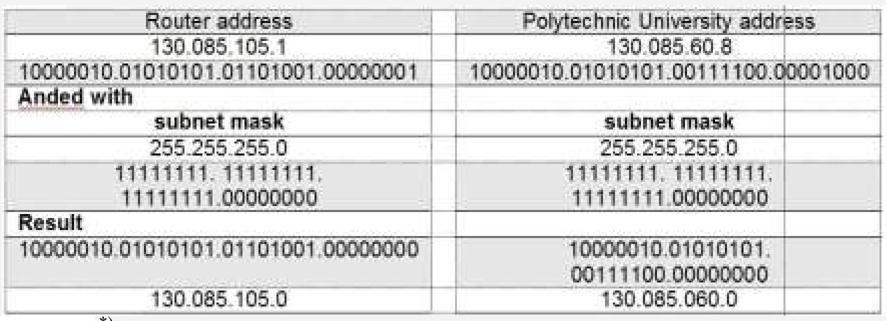
Lets use an example to demonstrate how this works. Grivita building has an IP address of 130.85.105.3 and a subnet mask of 255.255.255.0. The router interface supporting that building has an IP address of 130.85.105.1 and a subnet mask of 255.255.255.0. When the router interface sees a packet with Grivita destination address it performs the logical and comparison on both its own IP address and subnet mask as well as the destination addresses IP address. It then compares the two resulting values, if they are equal the router knows the packet is on the same network and does not need to be forwarded. In the examples that follows are shown the both cases. In the sample the addresses expressed as dotted decimal addresses are translated into the equivalent of that in binary to easy apply the bitwise logical And operation (anded):

a) the result of applying the bitwise And operation produces the same value



Router address	Grivita address
130.085.105.1	130.085.105.3
10000010.01010101.01101001.00000001	10000010.01010101.01101001.00000011
Anded with	
subnet mask	subnet mask
255.255.255.0	255.255.255.0
11111111. 11111111. 11111111.00000000	11111111. 11111111. 11111111.00000000
Result	
10000010.01010101.01101001.00000000	10000010.01010101.01101001.00000000
130.085.105.0	130.085.105.0

b) the result of applying the bitwise And operation do not produces the same value*)



Router address		Polytechnic University address	
130.085.105.1		130.085.60.8	
10000010.01010101.01101001.00000001		10000010.01010101.00111100.00001000	
Anded with			
subnet mask		**subnet mask**	
255.255.255.0		255.255.255.0	
11111111. 11111111. 11111111.00000000		11111111. 11111111. 11111111.00000000	
Result			
10000010.01010101.01101001.00000000		10000010.01010101. 00111100.00000000	
130.085.105.0		130.085.060.0	

*) The results are not equal and the router must consult it's routing table to forward the packet on to the next destination.

2.1.4 The DNS

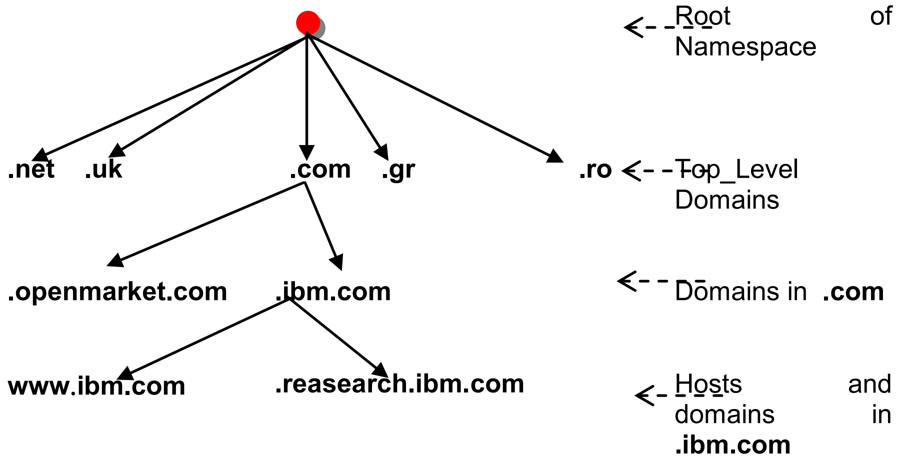
The usage of these dotted-decimal addresses can be very restrictive for common users. In Internet the common users access the servers and other shared resources by using **names**, almost of the time meaningful names. These names are associated to the dotted-decimal address of the station and are allowed and managed by DNS (Domain Name System).

DNS, was specified in 1983 and, allows the mapping of symbolic names to Internet addresses. Originally was realized statically in a centralized file (in Linux can be done statically in /etc/hosts). As Internet grew there was a need for a dynamic distributed system.

DNS defines:

- A hierarchical namespace for hosts;
- A host table implemented as a distributed database;
- Library routines for access ;
- Routing for e-mail;
- A protocol to exchange naming information.

The hierarchy structure for DNS is:



The DNS:

- Is organized as tree of domains with ascending authority;
- Offers two types of top-level domains
    - 3 letter in US (such as com, edu, gov, mil, net, org, intr, arpa etc). The "three" letters (can be more than three) can be used worldwide;
    - 2 letter national (ISO – such as ro, fr, uk, gr, us etc);
- Contains second level domains assigned by RIR (Regional Internet Registry), InterNIC (Internet Network Information Center) or RIPE in Europe, for example;
- Allows creating as desired, by organizations having second level domains, lower level sub-domains (e.g. ie.ase.ro)

- Allows delegating authority to create further sub-domains (e.g. vb.ie.ase.ro).

Note that domains reflect organizational structure whereas IP addresses reflect network connectivity (for routing purposes). These are often the same but do not need to be.

The letters defining a domain are used as Internet Domain Name Suffixes. The list of common Internet Domain Name Suffixes is shown in table 2.6.

Table 2.6 **Common Internet Domain Name Suffixes (http://www.icann.org)**

Ending	Kind of application
.arts	**Cultural groups**
.arpa	**ARPAnet site (USA)**
. aero	**Air-transport industry**
. asia	**Asian Countries**
.biz	**Restricted to Business**
.com	**General business and individuals**
. coop	**Cooperatives**
.edu or .ac	**Schools/Educational sites**
. eu	**European Countries**
.firm	**Businesses**
.gov	**Government**
.info	**Information services (unrestricted use)**
.int	**International Institutions**
.jobs	**Human resource management**
.mil	**Military (USA)**
.mobi	**Mobile**
.museum	**Museums**
.net	**Internet service providers or general/administrative network**
.nom	**Individuals**
.name	**Individuals**
.nato	**NATO site**
.org	**Organizations**
.pro	**Accountants, lawyers, and physicians**
.rec	**Recreation sites**
.store	**Retailers**
.travel	**Travel related business**
.web	**Web-related organizations**
.ro, .fr, .deu, .uk …	**The country domain**

There are a number of "root nameservers" in existence in various corners of the Internet which store the ultimate information for the root domain, as well as zones for a handful of top-level domains. Certain organizational units, such as countries and universities, have delegation of domains underneath the root and top-level domains. Entities wishing domain names must register, and perhaps receive delegation of, their domains from the appropriate registry.

2.1.5 URL

Web pages and related files are located and accessed in Internet by means of special constructions called URL. Internet addresses are separate and distinct from the domain names used as **U**niform **R**esource **L**ocators (URLs) through which you specify Web pages. The domain names give you a handle with a natural-language look. Internet addresses are, like everything in computing, binary codes. Even domain names are running short. Finding a clever and meaningful name for a web site is a challenge that's ever increasing. Believing that one of the problems in the shortage of URLs has been the relatively few suffixes available, one of the coordinating agencies for Internet names, the International Ad Hoc Committee, proposed seven additional suffixes in addition to the six already in use in the U.S. and the national suffixes used around the world (the ISO country two letters such as .ro for Romania, .us for United States, .uk for United Kingdom, .fr for France, and so on).

URL is an acronym for Uniform Resource Locator. URL is expressed as a character string that supplies the Internet address of a site or of a www resource. The general syntax of URL is:
communicationservice://hostname[:portnumber]/pathname/resourcename

In a Web page the links are represented by specially formatted text strings or by graphical elements that, when acted (by a mouse click, for example) displays more text or graphics. This files tagged by links can be represented by other Web pages or any kind of files such as graphic, image, sound, video, data fill-in forms, Java applets, movies and any kind of necessary file. A hypothetical URL can take the following general structure:

$$\underbrace{\textit{http://}}_{\text{protocol}}\underbrace{\textit{www.sels.ase.ro}}_{\text{domain}}\underbrace{\textit{/courses/generalinformatics/}}_{\text{directory path}}\underbrace{\textit{index.html}}_{\text{file name}}$$

An URL address can consist of five parts: protocol, domain, directory path, file name and anchor. For these elements a brief description follows:

- **protocol**: represented by rules that governs the data transfer in the network. Internet uses for Web pages (HTML pages) the http (HyperText Transport Protocol) – the word in the example URL *http:*
- **domain**: represented by the name of the host computer (hostname) and the Internet namespace - *www.sels.ase.ro*

- **directory path**: the absolute or relative location of the file - *courses/generalinformatics*
- **file name**: the web page, graphic, or sound file - *index.html*
- **anchor**: a marker which identifies a location inside a file (like a bookmark in normal documents) to which you can link. Once an anchor is placed in a location you can create a link to that spot.

The most common URL type is:

- *file://* - a local URL located in your hard drive(s) for example file://c|/index.htm that points the file called index.htm stored in the root of the local drive C: (Windows, MS-DOS);
- *http://* - which gives the Internet address of a Web page (hypertext URLs);
- *gopher://* - gives the Internet address of a Gopher directory. Gopher is a system used to locate and transfer information that index the filenames in Internet. The syntax of a gopher URL is gopher://hostname:port/filename, where hostname is the name of the host computer (that usually is a LAN), port is the address of his port;
- *telnet://* - allows connect you in real time with another computer in Internet and then to use that computer as you use a local one. For example for networks running under UNIX operating system (and clones), the syntax for telnet:// is telnet:// or tn3270:// followed by the name of the computer we want to connect to.
- *ftp://* - which gives the Internet address of a FTP resource. FTP - File Transport Protocol – is the common command set used to upload/download files to/from Web sites.

URI – Uniform Resource Identifiers

Uniform Resource Identifiers (URI) provide a simple and extensible means for identifying a resource. A URI is a compact string of characters for identifying an abstract or physical resource.

URI's have been known by many names: WWW addresses, Universal Document Identifiers, Universal Resource Identifiers, and finally the combination of Uniform Resource Locators (URL) and Names (URN).

> [T. Berners-Lee – RFC 1630] The web is considered to include objects accessed using an extendable number of protocols, existing, invented for the web itself, or to be invented in the future. Access instructions for an individual object under a given protocol are encoded into forms of address string. Other protocols allow the use of object names of various forms. In order to abstract the idea of a generic object, the web needs the concepts of the universal set of objects, and of the universal set of names or addresses of objects.
>
> A Universal Resource Identifier (URI) is a member of this universal set of names in registered name spaces and addresses referring to registered protocols or name spaces. A Uniform Resource Locator URL), defined

elsewhere, is a form of URI which expresses an address which maps onto an access algorithm using network protocols.

URI are characterized by the following definitions for the words that gives his name:

1. *Uniform* - uniformity provides several benefits: it allows different types of resource identifiers to be used in the same context, even when the mechanisms used to access those resources may differ; it allows uniform semantic interpretation of common syntactic conventions across different types of resource identifiers; it allows introduction of new types of resource identifiers without interfering with the way that existing identifiers are used; and, it allows the identifiers to be reused in many different contexts, thus permitting new applications or protocols to leverage a pre-existing, large, and widely-used set of resource identifiers.
2. *Resource* - a resource can be anything that has identity.

   Familiar examples include an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), and a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources. The resource is the conceptual mapping to an entity or set of entities, not necessarily the entity which corresponds to that mapping at any particular instance in time. Thus, a resource can remain constant even when its content - the entities to which it currently corresponds - changes over time, provided that the conceptual mapping is not changed in the process.
3. *Identifier* - an identifier is an object that can act as a reference to something that has identity.

   In the case of URI, the object is a sequence of characters with a restricted syntax. Having identified a resource, a system may perform a variety of operations on the resource, as might be characterized by such words as `access', `update', `replace', or `find attributes'.

A URI can be classified as a locator, a name, or both:

- the term "Uniform Resource Locator" (URL) refers to the subset of URI that identify resources via a representation of their primary access mechanism (e.g., their network "location"), rather than identifying the resource by name or by some other attribute(s) of that resource.

Although many URL schemes are named after protocols, this does not imply that the only way to access the URL's resource is via the named protocol. Gateways, proxies, caches, and name resolution services might be used to access some resources, independent of the protocol of their origin, and the resolution of some URL may require the use of more than one protocol (e.g., both DNS and HTTP are typically used to access an "http" URL's resource when it can't be found in a local cache).

- the term "Uniform Resource Name" (URN) refers to the subset of URI that are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable.

A URN differs from a URL in that it's primary purpose is persistent labeling of a resource with an identifier. That identifier is drawn from one of a set of defined namespaces, each of which has its own set name structure and assignment procedures. The "urn" scheme has been reserved to establish the requirements for a standardized URN namespace.

The following examples illustrate URI that are in common use:

ftp://ftp.ie.ase.ro/courses/generalinformatics.pdf - ftp scheme for File Transfer Protocol services

gopher://spinaltap.micro.umn.edu/00/Weather/California/Los%20Angeles - gopher scheme for Gopher and Gopher+ Protocol services

http://www.math.uio.no/faq/compression-faq/part1.html - http scheme for Hypertext Transfer Protocol services

mailto:courseadmin@ie.ase.ro - mailto scheme for electronic mail addresses

news:comp.infosystems.www.servers.unix - news scheme for USENET news groups and articles

telnet://melvyl.ucla.edu/ - telnet scheme for interactive services via the TELNET Protocol

2.2 Service protocols

The web server, whose base plate described in the §2.1.1, uses at the lower level the HTTP protocol that allows users to request services from his part. A web server may offer a lot of other specialized services, defined as protocols. The HTTP protocol is the core of all that protocols. We introduce in this paragraph these services and the positioning of that ones relatively to HTTP protocol.

2.2.1 TCP/IP - HTTP

Figure 2.8 shows the relationship between TCP/IP (see §2.1.1) protocols and HTTP (see §2.1.2) protocol. All user requests addressed to the HTTP server and the responses of this one are send, respectively, received by intermediate of Internet TCP/IP protocols. The common way the user interacts with the web server is the usage of a general web browser. HTTP is the protocol that governs how web browsers (clients) and web servers talk to each other. All messages sent between browsers and servers must be formatted according to the HTTP specification. The HTTP commands allow an

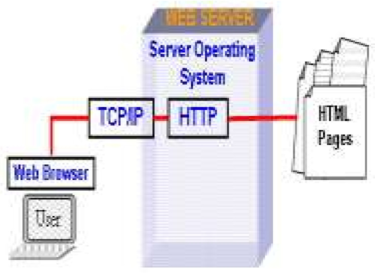


Figure 2.8 The positioning of TCP/IP and HTTP protocols

application to interpret a page together with his HTML (HyperText Markup Language) links. The HTTP server manages, interprets and acts the HTTP commands.

2.2.2 SMTP/POP

Figure 2.9 shows the position of the mail service and his associated protocols SMTP and POP.

SMTP - Simple Mail Transfer Protocol - is an electronic mail protocol that allows mails to travel over the internet.

POP – Post Office Protocol – is a utility that allows users to receive their mails.

SMTP. The objective of the Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently.

SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel. An important feature of SMTP is its capability to transport mail across networks, usually referred to as "SMTP mail relaying". A network consists of the mutually-TCP-accessible hosts on the public Internet, the mutually-TCP-accessible hosts on a firewall-isolated TCP/IP Intranet, or hosts in some other LAN or WAN environment utilizing a non-TCP transport-level protocol. Using SMTP, a process can transfer mail to another process on the same network or to some other network via a relay or gateway process accessible to both networks. In this way, a mail message may pass through a number of intermediate relay or gateway hosts on its path from sender to ultimate recipient.

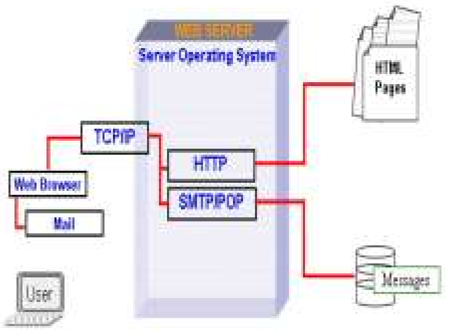


Figure 2.9 The positioning of TCP/IP and HTTP protocols

The SMTP design can be pictured as in figure 2.10. When an SMTP client has a message to transmit, it establishes a two-way transmission channel to an

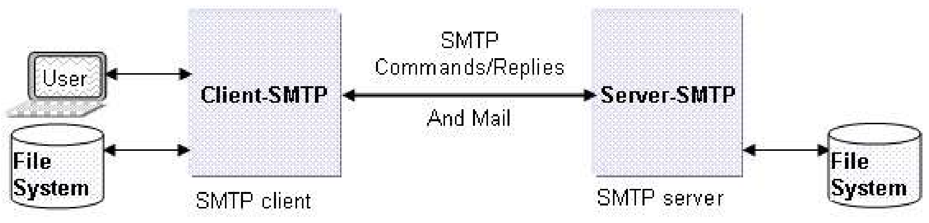


Figure 2.10 SMTP

SMTP server. The responsibility of an SMTP client is to transfer mail messages to one or more SMTP servers, or report its failure to do so. Message transfer can occur in a single connection between the original SMTP-sender and the final SMTP-recipient, or can occur in a series of hops through intermediary systems.

When the user agent on a client host wishes to enter a message into the transport system, it establishes an SMTP connection to its relay host and sends all mail to it. An SMTP client determines the address of an appropriate host running an SMTP server by resolving a destination domain name to either an intermediate Mail eXchanger host or a final target host. An SMTP server may be either the ultimate destination or an intermediate "relay" (that is, it may assume the role of an SMTP client after receiving the message) or "gateway" (that is, it may transport the message further using some protocol other than SMTP). SMTP commands are generated by the SMTP client and sent to the SMTP server. SMTP replies are sent from the SMTP server to the SMTP client in response to the commands. Message transfer can occur in a single connection between the original SMTP-sender and the final SMTP-recipient, or can occur in a series of hops through intermediary systems. In either case, a formal handoff of responsibility for the message occurs: the protocol requires that a server accept responsibility for either delivering a message or properly reporting the failure to do so.

Once the transmission channel is established and initial handshaking completed, the SMTP client normally initiates a mail transaction.

Such a transaction consists of a series of commands to specify the originator and destination of the mail and transmission of the message content (including any headers or other structure) itself. When the same message is sent to multiple recipients, this protocol encourages the transmission of only one copy of the data for all recipients at the same destination (or intermediate relay) host.

The server responds to each command with a reply; replies may indicate that the command was accepted, that additional commands are expected, or that a temporary or permanent error condition exists.

Once a given mail message has been transmitted, the client may either request that the connection be shut down or may initiate other mail transactions. In addition, an SMTP client may use a connection to an SMTP server for ancillary services such as verification of email addresses or retrieval of mailing list subscriber addresses.

This transmission normally occurs directly from the sending user's host to the receiving user's host when the two hosts are connected to the same transport service. When they are not connected to the same transport service, transmission occurs via one or more relay SMTP servers. An intermediate host that acts as either an SMTP relay or as a gateway into some other transmission environment is usually selected through the use of the domain name service (DNS) Mail eXchanger mechanism.

POP (Post Office Protocol). On certain types of smaller nodes in the Internet it is often impractical to maintain a message transport system (MTS). For example, a workstation may not have sufficient resources (cycles, disk space) in order to permit a SMTP server and associated local mail delivery system to be kept resident and continuously running. Similarly, it may be expensive (or impossible) to keep a personal computer interconnected to an IP-style network for long amounts of time (the node is lacking the resource known as "connectivity"). Despite this, it is often very useful to be able to manage mail on these smaller nodes, and they often support a user agent (UA) to aid the tasks of mail handling. To solve this problem, a node which can support an MTS entity offers a maildrop service to these less endowed nodes. The Post Office Protocol - Version 3 (POP3) [RFC1733] is intended to permit a workstation to dynamically access a maildrop on a server host in a useful fashion. Usually, this means that the POP3 protocol is used to allow a workstation to retrieve mail that the server is holding for it. POP3 is not intended to provide extensive manipulation operations of mail on the server; normally, mail is downloaded and then deleted.

> Initially, the server host starts the POP3 service by listening on TCP port 110. When a client host wishes to make use of the service, it establishes a TCP connection with the server host. When the connection is established, the POP3 server sends a greeting. The client and POP3 server then exchange commands and responses (respectively) until the connection is closed or aborted.
>
> Commands in the POP3 consist of a case-insensitive keyword, possibly followed by one or more arguments. All commands are terminated by a CRLF pair (Carriage Return and Line Feed). Keywords and arguments consist of printable ASCII characters. Keywords and arguments are each separated by a single SPACE character. Keywords are three or four characters long. Each argument may be up to 40 characters long. Responses in the POP3 consist of a status indicator and a keyword possibly followed by additional information. All responses are terminated by a CRLF pair. Responses may be up to 512 characters long, including the terminating CRLF. There are currently two status indicators: positive ("+OK") and negative ("-ERR"). A POP3 session progresses through a number of states during its lifetime. Once the TCP connection has been opened and the POP3 server has sent the greeting, the session enters the AUTHORIZATION state. In this state, the client must identify itself to the POP3 server. Once the client has successfully done this, the server acquires resources associated with the client's maildrop, and the session enters the TRANSACTION state. In this state, the client requests actions on the part of the POP3 server. When the client has issued the QUIT command, the session enters the UPDATE state. In this state, the POP3 server releases any resources acquired during the TRANSACTION state and says goodbye. The TCP connection is then closed.

2.2.3 FTP

The FTP (File Transfer Protocol) protocol is used in Internet as a standard for transfer files (for moving files across the Internet). FTP is available as a feature of web browsers for downloading and/or uploading files (figure 2.11).

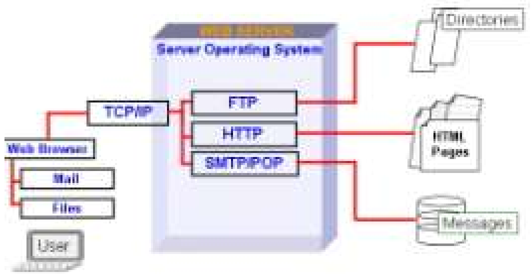


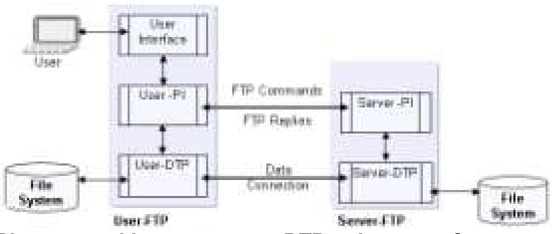
Figure 2.11 The FTP positioning

A FTP site is a server offering libraries of files (images, movies, applications etc). The FTP servers are real mines of freeware (software with no charge for usage) and shareware (applications available at a very lower price) software, images, video, movies, music etc.

The objectives of File Transfer Protocol (FTP), as defined in its specifications, are:

1) to promote sharing of files (computer programs and/or data),
2) to encourage indirect or implicit (via programs) use of remote computers,
3) to shield a user from variations in file storage systems among hosts, and
4) to transfer data reliably and efficiently.

FTP, though usable directly by a user at a terminal, is designed mainly for use by programs. Figure 2.12 describes a model for the FTP service, in which:



PI – protocol interpreter DTP – data transfer process

Figure 2.12 The FTP service

- The user and server sides of the protocol have distinct roles implemented in a user protocol interpreter (User-PI) and a server protocol interpreter (Server-PI);
- The user protocol interpreter (User-PI) initiates the control connection from its port U to the server-FTP process, initiates FTP commands, and governs the user data transfer process (User-DTP) if that process is part of the file transfer;
- The user data transfer process (User-DTP) "listens" on the data port for a connection from a server-FTP process. If two servers are transferring data between them, the user-DTP is inactive;
- The server data transfer process (Server-DTP), in its normal "active" state, establishes the data connection with the "listening" data port. It sets up parameters for transfer and storage, and transfers data on command from its protocol

interpreter (PI). The DTP can be placed in a "passive" state to listen for, rather than initiate a connection on the data port;
- The FTP commands specify the parameters for the data connection (data port, transfer mode, representation type, and structure) and the nature of file system operation (store, retrieve, append, delete, etc.). The User-DTP or its designate should "listen" on the specified data port, and the server initiate the data connection and data transfer in accordance with the specified parameters. The data port need not be in the same host that initiates the FTP commands via the control connection, but the user or the user-FTP process must ensure a "listen" on the specified data port. The data connection may be used for simultaneous sending and receiving.

In the model described in Figure 2.12, the user-protocol interpreter (User-PI) initiates the control connection. The control connection follows the Telnet protocol. At the initiation of the user, standard FTP commands are generated by the User-PI and transmitted to the server process via the control connection (The user may establish a direct control connection to the server-FTP, from a Telnet terminal for example, and generate standard FTP commands independently, bypassing the user-FTP process). Standard replies are sent from the Server-PI to the User-PI over the control connection in response to the commands.

Telnet allows someone to be on to be on one computer system while doing work on another. Telnet is the protocol that establishes an error-free, rapid link between two computers, allowing you, for example, to log on to your business computer from a remote computer when you are on the road or working from home. Is possible also to log in and use third-party computers that have been made accessible to the public. Telnet uses the computer address you supply to locate the computer you want to reach and connect you to it.

In the situation a user wish to transfer files between two hosts (neither of which is a local host) the user must sets up control connections to the two servers and then arranges for a data connection between them. In this manner, control information is passed to the User-PI but data is transferred between the server data transfer processes following the model of this server-server interaction as shown in figure 2.13.

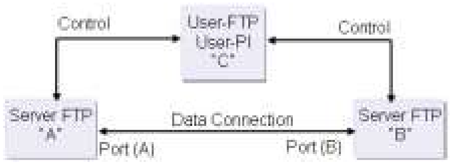


Figure 2.13 The FTP server-server interaction model

Using FTP line commands

The operating systems offer a tool (named by extension FTP) that allows users to type the FTP commands to the keyboard. By using that tool the user can connect to a FTP server to list the files (dir, mdir and ls commands in figure 2.15), to download files (get for singularly file, mget for many files) or, if it has the necessary rights to upload files (put or mput), to create/delete directories (mkdir, rmdir), change the name (rename) and so on.

The way to use the commands and specify their parameters follows the same rules as for the command line commands (MS-Dos prompt option on Start, Programs, Accessories in Windows xx Operating Systems).

Figure 2.14 shows the user model for interaction with the FTP interpreter and figure 2.16 lists the commands available in the Windows Millennium FTP tool.

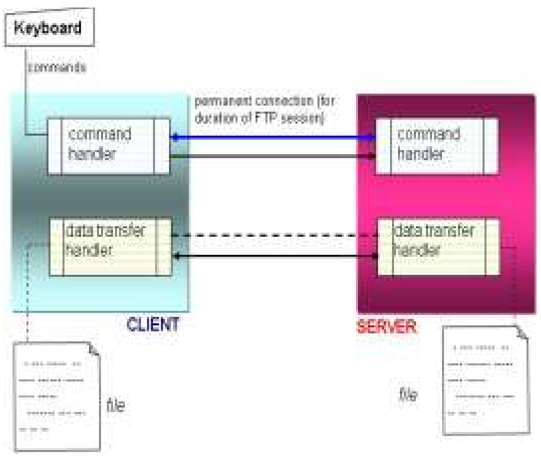


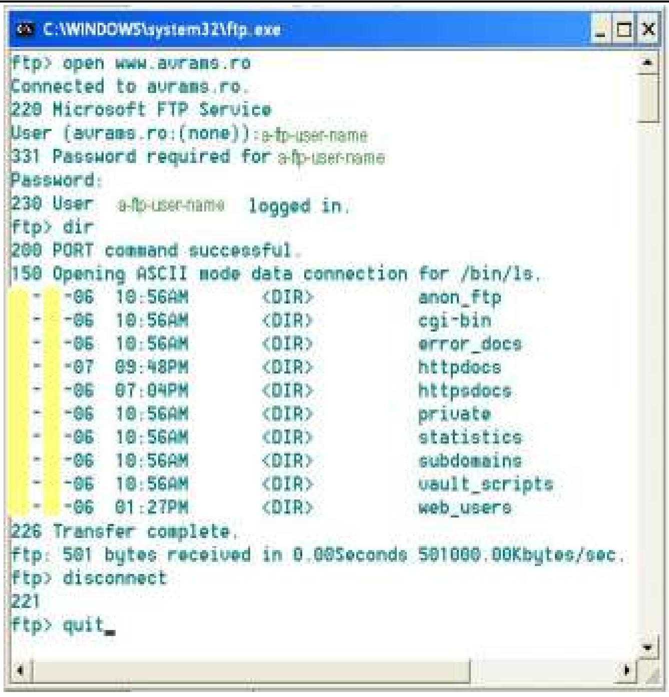
Figure 2.14 The user interaction with the FTP tool

```
C:\>ftp 127.1.80.1
ftp> ?
Commands may be abbreviated.  Commands are:
!            delete       literal      prompt       send
?            debug        ls           put          status
append       dir          mdelete      pwd          trace
ascii        disconnect   mdir         quit         type
bell         get          mget         quote        user
binary       glob         mkdir        recv         verbose
bye          hash         mls          remotehelp
cd           help         mput         rename
close        lcd          open         rmdir
ftp> ? mdir
mdir         List contents of multiple remote directories
ftp> quit
```

Figure 2.15 The FTP tool commands

The virtual file structure that FTP supports is not a general one but cover a wide range of possible files. FTP must consider the following attributes:

- the File Type that can be:
  - character files which contains only characters (transmissible and printable) such as text files or html documents (ASCII);
  - binary files which are considered as stream of bits such as executable files, image files, archive files etc or, in other words, any non text or html file (BINARY);
- the File Structure that can be:



```
C:\WINDOWS\system32\ftp.exe
ftp> open www.avrams.ro
Connected to avrams.ro.
220 Microsoft FTP Service
User (avrams.ro:(none)):a-ftp-user-name
331 Password required for a-ftp-user-name
Password:
230 User  a-ftp-user-name  logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
  -  -06  10:56AM       <DIR>          anon_ftp
  -  -06  10:56AM       <DIR>          cgi-bin
  -  -06  10:56AM       <DIR>          error_docs
  -  -07  09:48PM       <DIR>          httpdocs
  -  -06  07:04PM       <DIR>          httpsdocs
  -  -06  10:56AM       <DIR>          private
  -  -06  10:56AM       <DIR>          statistics
  -  -06  10:56AM       <DIR>          subdomains
  -  -06  10:56AM       <DIR>          vault_scripts
  -  -06  01:27PM       <DIR>          web_users
226 Transfer complete.
ftp: 501 bytes received in 0.00Seconds 501000.00Kbytes/sec.
ftp> disconnect
221
ftp> quit_
```

Figure 2.16 The FTP tool commands

- unstructured files which are considered as a stream of bytes;
- record structured files in the case of character files;

- the Transmission Mode for which a choice can be made for transmitting the file:
    - as a stream of bytes;
    - as a series of blocks of bytes.

FTP tool requires, when starting, a permanent connection between the command handlers of both client and server (figure 2.14). Both client and server process have a separate component that takes care of all component that is responsible for data transfer. When starting FTP session a permanent setup connection is set up the two command handlers. Each time a file is transferred during session a separate connection between data transfer handlers is set and subsequently closed after file transfer has taken place. Figure 2.16 is a snapshot of a ftp session realized in Windows XP environment by using `ftp.exe` tool (Start, Run, ftp).

2.2.4 NNTP

Network News Transfer Protocol (NNTP) specifies a protocol for the distribution, inquiry, retrieval, and posting of news articles using a reliable stream-based transmission of news among the Internet community. NNTP is designed so that news articles are stored in a central database (News Database - figure 2.17) allowing a subscriber to select only those items he wishes to read. There is a central repository of the news articles in one place (customarily a spool directory of some sort), and a set of programs that allow a subscriber to select those items he wishes to read. The database is provided with indexing, cross-referencing, and expiration of aged messages.

The news server uses a stream connection (such as TCP) and SMTP-like commands and responses. It is designed to accept connections from hosts, and to provide a simple interface to the news database. This server is only an interface between programs and the news databases. It does not perform any user interaction or presentation-level functions. These "user-friendly" functions are better left to the client programs, which have a better understanding of the environment in which they are operating.

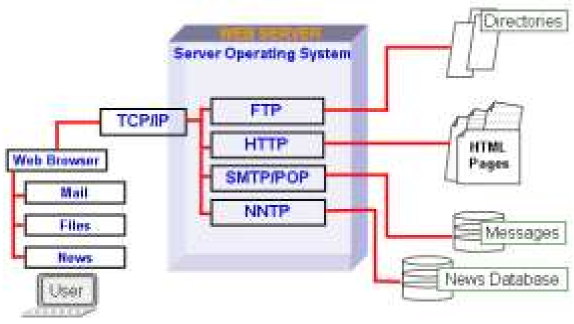


Figure 2.17 NNTP positioning

Using NNTP, hosts exchanging news articles have an interactive mechanism for deciding which articles are to be transmitted. A host desiring new news, or which has new news to send, will typically contact one or more of its neighbors using NNTP. First it will inquire if any new news groups have been created on the serving host by means of the NEWGROUPS command. If so, and those are appropriate or desired (as established by local site-dependent rules), those new newsgroups can be created.

The client host will then inquire as to which new articles have arrived in all or some of the newsgroups that it desires to receive, using the NEWNEWS command. It will receive a list of new articles from the server, and can request transmission of those articles that it desires and does not already have.

Finally, the client can advise the server of those new articles which the client has recently received. The server will indicate those articles that it has already obtained copies of, and which articles should be sent to add to its collection.

In this manner, only those articles which are not duplicates and which are desired are transferred.

There are popularly two methods of distributing news over Internet: the USENET news system and the Internet method of direct mailing (LISTSERV).

- USENET newsgroups (Forums) are worldwide discussion groups in which people share information and ideas on a defined topic. Discussions take places in large electronic bulletin boards where anyone can post messages for others to read.

- LISTSERV allows discussions or messaging to be conducted through predefined groups but uses e-mail mailing list servers instead of bulletin boards for communications.

2.2.5 RPC and Multimedia

Multimedia is the term used to describe software which (together with appropriate hardware) can interact with user through different techniques such as text, sound, animation or video. The type of hardware required to support multimedia includes sound and video card and capture using microphones, video cameras and scanners. Multimedia software is most common in home computers but also has business applications such as training courses and product promotions. Multimedia functions can be incorporated into both general-purpose software (e.g. word processors and e-mail can include multimedia elements) and application-specific software.

Usually the browser interprets the files with a *html*, *htm*, *gif* or *jpg* extension. If a hypertext link points to a file having another extension such as *pdf*, *mov, avi* or *doc* the browser requires an appropriate viewer or reader to read and interpret this file; these viewers are of two categories: plug-ins or add-ons. The **plug-in** is a small program or accessory that can be used to extend a web browser's capability. The difference between them is that the plug-in are integrated into the browser and the user cannot see the difference between this and the browser while the add-on are superposed to the browser for execution. Some of products fits in both categories as for example VDOlive and Real Audio or only in one category, as for example, Flash and Acrobat Reader that are add-on. The multimedia server offers together with the multimedia information the viewers, readers or specific software that can be accessed remotely, via RPC protocol, by the client (figure 2.18).

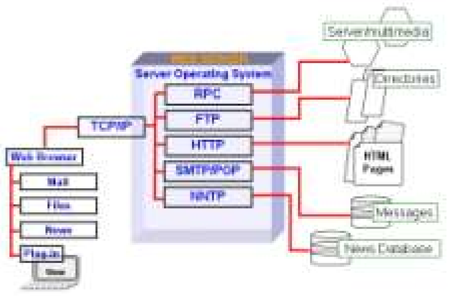


Figure 2.18 The RPC protocol

The remote procedure call (RPC) model is similar to the local procedure call model. In the local case, the caller places arguments to a procedure in some well-specified location (such as a result register) and then transfers control to the procedure, and eventually gains back control. At that point, the results of the procedure are extracted from the well-specified location, and the caller continues execution. The remote procedure call is similar, in that one thread of control logically winds through two processes - one is the caller's process, the other is a server's process. That is, the caller process sends a call message to the server process and waits (blocks) for a reply message. The call message contains, among other things, the procedure's parameters. The reply message contains, among other things, the procedure's results. Once the reply message is received, the results of the procedure are extracted, and caller's execution is resumed. On the server side, a process is dormant awaiting the arrival of a call message. When one arrives, the server process extracts the procedure's parameters, computes the results, sends a reply message, and then awaits the next call message.

The RPC protocol provides the fields necessary for a client to identify itself to a service and vice-versa. Security and access control mechanisms can be built on top of the message authentication.

The RPC call message has three unsigned fields: remote program number, remote program version number, and remote procedure number. The three fields uniquely identify the procedure to be called. Program numbers are administered by some central authority. Once an implementor has a program number, he can implement his remote program; the first implementation would most likely have the version number of 1.

Because most new protocols evolve into better, stable, and mature protocols, a version field of the call message identifies which version of the protocol the caller is using. Version numbers make speaking old and new protocols through the same server process possible.

2.2.6 Applications gateways

The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as HTTP or Web servers (figure 2.19). The three words that give the name to the standard describe his functionality:

- **Common** specifies a universal method for accessing CGI scripts, that allows to any user, does not mother the used platform, to exchange information with a CGI script;
- **Gateway** defines a bridge between CGI script, Web server and other CGI applications offering the possibility that external programs accept input data and transmit data to other applications;
- **Interface** that reduce the complexity of linking diverse applications to some basic actions describing how external programs can be accessed by clients. For almost Internet users (clients) the process is very simple: the client fill-in the fields of displayed form (by the browser) and press the submit button.

  Practically the process follows the steps:

  1. The client send data to the Web server;
  2. The Web server passes data to a CGI script;
  3. The CGI script process data received from server, eventually passes that data to another application and send a response to the Web server;

4. The Web server returns the response to the client (the response can be, for example, the result of querying a database, figure 2.20).

The CGI is a simple interface for running external programs, software or gateways under an information server in a platform-independent manner. The CGI allows an HTTP server and a CGI script to share responsibility for responding to client requests. The client request comprises a Uniform Resource Identifier (URI), a request method and information about the request provided by the transport protocol.

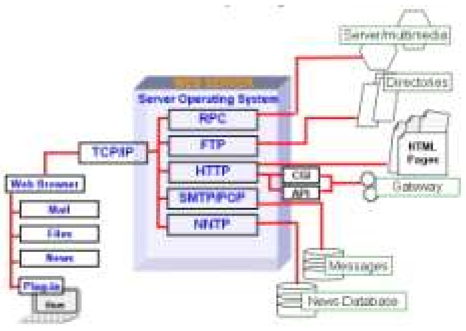


Figure 2.19 The CGI positioning

The CGI defines the abstract parameters, known as **meta-variables**, which describe a client's request. Meta-variables contain data about the request passed from the server to the script, and are accessed by the script in a system-defined manner.

The server is responsible for managing connection, data transfer, transport and network issues related to the client request, whereas the CGI script handles the application issues, such as data access and document processing.

A plain HTML document that the Web server retrieves is static, which means it exists in a constant state: a text file that doesn't change. A CGI program, on the other hand, is executed in real-time, so that it can output dynamic information.

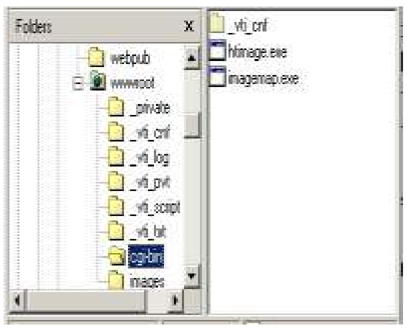


Figure 2.20 CGI-BIN on a Microsoft Personal Web Server

The server acts as an application gateway: it receives the request from the client, selects a CGI script to handle the request, converts the client request to a CGI request, executes the script and converts the CGI response into a response for the client. The script is invoked in a system-defined manner. Unless specified otherwise, the file containing the script will be invoked as an executable program (figure 2.20).

A CGI program can be written in any language that allows it to be executed on the system, such as C/C++, Fortran, PERL, TCL, any Unix/Linux shell, Visual Basic, JavaScript etc or, in a scripting language, such as PERL, TCL, or a Unix shell. The scripts must reside in a special directory so that the Web server knows to execute the program (this directory is administered by the webmaster and is called /cgi-bin, figure 2.20).

For example, the Client-Server dialog for processing a form (filled by the user in his browser, for example, like the registration forms displayed by many sites when a user access this for the first time requesting some services requiring authentication) follows the steps (figure 2.21):
1. The Client lunch a URI request that includes a form;
2. The Server receives the request, analyzes that and emits the form to be filled by the Client (user);
3. The Client fill-in the form and send that to Server;
4. The Server runs the CGI script (specified in the message received from Client) that realizes some processing on data contained by the form (e.g. queries a database, computes some values etc) and prepares the obtained results;
5. The CGI script transmits the results to the Server in a Server understandable format;
6. The Server receives the results and sends that to the Client.

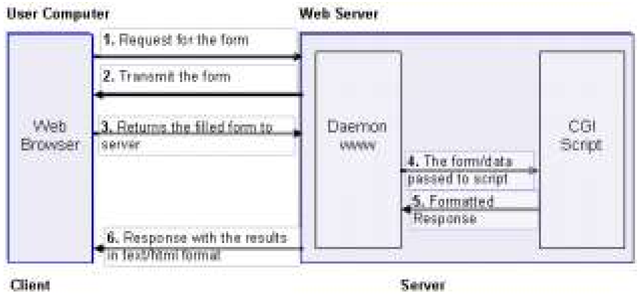


Figure 2.21 The Client-Server dialog when processing a form using CGI scripts

When a client activates a link to a CGI script the input data are passed to the server. The server associate (and assigns) the transmitted data (stored in meta-variables) with (to) predefined environment variables and verifies if some data are present to the standard input (stdin) device. The environment variables are passed to the CGI script (application). After the CGI script processes data it must return the results (output data) as an answer to client request; as a rule the answer is in a HTML document format that includes a header followed by an empty line and the content (body). The body is formatted accordingly to the information supplied in the header. The statements for server are specified in the CGI header by intermediate of a predefined set of meta-variables whose content defines the request from which:
1) *content-type* identifies the MIME data type of the response (e.g.: content-type: text/html);
2) *location* for defining the URL of the document returned to the client if the document not directly generated (created) by the script (e.g.: location: http//www.ie.ase.ro);
3) *status* providing the status information (error code and explanatory text).

The system software manufacturers offer different alternatives to CGI:
- NSAPI the API alternative to CGI proposed by Netscape for his HTTP servers;

- TSAPI the alternative to CGI proposed for Windows NT servers (Microsoft);
- ODBC (Open DataBase Connectivity) the interface for data access of Windows applications (Microsoft);
- JDBC (Java DataBase Connectivity) the interface for data access of Java applications (Sun);
- SQL (Structured Query Language) the standard language (ISO and ANSI) for data access (initially the SQL used only for relational database model but his extensions allow accessing data managed file managers or database management systems not necessarily for relational model).

2.2.7 Applets

An applet is a program written in the Java programming language that can be included in an HTML page (figure 2.22), much in the same way an image is included in a page. When you use a Java technology-enabled browser to view a page that contains an applet, the applet's code is transferred to your system and executed by the browser's Java Virtual Machine (JVM).

Applets have the file extension "class". Some applets consist of more than just one class file, and often other files need to be present for the applet to run (such as JPG or GIF images used by the applet). When you intent to pickup an available applet you must check the documentation for the applet to see if you have all files for it to run. Before embedding an applet on your page you need also to upload the required files to your server.

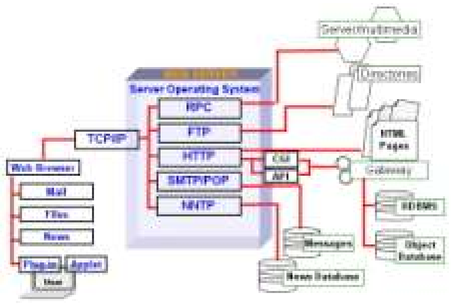


Figure 2.22 The applets positioning

You do not need to know Java to install applets on your pages. There are thousands of free applets available on the internet for almost any purpose. Most of them can be customized without programming. Most of today's browsers can run applets.

An applet can be embedded into a webpage. Usually the applet has several settings that will allow you to personalize it. For instance, if you insert an applet that will work as a menu, you can specify which options should be in the menu, and which pages should be loaded upon click on an option.

Since Java is a real programming language there aren't many limitations to it. Any program running on your computer could possibly have been made as an applet. Spreadsheets, wordprocessors, graphics programs... even entire browsers could be made with Java. However, most applets used on webpages serve much smaller purposes than the ones mentioned. The reason is simple: They need to be transferred through the internet, and therefore can't take up just any amount of space.

When you put an applet on your page you will need to save the applet on your server as well as the HTML page the applet is embedded in. When the page is loaded by a visitor the applet will be loaded and inserted on the page where you embedded it.

2.2.8 Wireless Web

Wireless Web denote Web based applications enabling users to access digital information from the Internet using wireless mobile communicating devices (such as PDAs, Internet enabled cell phones etc). These devices are characterized by tiny displays screens, low-bandwidth connections and minimal memory. The two main standards governing the Wireless Web for such devices are WAP and I-mode (figure 2.23 a and b).

WAP (Wireless Application Protocol) - is a system of protocols and technologies using WML (Wireless Markup Language).

WML is based on XML and optimized for tiny display and was designed only for describing data and not for defining the way data are displayed (as HTML do, for example). To speed the access, each WML file is referred to a 'deck' and consists of several cards that can be displayed sequentially without reconnecting. WAP architecture uses a built-in micro-browser to make a request in WML. The request (figure 2.23 a) is passed to a WAP gateway which retrieves the information from the Internet Server in either HTML or WML format. The Gateway converts HTML to WML so that the client can receive. WAP supports most wireless network standards and operating systems for handheld computing devices such as PalmOS and Windows Mobile 5 for pocket PC, for example.

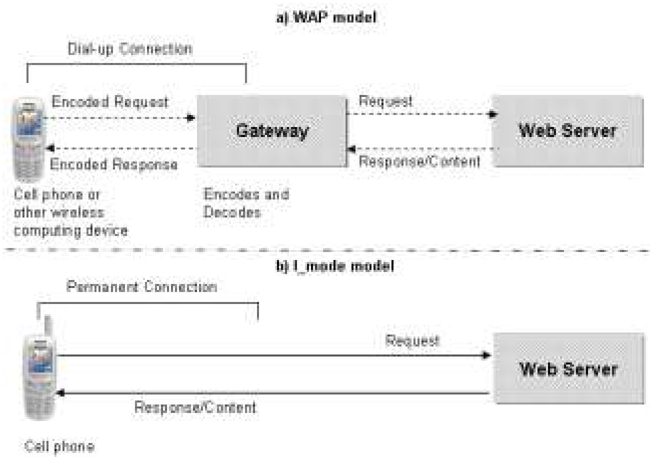


Figure 2.23 WAP and I-mode models

The WAP standard protocols stack looks like:

	Wireless Application Protocol (WAP)	Internet
	Wireless Application Environment (WAE)	HTML/Java
	Wireless Session Protocol (WSP)	HTTP

W M L	Wireless Transaction Protocol (WML)	
	Wireless Transport Layer (WTLS)	SSL
	Wireless Datagram Protocol (WDP)	TCP/IP
	GSM - Global System for Mobile Communication, 9.6-14.4 Kbps HSCSD – High Speed Circuit Switched Data, 38.4-56 Kbps GPRS – General Packet Radio Service, 43-170 Kbps (2.5G) EDGE – Enhanced Data Rates for Global Evolution, 384 Kbps UMTS – Universal Mobile Telecommunication System, 384-2000 Kbps (3G)	

(Mobile Access Technologies - the bearers of WML messages)

I-mode – is a standard developed by the Japan’s NTT DoCoMo mobile phone network for enabling cell phones to receive Web-based content and services (figure 2.23 b). The I-mode uses a compact HTML to deliver the content that allows easy transformation of traditional HTML pages in the compact HTML. I-mode uses a packet switching technology which allows users to permanently connect to the network and content providers to broadcast relevant information to users.

2.3 Web pages, sites and Web browsers – an introduction

Web pages and web site - definitions

The documents for World Wide Web (www) are known as Web pages and they are stored on an Internet server and displayed by a Web browser on your computer. Web browsers display Web pages by interpreting the special HyperText Markup Language (HTM or HTML) tags which are used to encode Web pages with display information.

Web pages usually are linked to many different files, such as graphic and multimedia files. You typically keep these files in a folder or set of folders on your disk drive, while you construct your Web site (this folder is known as local web site).

A Web site is defined as a collection of files that are linked to a central Web page, made available via the Web (the pages forms a cohesive collection of information). The Web server is a type of server dedicated to storing, transmitting and receiving the Web pages and Web related files (such GIF and JPEG graphics, AVI sound and images and so on).

The site’s collection of linked files and Web pages are typically tied together into a cohesive collection of information by a home page (generally called default.htm[l], index.htm[l] or simply home.htm[l]). The home page typically contains a topic list which links it to other Web pages in its Web site. All other pages, in a well designed Web site, must offer a button or a link to go back home (or that is provided by the Web browser). When you publish your Web site, you upload the local site folder (and its contents including subfolders) to a Web server, which contains the software that “serves” your Web pages out to Web browsers on computers that are connected to the Internet. Once your local site is published to the Web server it becomes a Web site. The main or home page of the Web site is accessed by using Internet URIs.

Web browsers

The www is accessed using a web browser. The interface used by a web browser makes use of hypertext linking techniques. A hypertext is a document that includes highlighted words or phrases. These highlighted sections represent links to other documents or sections of the same document. Clicking the mouse a above one of these links causes it to be activated. A link can be used to move to another document, transfer a file, view a section of video, listen to a sound file or carry out a number of other actions.

All web browsers provide users with a variety of tools that enable them to navigate through complex collections of WWW pages such as:

- **Navigation buttons** – these allows user to navigate backwards and forwards through the list of pages previously viewed. The browser can also provide additional buttons such as:
    - *Stop* – for canceling the action currently taken;
    - *Home* – for returning to the page designated by the user as "main page";
    - *Search* – this provides user access to search engines that can be used to locate specific information on the Internet;
    - *History* – for access to the list of pages previously viewed by the user;
    - *Address bar* – for directly entering (typing or choosing from list) the location of a WWW page.
- **Cache** – in order to increase the speed and efficiency the browser can use a temporary storage space to store the copies of any pages the user have viewed (if the user again access later a previously page viewed this one is retriever from that temporary space instead downloading from the original location).
- **Bookmarks** – allows user maintain a directory of web sites (the user can add, edit, delete and organizes addresses);
- **Security** – the modern browsers provide a range of security features, that can be used alone or in combination to obtain varying levels of security, such as:
    - *Digital ID* – provides a means of confirming the identity of a specific user through the use of a small data file called a personal certificate (the file contains encrypted information relating to the user's identity; that personal certificates are received and send by browser and this one is able to confirm his own identity to a third party or to verify the identity of a third part);
    - *Certificates* – a site certificates contains information regarding the identity of a particular site on the Internet (they encrypted to

protect the information they content and used for authenticity check when accessed by browsers);

  - *Ratings* – the ratings used to restrict access to inappropriate contents (such as pornography, for example). The check is based on a defined list of criteria defined by user in the browser to which the site ratings reports. If the site does not meet the criteria the access is denied.

- **Applets** – WWW pages can contain small programs that are activated when a page is accessed. Such programs can take a variety of forms and can include complete, self-contained applications known as applets. These programs are generally considered harmless, they can represent a potential security risk to an organization or individual. For that reason the browser must provide control over the operation of any applets embedded in a www page.
- **Plug-in** – a plug-in is a small program or accessory that can be used to extend the web browser capabilities.
- **Scripts** – all modern web browsers are capable of executing special commands that have been embedded within the body of a www page known as scripts.

In 2005 the classification under the number of peoples using of the first 5 search engines was:
- Google 33%
- Yahoo 31%
- MSN 15%
- AOL Search 10-15%
- Ask Jeevs 5.5%.

Figure 2.24 shows the classification of search engines under the preferences of users when realizing online research.

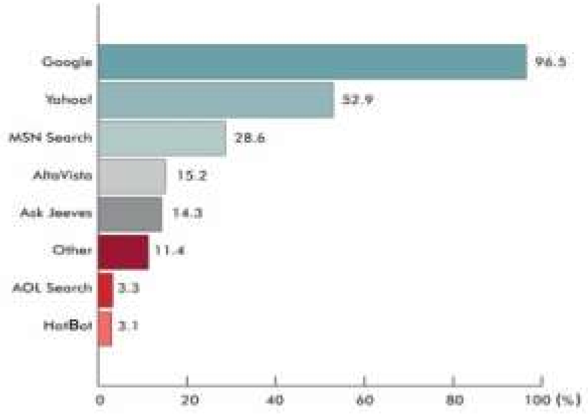


Figure 2.24 Classification of search engines under the preferences when realizing online research (Source: CMO Council, 2005)

Finding information on the Internet

Information can be found on the World Wide Web in the following main ways:

- By typing in the address bar the URI (URL) of a known web page;
- By using search engines (such as Google, Altavista, AskJeevs etc);
- By using directories / web catalogues / indexes (such as Yahoo);

- By ‘surfing’;
- By intermediate of Web guides (such as www.about.com and www.4anything.com).

Web addresses. The preferred method of reaching a web site is by typing the web address or URL (URI) directly into the web browser (for more information about URL/URI see §2.1.5. For example, by typing the Yahoo URL http://www.yahoo.com and pressing Go button the browser will open to you the Yahoo main page.

Search engines. The search engines provide an index of all words stored in WWW. Keywords typed by end user are matched against the index and the user is given a list of hyperlinks to pages containing the keywords. By following the hyperlink the user is taken to the relevant web page. One goal of all the search engines is to have the most complete index of files found on the web.

The search engine functionality can be described simply as: the search engine goes out into the Internet, follows the road signs and paths to get where it’s going, and collects all of the information in its path. From this point, the information is sent back to a group of servers where algorithms are applied in order to determine the importance of specific documents (to rank the pages and site). Essentially we have an entity that collects data, stores it, and then sorts through it to determine what’s important which it’s happy to share with others and what’s unimportant which it keeps tucked away. Both actions, the search on the web and the discovery of new pages, are realized by automated tools (software packages) called spiders or robots. In all major search engines the spiders crawl from one page to another following the links, as you would look down various paths along your way. An effective crawler needs to be able to index other information, including visible text, alt tags, images and even other non-HTML content such as PDF and word processor documents. Generally, the crawler gets a list of URL’s to visit and store that; it does not rank the pages, it only goes out and gets copies which it stores, or forwards to the search engine to later index and rank according to various aspects. Some of the most well known crawlers are Googlebot (Google), MSNBot (MSN), Slurp (Yahoo!) or Teoma (Ask Jeeves). Generally a crawler, when comes to visit a site, checks for a file called “robots.txt” that contains information about which files it can request and which files or directories not allowed to visit.

Most crawling search engines consist of the following main parts:
- *crawler* – a specialized automated program able to follows links found on web pages and to direct the spider by finding new sites for it to visit;

- *spider* – an automatic browser-like program that downloads documents found on the web by the crawler and store them (possibly in a compressed format – Google);
- *indexer* – a program that "reads" the pages that are downloaded by spiders and decides what the page is about and to calculate a quality ranking for each web page (for example, by considering the citations of the page together with the links going out of the page);
- *database* (the "index") – a simply storage of the pages downloaded and processed.
- *results engine* – that generates search results out of the database, accordingly to the user query.

Google Architecture (figure 2.25). Most of Google is implemented in C or C++ for efficiency and can run in either Solaris or Linux. In Google, the web crawling (crawler and spider: downloading of web pages) is done by several distributed crawlers (1). There is a URLserver (2) that sends lists of URLs to be fetched to the crawlers. The web pages that are fetched are then sent to the storeserver (3).

The storeserver then compresses and stores the web pages into a repository (4). Every web page has an associated ID number called a docID which is assigned whenever a new URL is parsed out of a web page. The indexing function is performed by the indexer (5) and the sorter (7).

The indexer performs a number of functions: reads the repository, un-compresses the documents, and parses them. Each document is converted into a set of word occurrences called hits. The hits

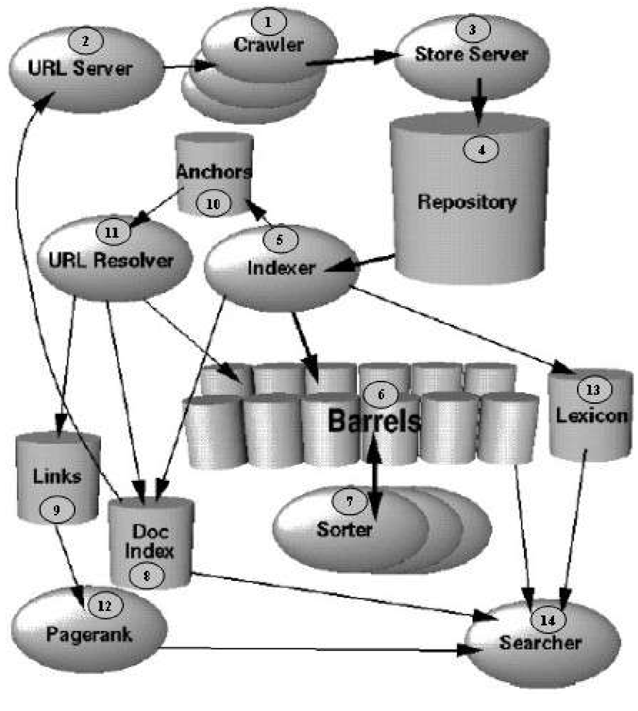


Figure 2.25 High level Google architecture

record the word, position in document, an approximation of font size, and capitalization.

The indexer distributes these hits into a set of "barrels" (6), creating a partially sorted forward index (8). The indexer performs another important function: it parses out all the links (9) in every web page and stores important information about them in an anchors file (10). This file contains enough information to determine where each link points from and to, and the text of the link.

The URLresolver (11) reads the anchors file and converts relative URLs into absolute URLs and in turn into docIDs. It puts the anchor text into the forward index, associated with the docID that the anchor points to. It also generates a database of links which are pairs of docIDs. The links database is used to compute PageRanks (12) for all the documents. The URLresolver reads the anchors file and converts relative URLs into absolute URLs and in turn into docIDs. It puts the anchor text into the forward index, associated with the docID that the anchor points to. It also generates a database of links which are pairs of docIDs. The links database is used to compute PageRanks for all the documents.

The sorter takes the barrels, which are sorted by docID (for simplicity purpose but the sort uses many other keys), and resorts them by wordID to generate the inverted index. This is done in place so that little temporary space is needed for this operation. The sorter also produces a list of wordIDs and offsets into the inverted index. A program called **DumpLexicon** (13) takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. The searcher (14) is run by a web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries.

Web catalogs or directories. Web directories provide a structured listing of web sites. They are grouped according to categories such as business, entertainment or sport. In turn each category is subdivided further (for example the sport category subdivided into football, rugby, swimming etc). The web catalogs (such as www.yahoo.com) work differently from search engines in that they have an hierarchy of information stored under different categories.

A *directory* is used to record information about a particular group of objects. The directory is not intended to be a general-purpose data store. Rather, it is a special type of information repository whose primary purpose is to efficiently store and retrieve information about objects relevant to a particular application or set of applications. A directory service is a physically distributed, logically centralized repository of infrequently changing data that is used to manage the entire environment. Directories are commonly used to store information about users, applications, and network resources such as file servers and printers.

Directories have five important characteristics:

- The storage of information is optimized so that it can be read much more frequently than it is written;
- Information is stored in a hierarchical fashion;
- Information in a directory is attribute-based;
- Directories provide a unified namespace for all resources for which they contain information;
- Directories can efficiently distribute information in a distributed system through replication.

A *directory service* stores and retrieves information from the directory on behalf of one or more authorized users. A traditional directory service provides a means for locating and identifying users and available resources in a distributed system. Directory services also provide the foundation for adding, modifying, removing, renaming, and managing system components without disrupting the services provided by other system components. Today's directory services are used to do the following:

- Store information about system components in a distributed manner. The directory is replicated among several servers so that a user or service needing access to the directory can query a local server for the information;
- Support common searching needs, such as by attribute (for example, "Find the phone number for James Smith") and by classification (for example, "Find all color printers on the third floor");
- Provide important information to enable single-user logon to services, resources, and applications;
- Enable a location-independent point of administration and management. Note that administrative tools do not have to be centrally located and managed;
- Replicate data to provide consistent access. Modifications made to any replica of the directory are propagated around the network so that any application accessing the directory anywhere sees consistent information after the change is propagated.

Web guides. The web guides (as for example www.about.com or www.4anyting.com) can be considered extensions of web catalogs because they consist of structured information about a particular topic providing articles, definitions, links and news about a particular topic. The web guides are edited by human who will create a structure and rate the information so that only relevant material will be included.

Other techniques of finding information. These are represented by the user applications that have an impact on the use of the Internet by organizations such as meta-search tools, offline readers or intelligent agents:

- *meta-search tools* that perform searches across a number of search engines (such as Ask Jeeves (www.askjeeves.com) and offers to user more comprehensive and up-to-date lists (the search results lists are collated and processed to remove duplicated items);

- *offline readers* that allows copying individual page, group of pages or entire sites in the local hard drive with preserving the entire functionality (including graphics, animation, scripts and any relevant data) and allowing the user to browse locally the site;
- *intelligent agents* represented by semiautonomous computer programs (as a software 'robot') capable of carrying out one or more tasks specified by user (such as monitoring news and locating the stories of interest to a specific user; searching for a specific product and return details about the manufacturer etc).

2.4 Web services – an introduction

The companies require both hardware and software to realize their processing. Instead of buying and installing software programs they can use Internet or private networks where, by paying a subscription, can rent the same functions from application service providers. An application service provider (ASP) is a business that delivers and manages applications and computer services from remote centers to multiple users via the Internet or private network. Today's Internet-driven business environment is changing so rapidly that getting a system up and running in three month instead of six could mean the difference between success and failure so that the ASP is a competitive alternative. The ASP enables also small and medium-sized companies to use applications that they otherwise could not afford. The applications offered in that way generally have a proprietary architecture and functionality and the customization do not produces, in all cases, the desired adaptation to the company needs. To avoid all that impediments a new way of deploying and using services emerges, known as Web services.

Web services are software components deliverable over the Internet that enable one application to communicate with another with no translation required. By allowing applications to communicate and share data regardless of operating system, programming language, or client device, Web services can provide significant cost savings, over traditional in-house development.

Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. They perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.

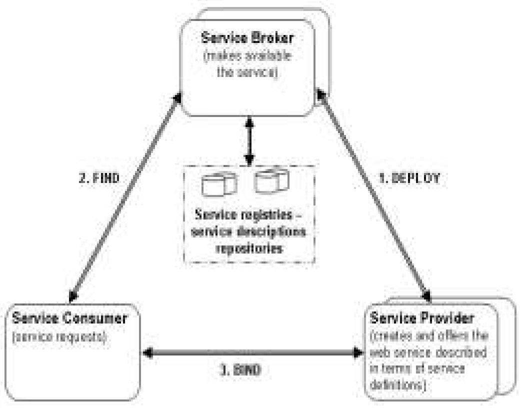
Web Services have emerged as a solution to problems associated with distributed computing. The previous technologies, primarily Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM), had some limitations. For example, neither has achieved complete platform independence or easy transport over firewalls.

Additionally, DCOM is not vendor independent, being a Microsoft product.

A Web Service forms a distributed environment, in which objects can be accessed remotely via standard interfaces. A Web service performs a specific task or a set of tasks, such as credit card processing, production scheduling, security, third-party billing and payment, for example. Web service uses a three-tiered model, defining three actors: service provider, service consumer, and service broker. This allows the Web Service to be a loose relationship, so that if a service provider goes down, the broker can always direct consumers to another one. Similarly, there are many brokers, so consumers can always find an available one. For communication, Web Services use open Web standards as: HyperText Transfer Protocol (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and the Universal Description, Discovery, and Integration (UDDI) project.

For both service consumer and service broker the Web service is available as an interface that describes a collection of operations that are network-accessible through standardized XML messaging. The Web service is described by using a standard, formal XML notation based on SOAP, called service description. The service description is realized by using WSDL (Web Services Description Language) and provides all of the details necessary to interact with the service, including message formats (that details the operations), transport protocols, and location.

The Web service is created, defined and deployed by **service provider** (figure 2.26). A service provider creates a Web service and its service definition and then **publishes** (1) the service in a service registry (or directory) based on the standard Universal Description, Discovery, and Integration (UDDI) specification.


Service Broker
(makes available the service)
2. FIND
1. DEPLOY
Service registries - service descriptions repositories
Service Consumer
(service requests)
3. BIND
Service Provider
(creates and offers the web service described in terms of service definitions)
Figure 2.26 Web services model

Once a Web service is published, a **service requester** may **find** (2) the service via the UDDI interface. The UDDI registry provides the service requester with a WSDL **service description** and a URI pointing to the service itself. The service requester may then use this information to directly **bind** (3) to the service and invoke it.

SOAP. Simple Object Access Protocol is a mechanism for sending information in an extensible format. It allows applications to pass data and instructions to one another. SOAP is the envelope syntax for sending and receiving XML messages with Web services. That is, SOAP is the "envelope" that packages the XML messages that are sent over HTTP between clients and Web services. SOAP can be used to send information or remote procedure calls encoded as XML. A typical SOAP message has the structure:

```
<SOAP:Envelope xmlns:SOAP=http://schemas.xmlsoap.org/soap/envelope/>
  <SOAP:Header>
                  <!-- SOAP header go here -->
  </SOAP:Header>
  <SOAP:Body SOAP:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/>
          <!-- SOAP body go here -->
  </SOAP:Body>
</SOAP:Envelope>
```

The SOAP Envelope is used for defining messages. It contains an optional SOAP Header and a SOAP Body. Messages are sent in the SOAP body, and the SOAP head is used for sending other information that wouldn't be expected in the body. For example, if the SOAP:actor attribute is present in the SOAP header, it indicates who the recipient of the message should be. SOAP handles data by encoding it on the sender side and decoding it on the receiver side. The data types handled by SOAP are based on the W3C XML Schema specification. Simple types include strings, integers, floats, and doubles, while compound types are made up of primitive types. Because they are text based, SOAP messages generally have no problem getting through firewalls or other barriers. They are the ideal way to pass information to and from web services.

WSDL. Web Service Description Language was created to provide information about how to connect to and query a web service. It allows Web service to be described so that it can be used by other applications. The WSDL file defines a service, made up of different endpoints, called ports. The port is made up of a network address and a binding. In turn, the binding identifies the binding style and protocol for each operation.

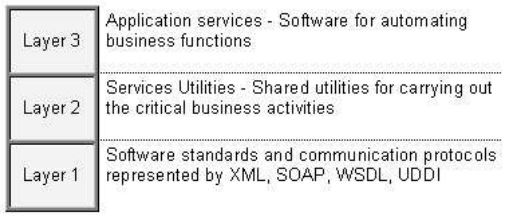
UDDI. Universal Description, Discovery, and Integration standard defines registries in which services can be published and found. It allows Web service to be listed in a directory of Web services so that they can be easily located. The UDDI specification was created by Microsoft, Ariba, and IBM. UDDI defines a data structure and Application Programming Interface (API). The UDDI plays the role of service broker and its function is to enable service consumer.

Web services use a “plug-and-play” like architecture on three layers, as shown in figure 2.27, that differs from the architecture of proprietary applications (such as CORBA or DCOM). These layers are:

- 1st layer consists of software standards and communication protocols such as XML, SOAP, WSDL and UDDI allowing information to be easily exchanged between applications;

- 2nd layer consists of a service grid to create the environment essential for carrying out critical business activities, utilities for transporting messages, utilities for identifying the available services, shared utilities for security, etc;
- 3rd layer consists of application services such as credit card processing, production scheduling or, generally, applications that automates specific business functions.

The companies can use the Web services in conjunction with their existing informatics systems by connecting this to outside services as the needs to do that arises.



Layer 3
Application services - Software for automating business functions
Layer 2
Services Utilities - Shared utilities for carrying out the critical business activities
Layer 1
Software standards and communication protocols represented by XML, SOAP, WSDL, UDDI

Figure 2.27 Web services architecture

3 BUSINESS CATEGORIES AND MODELS IN Internet

3.1 Business Categories

The Internet is creating a new 'universal' technology platform on which to build all sorts of new products, services, strategies, and organizations. It is reshaping the way information systems are being used in business and daily life. By eliminating many technical, geographic and cost barriers obstructing the global flow of information, the Internet is inspiring new uses of information systems and business model. The Internet provides the primary technology platform for the **digital firm**. In the following paragraphs are introduced the main concepts used to define business-categories and business environment in Internet.

Digital firm. A **digital firm** is one where nearly all of the organization's significant business relationships with customers, suppliers, and employees are digitally enabled and mediated. In a digital firm, any piece of information required to support key business decisions is available at any time and anywhere in the firm. Core business processes are accomplished through digital networks spanning the entire organization or linking multiple organizations.
Business processes. A **business processes** refer to the unique manner in which the work is organized, coordinated, and focused to produce a valuable product or service. Key corporate assets – intellectual property, core competencies, financial and human assets – are managed through digital means. Digital firms sense and respond to their environments far more rapidly than traditional firms, giving them more flexibility to survive in turbulent times. Digital firms are distinguished from traditional firms by their near total reliance on a set of information technologies to organize and manage.
Electronic market. By linking thousands of organizations and millions of individuals into a single network the Internet creates the foundation for a vast marketplace. An **electronic market** is an information system that links together many buyers and sellers to exchange information, products, services, and payments. It allows participating sellers and buyers to exchange goods and services with the support of information technology. Electronic markets have three main functions:
1) matching buyers and sellers;
2) facilitating commercial transactions;
3) providing legal infrastructure.

The main players in an electronic market are represented by businesses, individuals, and government organizations.

Electronic business (e-business). The extensive use of business information system through an organization is commonly referred to as **electronic business** or **e-business**. There are two common definition of e-business concept:

- "all electronically mediated information exchange, both within an organization and with external stakeholders supporting the range of business processes";
- "the use of Internet and other digital technology for organizational communication and coordination and management of the firm".

The processes that can be enhanced by e-business are standard processes that occur within any organization.

[BIS-TDM; KLJL] There are seven main operating organizational processes:

1. Understand markets and customers;
2. Develop vision and strategy;
3. Design products and services;
4. Market and sell;
5. Produce and deliver services;
6. Produce and deliver services (services organization);
7. Invoice and service customer.

These processes have companion the following support and management processes:

1. Develop and manage human resources;
2. Manage information;
3. Manage physical and financial resources;
4. Execute environmental management program;
5. Manage improvement and change.

According to the resource-based-theory e-business is about [PG-07]:

- developing and applying internal and external resources for competitive advantage;
- applying an e-business model that supports the current or desired value configuration of a value chain, value shop, and/or value network;
- making progress over time, as both technology and market conditions evolve .

This requires an understanding of system dynamics, where feedback loops between company actions and market reactions create or destroy infrastructure initiatives.

Electronic business brings a unique set of challenges to the information technology infrastructure (such as site capacity, scalability, and fault-tolerance, business information systems, etc) and his viability depends on the ability of the underlying systems to offer timely and reliable services.

Electronic commerce (e-commerce). Electronic commerce (e-commerce) is the process of buying and selling goods and services electronically with computerized business transactions using Internet, networks, and other digital technologies. It also encompasses activities supporting those market transactions, such as advertising, marketing, customer support, delivery, and payment. E-business requires some “specialized” network support such as intranet and extranet and refers to a broader definition of EC, not just the buying and selling of goods and services, but also servicing customers, collaborating with business partners, and conducting electronic transactions within an organization.

Intranet. An **intranet** is an internal network based on Internet and World Wide Web technology and standards (figure 3.1). An intranet allows organizations include all geographically distributed branches or divisions (the local networks deserving them) in a “global” area network (Wide Area Network, as suggested by the background cloud, inside of the organization boundaries) in which the users work as they are in a local area network. The local area networks can be connected by using organizations private connections or by using VPN over the Internet. The intranets are inexpensive, scalable to expand or contract as needs change, and accessible from most computing platforms. The intranets provide instant connectivity by the usage of the Web software that provide a uniform interface that allows unifying the computers, does not mother the platform to which they conform. The intranets provides a reach set of tools for creating collaborative environments in which members of an organization can exchange ideas, share information, and work together on common projects and assignments regardless of their physical location. Companies can connect their intranets to internal company transaction processing systems, enabling employees to take actions central to a company’s operations. Intranets can also be used to simplify and integrate business processes spanning more than one functional area and, in that way, a better informatic modeling of his information system. In companies where the processing is done manually or by non-integrated informatic support systems the same data are collected and processed in a redundant way and having a “distributed responsiveness”. The cross-functional processes can be coordinated electronically, increasing organizational efficiency (at least by eliminating redundant operations and tasks) and responsiveness (from a distributed one to a centralized one), and they can also be coordinated with business processes of other companies. By the usage of intranet the company can better model, in his integrated informatic system, the systems interdependencies and interrelationships or can integrate his individual software packages by defining electronic pathways for automatic online communication. The foreground graphic (the rectangles and lines connecting them) represent the various informatic subsystems deserving the company operation and control focused to a centralized control (does no matter if the applications processing and/or data-sources can run in parallel, distributed or centralized). The information inputs and outputs of the company do not exclude electronic data interchange between companies (suppliers and/or customers).

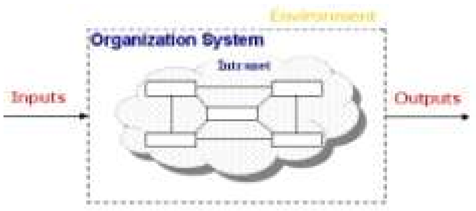


Figure 3.1 Intranet

Extranet. When a company gives access to third parties (such as suppliers and customers) to his intranet the obtained network is called extranet (figure 3.2). An **extranet** is a private intranet that is accessible to authorized outsiders, as suggested in the figure by the background cloud crossing the boundaries and including parts of external networks. The front (suppliers) and end (clients; buyers) sides linked to

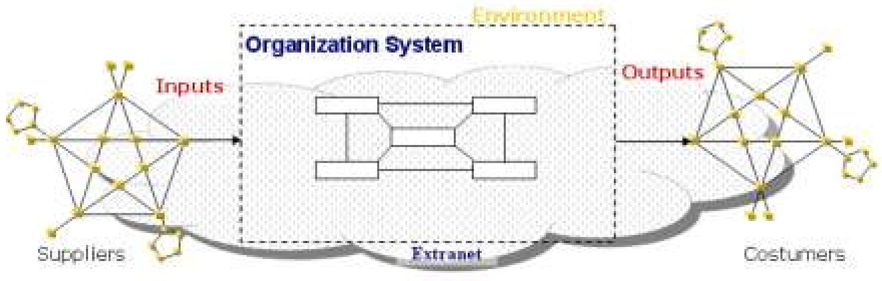


Figure 3.2 Extranet

complex networks not necessarily distinct (can be the same one Internet!) tray to suggest two of the roles played by the organization’s partners (suppliers and consumers). The usage of extranet allows an extension of the integration of different categories of informatic systems. This extended integration gives many benefits to the company (reducing time for operation, increasing speed of feed-back, increasing the accuracy of cross-boundary data and information etc). For example, if a supplier sends you the invoice electronically you do not need to retype the transactions included; instead of typing you must check and validate the electronic data and accept as input for your applications.

Trading partners can communicate with each other, bypassing intermediaries and inefficient multilayered procedures. Web sites are available to consumer 24 hours a day. Some information-based products, such as software, music, books, and video, can actually be physically distributed via Internet. Internet technology has proved especially useful for supply chain management and collaborative commerce.

Private industrial network. A private industrial network or net marketplace (or e-hubs) is a web-enabled network linking systems of multiple enterprises (firms, companies, organizations etc) for the coordination of enterprises trans-organizational business processes. The network is owned by the buyer and it permits the firm and designated suppliers, distributors and other business partners to share product design and development, marketing, production scheduling, inventory management, and unstructured communication including graphic and e-mail. A net marketplace provides a single digital market place based on Internet technology for many different buyers and sellers.

The e-business – e-commerce relationships

Figure 3.3 suggest the organization system faced to his e-commerce/e-business relationships with the consumers, as a sell side end, and suppliers, as a buy side front:

1) Inside of the company's system boundaries suggested two situations:

- the usage of traditional support software systems grouped in so called back-office systems interfaced to the buy side by specialized software designated to the management of the relationships with suppliers (SCM-Supply Chain management) and to the sell side by the specialized software designated to the management of the relationships with the costumers (CRM-Customer Relationship Management). Bellow is a brief presentation of both SCM and CRM solutions;

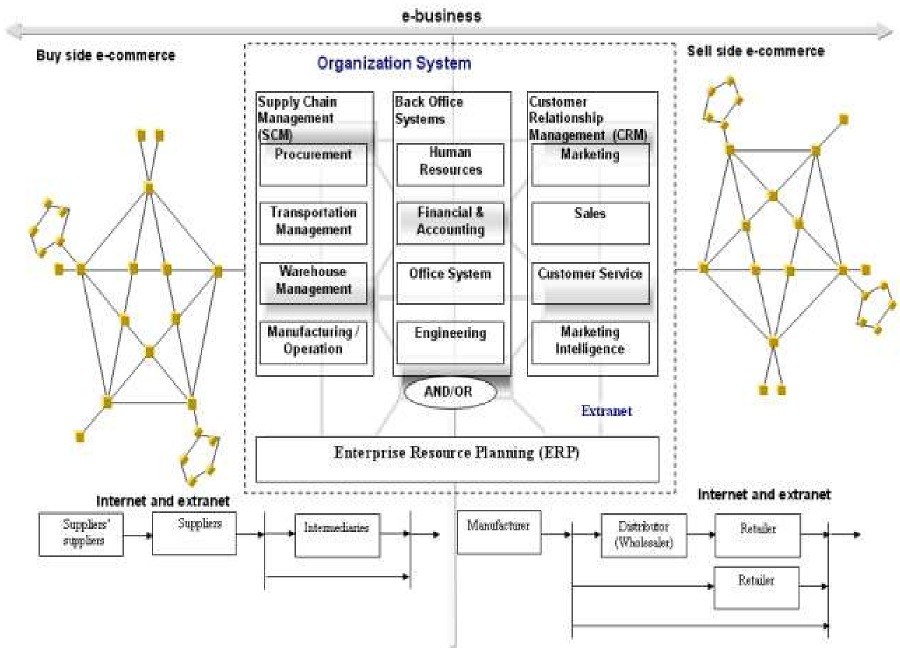


Figure 3.3 The extended company and management of customers and suppliers relationships

- the usage of ERP (Enterprise Resource Planning) systems that provides a single informatic solution from a single software developer with integrated functions for major business functions such as production, distribution, sales, finance & accounting, and human resources management. The implementation of enterprise resource planning (ERP) systems allows enterprises to integrate and optimize their internal operations, such as production, Engineering, financial controlling and human resources. The enterprise resource planning systems are integrating web connections to leverage the speed and ubiquitous nature of the Internet (for example, SAP's R/3 system is Internet compatible and can be combined with other types of software under the enterprise).

Both situations suppose the existence of intranets and extranets support.

2) the buy side e-commerce suggest the e-commerce transactions between purchasing organization and its suppliers. These relationships are mapped across the support offered by both intranet and extranets so that is possible to realize Internet-based supply-chain management allowing all supply-chain participants receive and exchange information on purchasing, production, and shipping at real-time. The organization suppliers can deliver directly their products and/or services or by using intermediaries. In turn the organization suppliers can have relationships with their own suppliers (suppliers' suppliers). The usage of SCM e-commerce is the source of disintermediation, defined as the removal of organizations or business processes layers responsible for certain steps in the value chain;

3) the sell side e-commerce suggests the e-commerce transactions between supplier organization (manufacturer) and its customers. This side includes the direct sell to customers by applying the disintermediation (the companies can obtain higher profit while charging lower prices) or the usage of different categories of intermediaries (Distributors and/or Retailers).

Electronic commerce, global competition, and the rise of digital firms have made companies think strategically about their business processes for managing their relationships with customers and suppliers.

The relationships between e-commerce (EC) and e-business (EB) can be described as illustrated in figure 3.4. Figure 3.5 shows the relationship between e-commerce and e-business by separating the activities specific to each other.

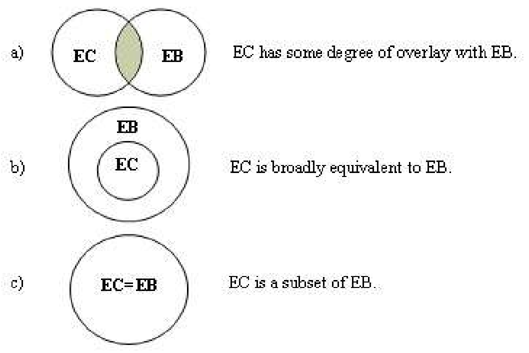

a)
EC
EB
EC has some degree of overlay with EB.
b)
EB
EC
EC is broadly equivalent to EB.
c)
EC=EB
EC is a subset of EB.

Figure 3.4 e-commerce – e-business relationships

(Source of figure [BIS-TDM])

CRM (Customer Relationship Management). Instead of treating customers as exploitable source of income, businesses are now viewing them as long-term assets to be nurtured through customer relationship management (CRM). CRM focuses on managing all of the ways that a firm deals with its existing and potential new customers. CRM is both a business and technology that uses information systems to coordinate all of the business processes surrounding the firm's interactions with its customers in sales, marketing and service.

SCM (Supply Chain Management). Supply chain management is the close linkage and coordination of activities involved in buying, making, and moving a product. It integrates supplier, manufacturer, distributor, and customer logistic processes to reduce time, redundant effort, and inventory costs. The supply chain is

a network of organizations and business processes for procuring materials, transforming raw materials into intermediate and finished products, and distributing the finished products to customers.
It links suppliers, manufacturing plants, distribution centers, conveyances, retail outlets, people, and information through processes such as procurement, inventory control, distribution, and delivery to supply goods and services from source to consumption. Materials, information, and payments flow through the supply chain in both directions. Goods start out as raw materials and move through logistics and

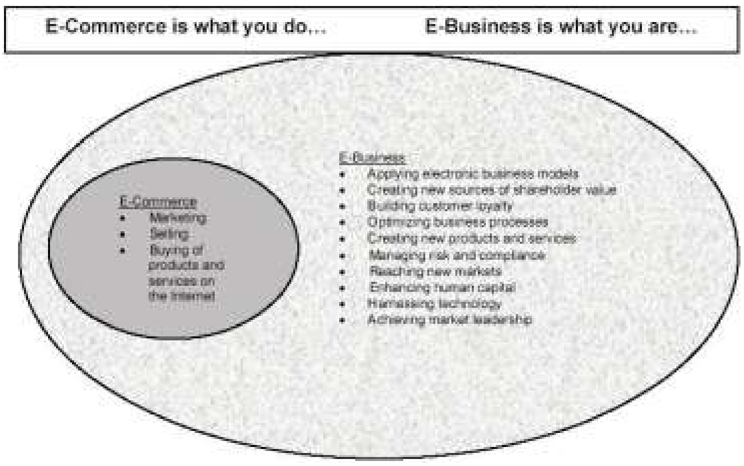

E-Commerce is what you do...
E-Business is what you are...
E-Commerce
Marketing
Selling
Buying of products and services on the Internet
E-Business
Applying electronic business models
Creating new sources of shareholder value
Building customer loyalty
Optimizing business processes
Creating new products and services
Managing risk and compliance
Reaching new markets
Enhancing human capital
Harnessing technology
Achieving market leadership

Figure 3.5 e-commerce is a part of e-business (source [PG-07]**)**

production systems until they reach customers. The supply chain include reverse logistic in which returned items flow in reverse direction from buyer back to the seller. The supply network is a critical component of any e-business strategy. Information sharing has always been the key to coordination. With the advancement of communication technologies, such as intranet, extranet, electronic data interchange (EDI), and virtual private network (VPN), companies have already started to coordinate their purchasing, production, and distribution activities to reduce cycle times and cut operational costs.

Business categories

There are many ways in which electronic commerce transactions can be classified such as [BIS-TDM; KLJL]:
- the nature of participants (figure 3.6): business-to-consumer (B2C), business-to-business (B2B), consumer-to-consumer (C2C);
- the participant's physical connection – mobile commerce (m-commerce).

Business-to-consumer (B2C) electronic commerce involves retailing products and services to individual shoppers;

Business-to-business (B2B) electronic commerce involves sales of goods and services among business. Companies can sell to other business using their own web sites as electronic storefronts or they can execute purchase and sell transactions through private industrial networks or net marketplace. Net marketplace is the faster-growing type of B2B.

Consumer-to-consumer (C2C) electronic commerce involves consumers selling directly to consumers.

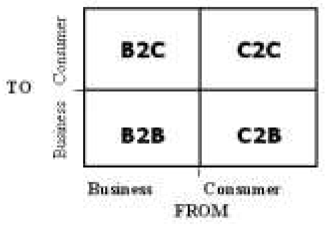


Figure 3.6 Business categories

The electronic business value chain described in figure 3.3 bottom can be described considering the business categories involved between participants, as shown in figure 3.7.

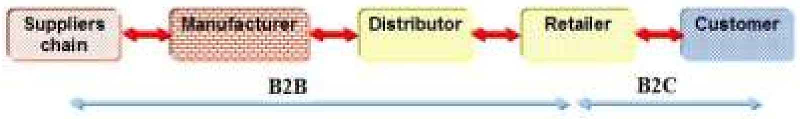


Figure 3. 7 The electronic business value chain

Mobile commerce (m-commerce) involve using of handheld wireless devices for purchasing goods and services. Both B2B and B2C transactions can take place using m-commerce technology.

3.2 Business Models

A **business model** describes how the enterprise delivers a product or service, showing how the enterprise creates wealth. Business models have been defined and categorized in many different ways, the models are implemented in a variety of ways, and they are perhaps the most discussed and least understood aspect of the web.

A definition of business model together with the evolution of business model concept can be given in reference [OPT]: "*A business model is a conceptual tool that contains a big set of elements and their relationships and allows expressing the business logic of a specific firm. It is a description of the value a company offers to one or several segments of customers and of the architecture of the firm and its network of partners for creating, marketing, and delivering this value and relationship capital, to generate profitable and sustainable revenue streams*".

In [PG-07] you can found the following alternative definitions:

"A business model can be defined as the method by which a firm builds and uses its resources to offer its customers better value than its competitors and to make money doing so. It details how a firm makes money now and how it plans to do so in the long run. The model is what enables a firm to have a sustainable competitive advantage, to perform better than its rivals in the long term.”

”An e-business model is a description of the roles and relationships among a firm's consumers, customers, allies, and suppliers that identifies the major flows of product, information, and money, and the major benefits to participants.”

A business model draws on a multitude of business subjects, including economics, entrepreneurship, finance, marketing, operations, and strategy.

Chesbrough and Rosenbloom [CheRo] outlined the following six components of the business model:

1. **Value proposition** - a description of the customer problem, of the product that addresses the problem, and of the value of the product from the customer's perspective;
2. **Market segment** - the target group of customers;
3. **Value chain structure** - the position and activities in the value chain of the firm and how that will capture part of the value that it creates in the chain;
4. **Revenue generation and margins** - how revenue is generated (sales, leasing, subscription, support, etc.), the cost structure, and target profit margins;
5. **Position in value network** - identification of competitors, complementors, and any network effects that can be utilized to deliver more value to the customer.
6. **Competitive strategy** - how the company will attempt to develop a sustainable competitive advantage.

Managing an organization’s e-business adoption strategy has proven to be a daunting task. Strategic decisions with far-reaching implications must be made on a timely basis.

The collapse of NASDAQ’s high-tech (dot-com) stocks during 2000/2001 offers painful proof of the extraordinary challenges associated with managing e-business. Gone are the days of evaluating new venture start-ups based on burn rates, over-inflated revenue estimates and the vita of a silicon-valley cowboy. Indeed, the “irrational exuberance” in dot-com company stock market valuations has come to fruition. The market has forced companies to focus, once again, on the basics: cost, quality and profitability. Lock-step with this *back-to-the-basics* pendulum swing, is the utilization of a business model that is long-term focused, profit-based, and includes the unique challenges (and opportunities) with conducting commerce via the Internet.

The business model should enable the cost, quality and profitability basic necessities, utilizing a long-term profit-based business plan, while simultaneously accommodating the unique business issues associated with e-commerce.

As firms integrate e-business into their existing business, they migrate from traditional physical business models to combined physical and virtual models.

This shift increases the role of the information technology infrastructure because information and online transaction processing become more important.

B2B interactions involve much more complexity [DMVA] than B2C and typical B2B transactions include, among others, the following steps: i) review catalogs, ii) identify specifications, iii) define requirements, iv) post request for proposals (RFP), v) review vendor reputation, vi) select vendor, vii) fill out purchase orders (PO), viii) send PO to vendor, ix) prepare invoice, x) make payment, xi) arrange shipment, and xii) product inspection and reception.

Several models and classifications have been proposed for B2B commerce, as Internet and e-market evolves. Figure 3.8 illustrates an electronic marketplace for B2B trading as defined in [DMVA]. The model could be oriented to a vertical market (e.g., wholesale trade, chemicals, construction, and electronics) or to a horizontal approach (e.g., office supply, and logistics).

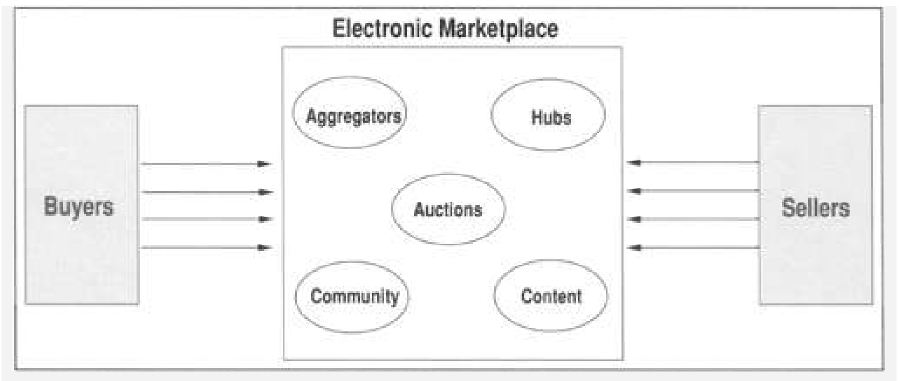


Figure 3. 8 Electronic marketplace for B2B commerce

Model	Description
Aggregators	One company aggregates buyers in order to form a virtual buying entity and/or aggregates suppliers to constitute a virtual distributor. The aggregator takes the responsibility for selection and fulfillment, pricing, and marketing segmentation. For example, in the science marketplace, one company became the central buying location for thousands of buyers to implement their own purchasing rules and obtain volume discounts. Another example is an electronic company that offers a total home buying service, from search to financing, under one site.
Hubs or Process Integration	Focuses on producing a highly integrated value proposition through a managed process. Hubs have been defined as neutral Internet-based intermediaries that focus on a specific industry or a specific business process. Hubs host electronic markets and create value by reducing costs

	of transactions between sellers and buyers. There are examples of vertical hubs that serve a vertical market or a specific industry, such as energy, steel, telecommunications, and plastic. On the contrary, functional hubs specialize in horizontal markets across different industries. Functional hubs focus on business processes such as maintenance, repair and operating, procurement. For instance, an electronic business company that provides office supplies to many industries is a good example of a functional hub in a B2B commerce.
Community or Alliance	In the community model, alliances are used to achieve high value integration without hierarchical control. Members and end users play key roles as contributors and customers. Basically, communities produce knowledge with economic value, such as Linux, MP3, and Open Source.
Content	Content is the end product of this model of B2B commerce. It has the purpose of facilitating trading. Revenue can be generated from subscriptions, membership, or advertising. For example, there are e-companies that sell information about contracts to bid, market intelligence and analysis, and jobs by industry.
Auctions or Dynamic Pricing Markets	Auctions or dynamic pricing markets handle complex exchanges between buyers and sellers in B2B commerce. Auctions are dynamic and efficient mechanisms for mediating and brokering in complex marketplaces, like supply-chain and procurement systems. Bundle auctions allow agents to bid for bundles of items and are useful for B2B applications such as automatic supply-chain or procurement.

3.2.1 Classification of e-business models

There exists many classification criteria of Internet business models and we don't have yet a common point of view.

In determining an appropriate e-business model, several criteria can be used, such as [PG-07]:

- **Involved parties**, such as business-to-business, business-to-consumer, and/or consumer-to-consumer;
- **Revenue sources**, such as transaction fee, product price, and/or exposure fee;

- **Value configuration**, such as value chain, value shop, and/or value network;
- **Integration** with customers and/or partners;
- **Relationships**, such as one-to-many, many-to-many, and/or many-to-one;
- **Knowledge**, such as know-how, know-what, and know-why.

One classification of Internet business models is presented by K. Laudon and J. Laudon ([KLJL]):

- **Virtual storefront:** Sells physical products directly to consumers or to individual businesses (Amazon.com, EPM.com)
- **Information broker:** Provides product, pricing, and availability information to individuals and businesses. Generates revenue from advertising or from directing buyers to sellers (Edmunds.com, Kbb.com, Insweb. com, IndustralMall.com)
- **Transaction broker:** Saves users money and time by processing online sales transactions, generating a fee each time a transaction occurs. Also provides information on rates and terms (etrade.com, Expedia.com)
- **Online marketplace:** Provides a digital environment where buyers and sellers can meet, search for products, display products, and establish prices for those products (eBay.com, Priceline.com, ChemConnect.com, Pantellos.com)
- **Content provider:** Creates revenue by providing digital content, such as digital news, music, photos, or video, over the Web (WSJ.com, CNN.com, TheStreet.com, Gettyimages.com, MP3.com)
- **Online service provider:** Provides online service for individuals and businesses. Generates revenue from subscription or transaction fees, from advertising, or from collecting marketing information from users (@Backup.com, Xdrive.com, Employease.com, Salesforce.com)
- **Virtual community:** Provides online meeting place where people with similar interests can communicate and find useful information (Motocross.com, iVillage.com, Sailnet.com)
- **Portal:** Provides initial point of entry to the Web along with specialized content and other services (Yahoo.com, MSN.com, StarMedia.com)

Another way to classify the Internet business models, proposed by Weill and Vitale ([WV-01]) and reconsidered and updated in [PG-07], identifies a finite number of atomic e-business models that can be combined as building blocks to create tailored e-business modes and initiatives, as follows:

1. **Direct to customer:** buyer (individual or business) and seller (does no matter is a retailer, a wholesaler, or a manufacturer) communicate directly. Examples of the direct-to-customer model are Dell Computer Corporation and Gap, Inc.
2. **Full-service provider:** provides total coverage of customer needs in a particular domain, consolidated via a single point of contact. Examples of the full-service provider are the Prudential Advisor and GE Supply Company.

3. **Whole of enterprise:** The single point of contact for the e-business customer is the essence of the whole-of-enterprise atomic business model. Although many of this model's breakthrough innovations have occurred in public-sector organizations, the model is applicable in both the for-profit and the public sectors.
4. **Intermediaries** such as portals, agents, auctions, aggregators, and other intermediaries. E-business is often promoted as an ideal way for sellers and buyers to interact directly, shortening old-economy value chains by disintermediating some of their members. Examples of intermediaries are electronic malls, shopping agents, specialty auctions, electronic markets, electronic auctions, and portals.
5. **Shared infrastructure:** The firm provides infrastructure shared by its owners. Other suppliers, who are users of the shared infrastructure, but not owners, can also be included. Customers who access the shared infrastructure directly are given a choice of suppliers and value propositions. The owner and the non-owner suppliers are generally represented objectively.
6. **Virtual community:** By using IT to leverage the fundamental human desire for communication with peers, virtual communities can create significant value for their owners as well as for their members. Once established, a virtual community is less susceptible to competition by imitation than any of the other atomic business models. In this business model, the firm of interest—the sponsor of the virtual community—sits in the center, positioned between members of the community and suppliers.
7. **Value net integrator:** control the virtual value chain in their industries by gathering, synthesizing, and distributing information. Value net integrators add value by improving the effectiveness of the value chain by coordinating information. Examples of value net integrators are Seven-Eleven Japan and Cisco Systems.
8. **Content provider:** a firm that creates and provides content (information, products, or services) in digital form to customers via third parties. The physical-world analogy of a content provider is a journalist, recording artist, or stock analyst. Digital products such as software, electronic travel guides, and digital music and video are examples of content. A virtual-world example of a content provider is weather forecasters such as Storm Weather Center.

3.2.2 Common of Internet e-business models

The most common of Internet e-business models are the following:

The Merchant. A merchant is a wholesaler or retailer of goods and services.

The merchant provides a website with product information and an online ordering mechanism. Users select the products they want to buy and place an order. The product price can be fixed or negotiable. The merchant makes his money the same way as traditional "brick-and-mortar" shops: through the profit margin in the product price. This model is mainly suited for physical goods and services, such as books, computers or a pizza delivery service. The merchant can directly reach end users and sell to them without needing wholesalers or retailers.

Click-and-mortar merchants. "Click-and-mortar" shops combine a website with a physical store.

In that way these business have the additional advantage that they (usually) already have an established brand name, and that they can use their physical store to promote the website. Because they have a physical location the users can return unwanted or defective products simply by going to the store, rather than mailing it to a web site operator. Traditional mail-order businesses (catalog merchants) already have the necessary facilities to process orders over the Internet (basically, orders come in by e-mail rather than by letter or phone, the shipping and handling is the same).

Build to order merchants. Offering of goods and services for sale that can be customized as client suggest.

A manufacturer (such as a computer vendor, for example) can use this model by offering his goods or services for sale and having the ability to order customized versions. The customized product is then assembled individually and shipped to the customer. This provides added value to consumers and allows the manufacturer to create only those products that will be sold.

The service provider. Offering services via Internet.

For some services, the merchant model is quite appropriate. For example, a pizza delivery service can operate on a pay-per-item basis. However, many Internet-based services cannot easily be handled in this way. It is often difficult to define the "product" that is sold, or to set a price for this product. For instance, a news site can offer the service of access to its archive, but even one dollar is probably too much for retrieving one article. Some service providers provide advertising-based access to their service, hoping to recover the costs through revenue from the advertisers. However, this appears to be a doomed strategy, since few ad-driven sites are able to get sufficient income (Yahoo! being one of the very few successful ones).

Subscription-based access. Allow client access their services, in a gradual manner, based on client subscription amount.

Many service operators provide subscription-based access to their service. A user pays a fixed amount per month or year and in return gets unlimited actions to the service. Alternatively, a base fee can be paid per month and all access beyond a certain limit is subject to a surcharge. This model is typically used when accessing databases with articles, news, and patents but also for online games or adult websites. However, the viability of subscription-based models is doubtful (a 2000 survey by Jupiter Communications found that almost half of all Internet users would not pay to view content on the web). To entice users into subscribing, "teasers" or selective portions may be made available for free. For example, showing headlines for articles in a news archive or allowing access to patent documents one page at a time.

Prepaid access. The offered services require payment by units (minute, data transfer amount, etc).

Some services, in particular telephony, require payment by the minute. This can be handled via a subscription, but a viable alternative is prepaid access. In this scheme, users pay a certain amount of money, which gives them access to the service for a certain amount of time, or access to a certain amount of content. When the amount is spent, the user can prepay another amount for further access. Often, implementations involve a smartcard on which the available credit is stored. Payment is realized by buying such a smartcard. The available credit on the smartcard is reduced during usage of the service. Prepay schemes have the advantages that they do not require subscription details to be maintained, and that they give users greater control over how much to spend on the service.

The broker. Brokers or intermediaries create markets by bringing buyers and sellers together and facilitating transactions between them. Those can be business-to-business (B2B), business-to-consumer (B2C), or consumer-to-consumer (C2C) markets.

A broker makes money by charging a fee for every facilitated transaction, for instance as a percentage of the price of the transaction. Some special types of brokers are:

- **Group buying** - bringing individual potential buyers together in order to buy as a group, which should result in a lower price for each buyer (volume discounts, etc).
- **Classified ads** - sellers can advertise their product on a site where buyers can find it. The broker makes money in various ways: the seller pays a placement fee, or the broker receives a portion of the price paid by the buyer.
- **Bounties** - the broker offers a reward for finding a person, thing, idea, or other desired, but hard to find item. The broker may list items for a flat fee, or charge a percentage of the reward if the item is successfully found.

The broker is also used in the architecture of offered web services as described in the chapter 2, §2.4.

The sales representative. Business based on a commission.

Sales representatives often work on a commission basis: they sell an item for someone else and get a percentage of the price. On the Web, this model has developed into what is known as affiliate programs or referral fees. Someone creates a website on a particular topic and adds links to products on a merchant site which are related to his topic, so his visitors can buy those. For example, a music reviewer can add a link in a review to an online music store where the CD being reviewed can be ordered. If the reader likes the review, he can following the link and buy the CD. The merchant then pays the reviewer a commission or referral fee to the reviewer for referring to his site.

This model is realized as follows:

- The reviewer registers at the merchant site and receives a unique code;
- He adds this code to all the links to the merchant site whenever he links to a product at that site;
- When a reader follows the link, the merchant site sees the code and couples the reader's actions to the code;
- When the reader buys something the site registers the sale together with the code. Later all sales matching that code can be collected so the percentage can be computed and credit to the reviewer.

There are many variations on this theme. A fixed percentage can be paid to all sales resulting from the referral or a high percentage can be given to the actual product to which he linked, possibly with a low percentage on other sales that resulted from the link. This model is used by Amazon, Proxis, CD-Now and others. Some book authors link to their own book this way, making more money to the referral fee than to the royalties they get in. Subscription-based services also sometimes offer a referral fee to anyone who brings in a new subscriber. It is easy and safe to participate in an affiliate model, even for individuals. Anyone who can set up a website can link to a product, and if things go well, make money. If not, then no effort or investment is wasted. This explains the popularity of the model on the World-Wide Web.

The advertiser. Advertising-driven sites.

Advertising-driven sites are currently one of the cornerstones of E-commerce. The principle is simple and well known: the site offers free access to something and shows advertisements on every page and when a user clicks on an advertisement, he goes to an advertiser's page. The advertiser pays the site operator for showing his advertisement (eyeballs) or for every time someone clicks on the advertisement (click-through). The same idea is popular in computer programs. Users can download and use the program for free but advertisements are shown during operation or startup of the program. In particular, advertisements can be shown when the user needs to wait for some time-consuming operation, such as printing or scanning.

Targeted advertising. Sites realizing targeted advertising.

It is well known that an advertisement related to the topic at hand on the site will get higher exposure and click through since such advertisements are targeted to the site visitors. So, the site operator earns more money if he places targeted advertisements. When displaying advertisements in a computer program, it is possible to target the advertisements to the purpose of the program, e.g., a spreadsheet shows advertisements for a stock brokering service. Racing games, soccer games and the likes commonly show billboards in the game to emulate the look of the real playing field. The advertisements thereon can be chosen as "real" advertisements.

Search engines use this idea as well, but relate the advertisements to the keywords entered in a query. For instance, if someone searches information on holidays, an advertisement is shown for a hotel chain on the page with search results. The advertisement can further be targeted based on the user profile for the user doing the search (e.g., if the profile shows the user likes to swim, an advertisement is shown for a beach hotel).

The existence of advertising-driven sites created a business opportunity for companies such as DoubleClick, which collects advertisements from many sources and arranges for placements on different sites. The sources pay DoubleClick for placing their advertisements, and this revenue is then shared with the site owners. Additionally, DoubleClick tracks the users that view all the advertisements, which allows it to built a user profile. This profile can then be used to more accurately target advertisements to these users.

Updating advertisements. Scheduled advertising for off/on line presentation.

It is desirable to be able to present the user fresh advertisements periodically, even when he is not connected to the network. To this end, his browser or other client can download multiple advertisements simultaneously and display them one at a time when he is offline. A screensaver can also be used to present advertisements when the system is idle. The screensaver periodically downloads new advertisements and/or news messages, and presents them to the user.

Portal sites. Sites which provide the main method of access to other web sites.

A portal offers one-stop access to different content and services, such as searching, news, e-mail, stock information, message boards or chat. By offering the option to personalize the interface and presented content (see, for example, my.cnn.com or my.yahoo.com), the portal is made more attractive to the user. The portal site can target its advertisements based on the personalization information. Examples of portals include Yahoo (www.yahoo.com), MSN (www.msn.net), Netscape Netcenter (home.netscape.com), IBM (www.ibm.com), ASE (www.ase.ro) etc.

Attention/incentive marketing.

In this model, a user downloads and views many advertisements and clicks on them, which generates revenue for the intermediary which provided these advertisements to the users. This revenue is then shared with the users in proportion to the number of advertisements they viewed and clicked on. Often, the user is asked to enter demographic information, which the intermediary shares with the advertisers.

A difficult problem in this area is how to guard against fraud. A user could employ a computer program that automatically clicks on all advertisements sent by the intermediary. This way, he collects a large amount of money without actively seeing the advertisements. Thus, it is recommended to measure the time between showing the advertisement and the user's reaction. If that time is too short, or the same every time, it is likely that something is amiss.

If the advertisement is in the form of a video or audio fragment, the user could also be asked to press a particular button or answer a question at some point during the advertisement.

Another solution involves the use of a smartcard. The user must insert a smartcard in a television system or the like, and the reward (usually in the form of credits, although digital cash can also be used) is recorded on the smartcard. When the advertisement has been shown, the card is ejected, so that the user must re-insert the card for the next advertisement.

Free access. Users are given something for free, but the something comes with advertisements.

A few examples: free web space providers typically provide advertising banners at the top or bottom of its users' sites (or as a separate, pop-up window). Free Internet access providers show advertisements on the starting page its users see when they go online. Electronic greeting cards are sent with a personal message and an advertisement. Since the user base is very diverse, it is hard to accurately target advertisements, making the expected revenue low.

The auction room. In an auction, the price of a product is made dependent on what buyers are willing to pay.

There are a number of models for performing an auction, the two most well-known being the "open" auction and the "reverse" auction.

Open auctions. In the "open" auction, participants repeatedly place higher bids for a product under auction. The person who places the highest bid is awarded the product. Networks such as the Internet make it possible for a large number of bidders to participate simultaneously in one auction. Handling bids can even be automated, so that no human auctioneer is necessary.

> Famous auction site eBay offers the option to participate in an auction automatically. The bidder enters an initial bid, an amount with which to increase the bid and a maximum amount. The system then automatically raises the bid with the indicated amount whenever someone else places a higher bid, until either the bidder has won the auction or his maximum is reached.

Reverse auctions. In a "reverse" or "Dutch" auction, the price is initially set at a very high level, and drops at regular intervals. Participants can pick the price at which they want to buy, and have to determine the chance that someone else will find a higher price acceptable.

> In a variant of the reverse auction, customers indicate a product or service and a price, which they are willing to pay. Suppliers indicate a price at which they are willing to provide that product or service, and the auction service tries to match customers and suppliers. The intermediary pockets the difference between the price paid by the customer and the price paid to the supplier. This model is popular with high-priced items like automobiles or airline tickets.

The virtual mall. A virtual mall is a site that hosts many merchants, service providers, brokers and other businesses. The virtual mall operator typically charges a fee for setting up and maintaining the merchant's "booth", and for including him in the site-wide catalog. Additionally, he may charge a fee for every transaction the merchant performs. Virtual malls can operate within the context of a larger site, such as a portal.

> The virtual mall can act as an intermediary between individual customers and the business it hosts, for instance by facilitating payment and guaranteeing a full refund if a merchant does not deliver in time.
>
> When the virtual mall offers services such as payment facilitation or catalog browsing, it has the ability to create aggregated user profiles on the customers that visit any of the businesses in the mall. This can lead to the development of highly specialized malls (e.g., oriented at kids or sports lovers).

The virtual community. A virtual community is a website which has gathered a group of users with a common interest who work together on the site. Typically, users will share information and make contributions in other ways. Since they have contributed to it themselves, users feel highly loyal to the site and will visit it regularly. This offers possibilities for advertising.

> Probably the largest virtual community can be found on Slashdot, a Linux-oriented site on which users share interesting news articles and websites (which invariably fail under the load of hundreds of thousands of people visiting it shortly after its URL got posted on Slashdot - this is called "slashdotting").
>
> A specialized type of virtual community is the knowledge network or expert site, where people, layman and expert, share their expertise and experiences. These sites are typically ran like a forum where participants can get questions answered or raise topics for discussion. Long-time participants often meet together in real life. Usenet newsgroups are a good example of such a community.
>
> When a knowledge network is devoted to a particular product or company, the active participation of employees of that company is often very much appreciated and can offer a great PR opportunity for the company.

A simple way to monitor a virtual community is to require registration for access to the website, preferably for free. This allows inter-session tracking of users' site usage patterns and thereby generates data of greater potential value in targeted advertising campaigns. Registration can be made more attractive by offering limited access or "teasers" to unregistered users, by offering the option to customize the site after registration, or by allowing only registered users to actively participate in chat or message boards.

The infomediary (information intermediary). An infomediary collects, analyzes and sells information on consumers and their buying behavior to other parties who want to reach those consumers.

Typically, the infomediary offers the consumers something for free, such as free hardware or free Internet access. The later is especially useful, since it allows the infomediary to control and monitor the user's online activities. After all, the consumer connects through the infomediary's network. The information which the infomediary collects is extremely valuable for marketing purposes. Often the infomediary makes money with an advertising-based model, in which the advertisements are targeted based on the information it collected itself.

The infomediary needs to keep track of its users. A simple way to achieve this is to require registration for access to the website, preferably for free. This allows inter-session tracking of users' site usage patterns and thereby generates data of greater potential value in targeted advertising campaigns. Registration can be made more attractive by offering limited access or "teasers" to unregistered users, by offering the option to customize the site after registration, or by allowing only registered users to actively participate in chat or message boards.

The infomediary model is useful in combination with a virtual community model or virtual mall, since those models offer the ability to collect the necessary information.

3.3 The E-Commerce Development And Functional Architecture

The e-commerce/e-business development

E-commerce/e-business is creating tremendous impact on our economy and its subsequent economic rules. The volume of e-commerce as a percentage of the nation’s GNI (GBP) grows at an increasing rate. Table 3.1 shows the evolution of e-commerce between 2000 (marked by the beginning of the crash of so called “dot-coms”) and 2004.

The nature of e-commerce/e-business is getting more and more complex as the market evolves (see figure 3.9). The elements outlined in the figure axis have been introduced in chapter 1 and will be extended also in chapter 3. Briefly, the first generation – e-Commerce emerged as companies rushed to set up their homepages to claim their web appearances, and the second generation e-business is characterized by the emergence of “mission critical, industrial strength platforms” that support new markets and new models.

It is now widely understood that a successful e-business is built on a business model with a valid value proposition, a clearly defined e-business strategy, and an integrated information technology (IT) infrastructure that facilitates the strategy.

The venue of conducting e-business has also been greatly expanded, especially with the growth of business-to-business (B2B) e-commerce. From implementing individual web-based applications to transforming traditional businesses into click-and-mortar, enterprises are continuously exploring new opportunities and new markets for e-business.

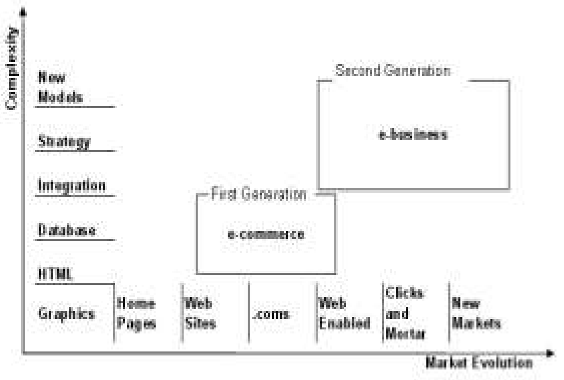


Figure 3.9 The Market Evolution and Complexity of e-Commerce / e-Business Development

Source: e-Business Management Models: A Services Perspective and Case Studies, Revere Group, *Todd Miller, Matthew L. Nelson, Stella Ying Shen and Michael J. Shaw*

The evolution of e-commerce

Table 3.1 shows the evolution of e-commerce between 2000 (marked by the beginning of the crash of so called "dot-coms") and 2004. The world e-commerce turn-over amount was 6.8 trillion US dollars in 2004 and estimated for 2005 to 8.5 trillion US dollars. According to previous table data USA realizes 47% from total world e-commerce turn-over amount followed by Japan 13%, and Germany 5.7%.

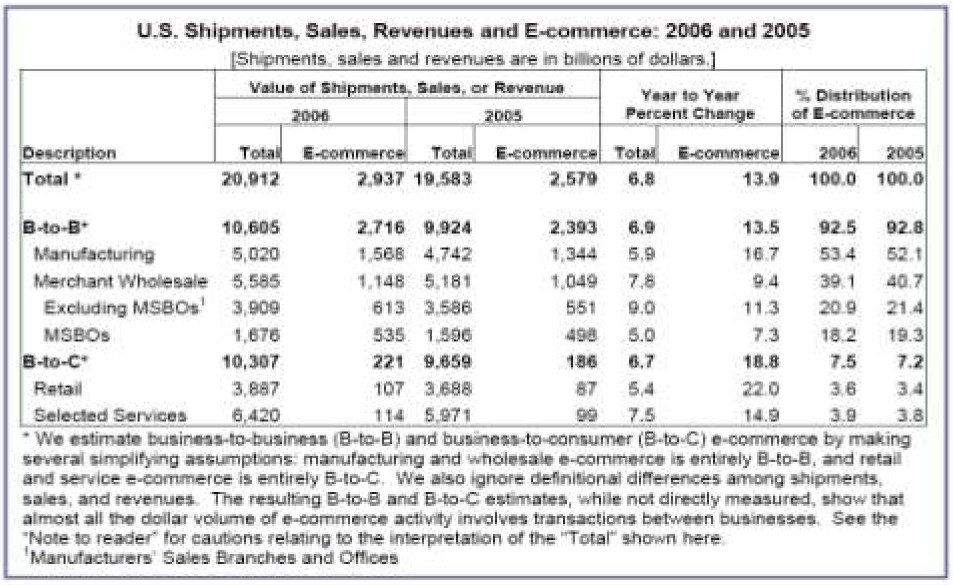
Table 3.1 The evolution of e-commerce (billion $)

	2000	2001	2002	2003	2004	% of total 2004
Total ($-billions) from which:	$657.0	$1,233.6	$2,231.2	$3,979.7	$6,789.8	8.6%
North America	$509.3	$908.6	$1,498.2	$2,339.0	$3,456.4	12.8%
USA	$488.7	$864.1	$1,411.3	$2,187.2	$3,189.0	13.3%
Canada	$17.4	$38.0	$68.0	$109.6	$160.3	9.2%
Mexico	$3.2	$6.6	$15.9	$42.3	$107.0	8.4%
Asia Pacific	$53.7	$117.2	$286.6	$724.2	$1,649.8	8.0%
Japan	$31.9	$64.4	$146.8	$363.6	$880.3	8.4%
Australia	$5.6	$14.0	$36.9	$96.7	$207.6	16.4%
Korean	$5.6	$14.1	$39.3	$100.5	$205.7	16.4%
West Europe	$87.4	$194.8	$422.1	$853.3	$1,533.2	6.0%
Germany	$20.6	$46.4	$102.0	$211.1	$386.5	6.5%
England	$17.2	$38.5	$83.2	$165.6	$288.8	7.1%
France	$9.9	$22.1	$49.1	$104.8	$206.4	5.0%
Italy	$7.2	$15.6	$33.8	$71.4	$142.4	4.3%
Holland	$6.5	$14.4	$30.7	$59.5	$98.3	9.2%
Latin America	$3.6	$6.8	$13.7	$31.8	$81.8	2.4%

Source: Forrester Research, Inc.

The amount of turn-over by regions, in percents is:

Region	**Percent**
North America	50.9%
Asia/Pacific	24.3%
Europe	22.6%
Latin America	1.2%



U.S. Shipments, Sales, Revenues and E-commerce: 2006 and 2005

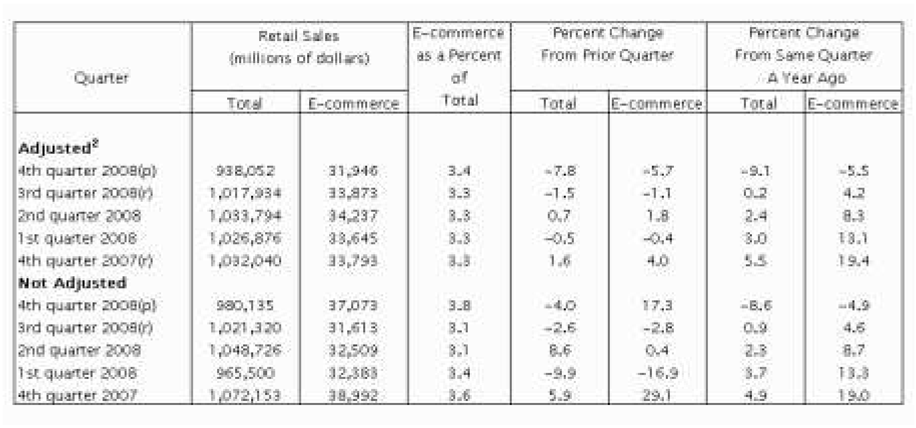
[Shipments, sales and revenues are in billions of dollars.]

	Value of Shipments, Sales, or Revenue				Year to Year Percent Change		% Distribution of E-commerce	
	2006		2005					
Description	Total	E-commerce	Total	E-commerce	Total	E-commerce	2006	2005
Total *	**20,912**	**2,937**	**19,583**	**2,579**	**6.8**	**13.9**	**100.0**	**100.0**
B-to-B⁺	**10,605**	**2,716**	**9,924**	**2,393**	**6.9**	**13.5**	**92.5**	**92.8**
Manufacturing	5,020	1,568	4,742	1,344	5.9	16.7	53.4	52.1
Merchant Wholesale	5,585	1,148	5,181	1,049	7.8	9.4	39.1	40.7
Excluding MSBOs[1]	3,909	613	3,586	551	9.0	11.3	20.9	21.4
MSBOs	1,676	535	1,596	498	5.0	7.3	18.2	19.3
B-to-C⁺	**10,307**	**221**	**9,659**	**186**	**6.7**	**18.8**	**7.5**	**7.2**
Retail	3,887	107	3,688	87	5.4	22.0	3.6	3.4
Selected Services	6,420	114	5,971	99	7.5	14.9	3.9	3.8

* We estimate business-to-business (B-to-B) and business-to-consumer (B-to-C) e-commerce by making several simplifying assumptions: manufacturing and wholesale e-commerce is entirely B-to-B, and retail and service e-commerce is entirely B-to-C. We also ignore definitional differences among shipments, sales, and revenues. The resulting B-to-B and B-to-C estimates, while not directly measured, show that almost all the dollar volume of e-commerce activity involves transactions between businesses. See the "Note to reader" for cautions relating to the interpretation of the "Total" shown here.

[1]Manufacturers' Sales Branches and Offices

Figure 3. 10 The evolution of e-commerce in US (Source: US Census Bureau http://www.census.gov/**)**



Quarter	Retail Sales (millions of dollars)		E-commerce as a Percent of Total	Percent Change From Prior Quarter		Percent Change From Same Quarter A Year Ago	
	Total	E-commerce		Total	E-commerce	Total	E-commerce
Adjusted[2]							
4th quarter 2008(p)	938,052	31,946	3.4	-7.8	-5.7	-9.1	-5.5
3rd quarter 2008(r)	1,017,934	33,873	3.3	-1.5	-1.1	0.2	4.2
2nd quarter 2008	1,033,794	34,237	3.3	0.7	1.8	2.4	8.3
1st quarter 2008	1,026,876	33,645	3.3	-0.5	-0.4	3.0	13.1
4th quarter 2007(r)	1,032,040	33,793	3.3	1.6	4.0	5.5	19.4
Not Adjusted							
4th quarter 2008(p)	980,135	37,073	3.8	-4.0	17.3	-8.6	-4.9
3rd quarter 2008(r)	1,021,320	31,613	3.1	-2.6	-2.8	0.9	4.6
2nd quarter 2008	1,048,726	32,509	3.1	8.6	0.4	2.3	8.7
1st quarter 2008	965,500	32,383	3.4	-9.9	-16.9	3.7	13.3
4th quarter 2007	1,072,153	38,992	3.6	5.9	29.1	4.9	19.0

Figure 3. 11 Estimated Quarterly U.S. Retail Sales: Total and E-commerce (Source: US Census Bureau http://www.census.gov/**)**

The functional architecture for e-commerce

Systems for Internet commerce have many masters. For analysis of architecture we consider four primary components of Internet commerce system (figure 3.12): customer, seller, transaction system, and payment gateway. For each one we present some security considerations.

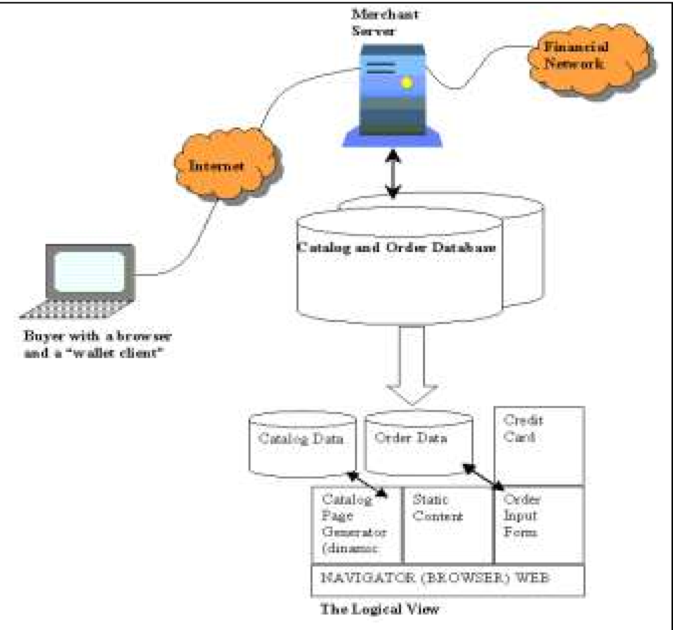


Figure 3.12 The functional architecture for e-commerce

1) **Customers** (Buyer, Clients) – The client is a computer system, typically a PC, connected directly to Internet via an ISP (Internet Service Provider), or indirectly via a corporate network. The primary tool for using www is a browser (a Web client). It is possible also to access www via specialized applications designed for e-commerce (particular for payments) called wallets. The buyer can be represented by:
   - **Retail customer** – the buyer that use the system for business-to-consumer commerce. This category of customers would like to retain their privacy, releasing as little information as possible to sites on the Net. Generally due to commercial interests this information is combined with other sources of data to build up a very detailed picture of the costumer. A major interest of this category of customers refers to the security. They want to be assured that their credit card numbers and other sensitive information are adequately protected;
   - **Business customers** – the buyers that use Internet commerce systems in the course of their daily jobs (i.e. an administrator reordering office suppliers). For this customers the security required refers to keep their competitors from finding out what they are doing and assuring the integrity of business records in company computer systems;
2) **Seller** (Merchant, Vendor) – The computer system or systems containing the merchant's electronic catalog or products. Sellers include merchants engaged in business-to-business or business-to-consumer commerce or publishers and

content providers engaged in information commerce. The seller's are extremely interested in the integrity of their marketing presence, their prices, their customer records, and their business records;

3) **Transaction system** – the computer system or systems that process a particular order and which are responsible for payment, record keeping, and other business aspects of transaction. The part for credit card processing system is operated by financial processors that accept transactions from merchants and forward them to the merchant's bank. Transaction security is a paramount for a financial processor and includes the privacy and accuracy of records, and the authenticity and integrity of requests;
4) **Payment gateway** – the computer system or systems that routes payments instruments into existing financial networks such as for credit card authorization and settlement.

The heart of every e-commerce application is its database containing generally the catalog, the buying transactions and the related payments transactions. That heart is the most attractive prize for crackers because generally it stores all your customers' information, possibly even their payment information. The simplest way to assure the protection is to permit access to that database only to authorized users granted to realize specific operations. The access realized on the basis of a username and password, generally from server-side scripts, by using connections strings containing, among other parameters, the following argument types: server name, user name, and password. To protect this vital information follows that rules:

- create a general user to access the database (not from administrators group) having insert, update, and select privileges and use these to define the connection string required to access and manipulate the database records;
- store the connection string in a separate script that will be included as a file when needed;
- encrypt all stored passwords.

For assuring secure electronic transactions Visa and MasterCard joined together (in 1995) to develop the Secure Electronic Transaction (SET) protocol, a technical standard for safeguarding payment card purchases made over open networks. SET is designed to mimic the traditional card transaction flow and in addition it includes the use of public key certificates to authenticate the parties to each other. Figure 3.8 illustrates the changes in the main architecture for e-commerce with SET and table 3.2 shows SET goals and requirements for different category of participants.

Table 3.2 SET goals and requirements

Category	Goals	Requirements
CardHolder	Provide confidentiality of information Authenticate merchant to cardholder Improve perception of safety of	Obtain and install cardholder software (wallet) Obtain SET client certificate

	electronic commerce	
Banks	Reduce merchant fraud Build electronic commerce volume	Implement certificate hierarchy Implement certificate systems for cardholders
Merchants	Easy integration Build electronic commerce volume Reduce transaction costs	Implement SET merchant software

By his nature the HTTP protocol do not ensure any protection for the text information sended or received. There's nothing to stop anybody out there from listening and recording your details. Fortunately, we have other methods that can ensure transactions are secure and that the credit card details and other confidential information are not compromised [HKSU]:

- **Encryption:** the message must be *encoded* before sent to the web server and received back from the web server. The web server has a public key, and users will have a private key that enables them to decode the information. Only having the public key and the private key together will allow you to encrypt the message. The web server will have a public key and its own private key at the other end. To encrypt messages, you use a secure communications protocol. Either Secure Sockets Layer (SSL) or Secure HTTP (HTTPS) would provide this functionality. In Windows environmeent you can specify encryption methods and whether to use SSL on a connection in the Web.config file.
- **Certificates:** To guarantee that the site you are dealing with at the other end is reputable, it can be certified by a Certificate Authority. Verisign (www.verisign.com) is perhaps the most common Certificate Authority. The authority is paid a yearly fee by the e-commerce vendor and in return, the authority performs checks on the business to prove that it is legitimate. These checks are then recorded in the form of a certificate. You can browse particular sites' certificates during the checkout process. To make your site trustworthy, you should go about obtaining a certificate from a Certificate Authority.

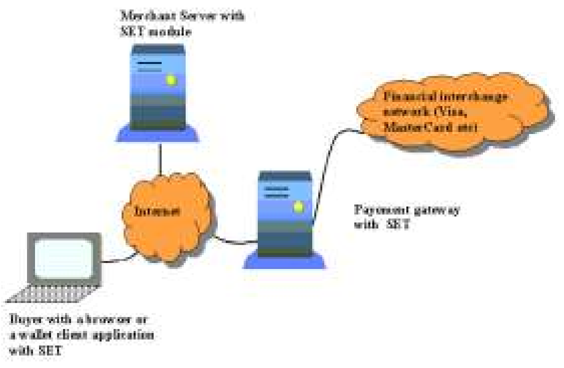

Merchant Server with SET module
Financial interchange network (Visa, MasterCard etc)
Internet
Payment gateway with SET
Buyer with a browser or a wallet client application with SET

Figure 3.13 The functional architecture for e-commerce with SET

The functional architecture with SET changes as depicted in figure 3.13.

Internet vulnerabilities and security

The modern organizations (and even individual) are more and more dependent on information technology in commercial (or personal) or governmental operations. Today, in order to be competitive, we must receive, process, and send information as soon and safety as possible to all partners.

If information is recorded electronically and is available on networked computers, it is more vulnerable than if the same information is printed on paper and locked in a file cabinet. Intruders do not need to enter an office or home, and may not even be in the same country. They can steal or tamper with information without touching a piece of paper or a photocopier. They can create new electronic files, run their own programs, and hide evidence of their unauthorized activity.

Because of the inherent openness of the Internet and the original design of the protocols, (earlier designed without security in mind) Internet attacks in general are quick, easy, inexpensive, and may be hard to detect or trace. An attacker does not have to be physically present to carry out the attack. In fact, many attacks can be launched readily from anywhere in the world - and the location of the attacker can easily be hidden. Nor is it always necessary to "break in" to a site (gain privileges on it) to compromise confidentiality, integrity, or availability of its information or service.

A **vulnerability** is a weakness that a person can exploit to accomplish something that is not authorized or intended as legitimate use of a network or system. When a vulnerability is exploited to compromise the security of systems or information on those systems, the result is a security incident. Vulnerabilities may be caused by engineering or design errors, or faulty implementation. The technical causes behind successful intrusion techniques are represented by the following (but not only) major technical vulnerabilities:

- flaws in software or protocol designs;
- weaknesses in how protocols and software are implemented;
- weaknesses in system and network configurations.

The basic security concepts important to information on the Internet are confidentiality, integrity, and availability. Table 3.3 shortly describes these concepts.

Table 3.3 The concepts of confidentiality, integrity, and availability

Concept	**Description**
Confidentiality	When information is read or copied by someone not authorized to do so, the result is known as *loss of confidentiality.* For some types of information, confidentiality is a very important attribute. Examples include research data, medical and insurance records, new product specifications, and corporate investment strategies.
Integrity	Information can be corrupted when it is available on an insecure network. When information is modified in unexpected ways, the result is known as *loss of integrity.* This means that unauthorized changes are made to information, whether by human error or

	intentional tampering. Integrity is particularly important for critical safety and financial data used for activities such as electronic funds transfers, air traffic control, and financial accounting.
Availability	Information can be erased or become inaccessible, resulting in *loss of availability.* This means that people who are authorized to get information cannot get what they need. Availability is often the most important attribute in service-oriented businesses that depend on information (e.g., airline schedules and online inventory systems). Availability of the network itself is important to anyone whose business or education relies on a network connection. When a user cannot get access to the network or specific services provided on the network, they experience a *denial of service* (DoS).

Concepts relating to the people who use that information are authentication, authorization, and nonrepudiation. Table 3.4 shortly describes these concepts.

Table 3.3 The concepts of authentication, authorization, and nonrepudiation

Concept	Description
Authentication	*Authentication* is proving that a user is whom he or she claims to be. That proof may involve something the user knows (such as a password), something the user has (such as a "smartcard"), or something about the user that proves the person's identity (such as a fingerprint).
Authorization	*Authorization* is the act of determining whether a particular user (or computer system) has the right to carry out a certain activity, such as reading a file or running a program.
Nonrepudiation	Users must be authenticated before carrying out the activity they are authorized to perform. Security is strong when the means of authentication cannot later be refuted - the user cannot later deny that he or she performed the activity. This is known as *nonrepudiation.*

A *network security incident* is any network-related activity with negative security implications. This usually means that the activity violates an explicit or implicit security policy (see the section on security policy). Incidents come in all shapes and sizes. They can come from anywhere on the Internet, although some attacks must be launched from specific systems or networks and some require access to special accounts. A typical attack pattern consists of gaining access to a user's account, gaining privileged access, and using the victim's system as a launch platform for attacks on other sites. The following table shows the category of incidents and a brief description of each:

Category	Description
Probe	A probe is characterized by unusual attempts to gain

	access to a system or to discover information about the system.
Scan	A scan is simply a large number of probes done using an automated tool.
Account Compromise	An account compromise is the unauthorized use of a computer account by someone other than the account owner, without involving system-level or root-level privileges (privileges a system administrator or network manager has).
Root Compromise	A root compromise is similar to an account compromise, except that the account that has been compromised has special privileges on the system.
Packet Sniffer	A packet sniffer is a program that captures data from information packets as they travel over the network.
Denial of Service	The goal of denial-of-service attacks is not to gain unauthorized access to machines or data, but to prevent legitimate users of a service from using it.
Exploitation of Trust	Computers on networks often have trust relationships with one another and attackers can forge their identity, appearing to be using the trusted computer, they may be able to gain unauthorized access to other computers.
Malicious Code	Malicious code is a general term for programs that, when executed, would cause undesired results on a system. Users of the system usually are not aware of the program until they discover the damage. Malicious code includes: Trojan horses, viruses, and worms. Trojan horses and viruses are usually hidden in legitimate programs or files that attackers have altered to do more than what is expected. Worms are self-replicating programs that spread with no human intervention after they are started. Generally they used as transport vector for viruses. Viruses are also self-replicating programs, but usually require some action on the part of the user to spread inadvertently to other programs or systems. These sorts of programs can lead to serious data loss, downtime, denial of service, and other types of security incidents.
Internet Infrastructure Attacks	These rare but serious attacks involve key components of the Internet infrastructure rather than specific systems on the Internet.

In the face of the vulnerabilities and incident trends, a robust defense requires a flexible strategy that allows adaptation to the changing environment, well-defined policies and procedures, the use of robust tools, and constant vigilance.

A *security policy* is a documented high-level plan for organization-wide computer and *information security*. It provides a framework for making specific decisions, such as which defense mechanisms to use and how to configure services, and is the basis for developing secure programming guidelines and procedures for users and system administrators to follow. Because a security policy is a long-term document, the contents avoid technology-specific issues.

A security policy covers the following (among other topics appropriate to the organization):

- high-level description of the technical environment of the site, the legal environment (governing laws), the authority of the policy, and the basic philosophy to be used when interpreting the policy;
- risk analysis that identifies the site's assets, the threats that exist against those assets, and the costs of asset loss;
- guidelines for system administrators on how to manage systems;
- definition of acceptable use for users;
- guidelines for reacting to a site compromise.

Factors that contribute to the success of a security policy include management commitment, technological support for enforcing the policy, effective dissemination of the policy, and the security awareness of all users. Management assigns responsibility for security, provides training for security personnel, and allocates funds to security. Technological support for the security policy moves some responsibility for enforcement from individuals to technology. The result is an automatic and consistent enforcement of policies, such as those for access and authentication. Technical options that support policy include (but are not limited to);

- challenge/response systems for authentication;
- auditing systems for accountability and event reconstruction;
- encryption systems for the confidential storage and transmission of data;
- network tools such as firewalls and proxy servers.

The commonly recommended practices for improving security are represented by the following:

- all accounts must have a password and the passwords are difficult to guess (maybe, a one-time password system is preferable to other);
- the cryptographic techniques must be used to ensure the integrity of system software on a regular basis;
- apply secure programming techniques when writing software;
- must be vigilant in network use and configuration and all necessary changes must be realized as vulnerabilities become known;
- apply the latest available fixes and keep systems current with upgrades and patches as vendors deliver them;
- regularly check on-line security archives for security alerts and technical advice;
- audit systems and networks, and regularly check logs.

4 DOCUMENTS AND WEB SITES – STRUCTURE, DESCRIPTION LANGUAGES

4.1 Web pages and Web sites

The documents for World Wide Web (www) are known as Web pages and they are stored on an Internet server and displayed by a Web browser on your computer. Web browsers display Web pages by interpreting the special HiperText Markup Language (HTM or HTML) tags which are used to encode Web pages with display information. The formatting of Web pages is controlled by a collection of markup codes called HTML tags that marks off parts of Web page to display in certain style.

Web pages usually are linked to many different files, such as graphic and multimedia files. Typically these files are stored locally in a folder or set of folders on the computer's disk drive where constructed the Web site (this folder is known as local web site). When the site published his files and folders are stored, typically, to the hard drive of the Web server that hosts the site. The mechanism used to create access path between documents is called hypertext. A hypertext document is made up of links to other documents, combined together with its displayed content. When a user clicks on a linked file, such as a piece of text, a graphic, or a portion of a graphic, their browser display the file that the link points to. Links embedded with text are easily identifiable. Most browsers default to coloring and underlining linked text, and user can set the color and underline option as they prefer.

A Web site is defined as a collection of files that are linked to a central Web page, made available via the Web (the pages forms a cohesive collection of information). The Web server is a type of server dedicated to storing, transmitting and receiving the Web pages and Web related files (such GIF and JPEG graphics, AVI sound and images and so on).

The site's collection of linked files and Web pages are typically tied together into a cohesive collection of information by a home page (generally called default.htm[l], index.htm[l] or simply home.htm[l]). The letter "l" from .html is optional and not included for compatibilities with the "8.3" filenames format (and for that reason included in []). The home page typically contains a topic list (as a table of contents or index) which links it to other Web pages in its Web site. All other pages, in a well designed Web site, must offer a button or a link to go back home (or that is provided by the Web browser). When you publish your Web site, you upload the local site folder (and its contents including subfolders) to a Web server, which contains the software that "serves" your Web pages out to Web browsers on computers that are connected to the Internet. Once your local site is published to the Web server it becomes a Web site. The main or home page of the Web site is accessed by using Internet URLs/URIs. The Internet has two attributes that improve the company's success factor: Web sites can be accessed by a global audience 24 hours a day, 365 days a year, and those sites can be made to appear personalized for individual users. With a Web site organizations can also service the needs of their customers on a global, 24-hour-a-day basis, and marketers can finally realize their dream of mass-customized, one-to-one marketing when they structure Web sites effectively.

In order to be easy found by the target auditory the website must get best search engine visibility. To obtain that visibility the website design must follow at least the rules:

- easy to read;
- easy to navigate;
- easy to find;
- consistent in layout;
- consistent in design;
- compliance with standards.

In a layered approach a web document can consists of up to three layers:

1) *content layer*, that is always present and comprise the information the author whishes communicate to the target audience. The content layer is embedded within HTML or XHTML markup languages;

2) *presentation layer*, that defines how the content will be presented to the user (the layer can be realized with Cascading Style Sheet language);

3) *behavior layer*, that involves real-time user interaction with the document. This layer is realized generally by intermediate of scripting languages, from which the most used and platform independent is JavaScript.

Depending on the way a site interacts with the end user and reacts to user actions, how pages delivered as answer to user request realized (existing or generated), and what type of resources the site is able to manipulate the site architectures can be grouped in the following categories of architectures:

1. Static (HTML) Architecture;
2. DHTML Architecture;
3. High Level Languages based Architecture;
4. Dynamic Pages Architecture;
5. Advanced Management Architecture;
6. Multi-tier (three tiers) Architecture.

The architectures will be presented following a layered decomposition starting with the one introduced in § 2.1.1 The Logical Structure of Web Servers. For the second level, denoted by HTTP server, the description can have additional functionalities that will be introduced in the paragraphs bellow.

4.2 Static (HTML) Architecture

An e-commerce site, for example, offering the products catalog (the services for product presentation) and the recording of orders can be realized only by using HTML. The functional structure of such site is shown in figure 4.1.

In this architecture the site must contain, with except of general use pages (home page, contact, company presentation etc.), the following page types:

- the page for presentation of offered products/services (the catalog) as a rule in a shape of a table containing information of general interest about the offered product/service (such as product/service name, image, price, identification/ reference etc.);
- the pages containing details about the product/service;
- page with order fill-in form.

  In order to realize electronic transactions and placing orders the order fill-in form can contain or can be accompanied by:

  - page with user account fill-in form – page required to collect user identification data (name, address, phone number, e-mail address etc);
  - login form – required to connect and place orders
  - page with payment fill-in form.

  If that accompanying page types present the site cannot have only a static architecture and requires data repository for the corresponding records.

Generally the page with order fill-in form, the login form, payment form and user account manipulation must run in a secure environment ensured by the protocol HTTPS (HTTP Secure).

This architecture type is satisfactory for business that sell (offers) a small number of products/services (for example, selling automotives, real estate, consultancy etc) and having a pronounced static behavior/character of their evolution. With except of the functionalities related to consumer (client, buyer) the site must also offer specialized functionalities represented by:

- a management module (the Control Workstation in the figure want suggests the access point in that module) for processing the clients orders;
- an administrative module (that can be accessed from the administrative workstation) for additions/deletions of product presentation pages and maintenance of products/services catalog.

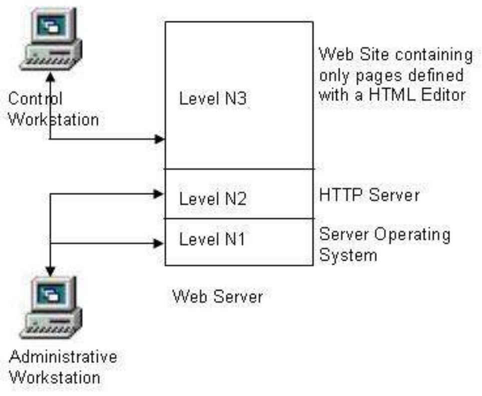


Figure 4.1 The functional structure of a site with static architecture

This operations can be realized at the host computer or from a company's workstation if the administration and maintenance of the site realized by the company itself. In that case, generally the Web service provider offers a tool for administration in a form of a control panel. Figure 4.2 shows a control panel of Plesk 7.5.6 for Microsoft Windows produced by the company SWsoft, Inc.

The level N3 is responsible for the management of site's Web pages: upload/download of pages, site directory maintenance, backup/recovery, security etc.

HTML Document

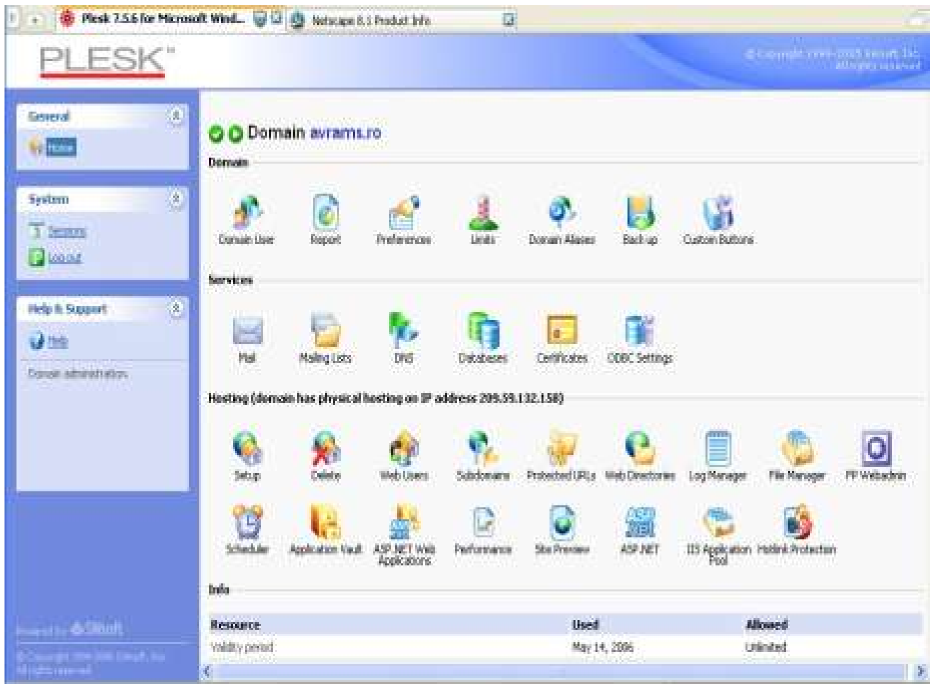
The commands for displaying text use their own language called Hypertext Markup Language, or HTML. HTML is nothing more than a coding system that combines formatting information in textual form with the readable text of a document.

In order to learn about HTML you must remember the following terms related to HTML and Web pages:

- **WWW** - World Wide Web
- **Web** - World Wide Web
- **SGML** - Standard Generalized Markup Language – a standard for describing markup languages
- **DTD** - Document Type Definition – this is the formal specification of a markup language, written using SGML
- **HTM**L - HyperText Markup Language – HTML is an SGML DTD

In practical terms, HTML is a collection of platform-independent styles (indicated by markup tags) that defines the various components of a World Wide Web document. HTML was invented by Tim Berners-Lee while at CERN, the European Laboratory for Particle Physics in Geneva (actually at MIT - USA and founder of World Wide Web Consortium, www.w3C.org). HTML is used to structure content of documents. The markup languages, as HTML or XML for example, have been hugely successful because they are both human-readable yet easily parseable by machines.

The browser reads the formatting commands and organizes the text in accordance with them, arranging it on the page, selecting the appropriate font and emphasis, and intermixing graphical elements. The HTML commands are set off by a special prefix (called tag's) so that the browser knows they are commands and not plain text. Writing in HTML is only a matter of knowing the right codes and where to put them. Web Authoring tools embed the proper commands using menu-driven interfaces so that you don't have to do the memorization. More than, today's HTML editors have a user friendly interface WYSIWYG, so that you can do the job visually and by using all the knowledge and practice you accumulate about word-processors (or, generally, about document editors).



Resource	Used	Allowed
Validity period	May 14, 2006	Unlimited

Figure 4.2 An example of Control Panel for the administrative workstation (Plesk 7.5.6 for Microsoft Windows; SWsoft, Inc)

WYSIWYG is an acronym for "what you see is what you get"; it means that you design your HTML document visually, as if you were using a word processor, instead of writing the markup tags in a plain-text file and imagining what the resulting page will look like. It is useful to know enough HTML to code a document before you determine the usefulness of a WYSIWYG editor, in case you want to add HTML features that your editor doesn't support.

HTML documents are plain-text (also known as ASCII) files that can be created using any text editor (e.g. *Emacs* or *vi* on UNIX machines; *SimpleText* on a Macintosh; *Notepad* on a Windows machine). You can also use word-processing software but you must save your document as "text only with line breaks" (Save as …, Save as type: Plain Text) or, if such option is present, save as HTML or as Web Page.

The HTML language includes a diversity of tags (markers), expressed following the next generalized syntax:

<*Tag_name*> Text associated………………….. [< / *Tag_name*>]

where <*Tag_name*> is the beginning of the tag and </*Tag_name*> is the ending of the tag. Tag’s are special text strings that are interpreted as formatting commands by the browser. Some tag's contains attributes (or parameters) that can take a finite number of specified values and whose syntax takes the format:

<*Tag_name attribut1*=" value 1" *attribut2*="value 2" … >

The attributes can be specified in any order, since they uses keywords, and the value assigned, does no matter his data type, must be enclosed in quotation marks. The pairs attribute-value and other keywords are separated by one or more spaces (at least one!). An HTML document contains hyperlinks, the embeded links to other documents on the web, that allows defining pathways between documents and surfing on the web. These hyperlinks contains several pieces of vital information, that instruct the Web browser where to go for content, such as:

- the protocol to use (generally HTTP);
- the server to request the document from;
- the path on the server to the document;
- the document's name (optional).

The information is assembled together in an URL.

Web documents are made up of several nested layers, each one delimited by a specific HTML tag. The first tag in a HTML document is DOCTYPE (document type) specified in constructions similar to the following:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

This tag specifies the following information:

- The document's top tag level is HTML (html);
- The document adheres to the formal public identifier (FPI) "W3C HTML 4.01 English" standards (PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN);
- The full DTD can be found at the URI http://www.w3.org/TR/html4/loose.dtd.

The minimal structure of an HTML document is shown in figure 4.3. Any HTML document includes a heading and a body, and any other tag's needed to specify the document structure, appearance and behavior:

a) <HTML>: defines the beginning/ending of the document;
b) <HEAD>: beginning/ending document heading;
c) <TITLE>: beginning/ending document title;
d) meta-tags: allows embeding extra-information within a webpage;
e) <BODY>:

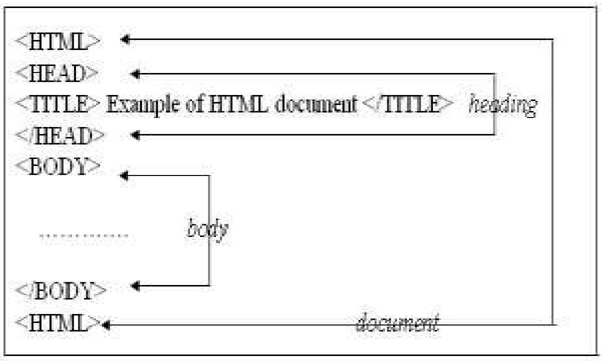


Figure 4.3 The HTML document structure

beginning/ending of the document itself.

An element is a fundamental component of the structure of a text document. Some examples of elements are *heads, tables*, *paragraphs*, and *lists*. Think of it this way: you use HTML tags to mark the elements of a file for your browser. Elements can contain plain text, other elements, or both. To denote the various elements in an HTML document, you use tags. HTML tags consist of a left angle bracket (<), a tag name, and a right angle bracket (>). Tags are usually paired (e.g., <H1> and </H1>) to start and end the tag instruction. The end tag looks just like the start tag except a slash (/) precedes the text within the brackets.

Some elements may include an attribute, which is additional information that is included inside the start tag. For example, you can specify the alignment of images (top, middle, or bottom) by including the appropriate attribute with the image source HTML code.

NOTE: HTML is not case sensitive. <title> is equivalent to <TITLE> or <TiTlE>. There are a few exceptions noted in Escape Sequences.

Not all tags are supported by all World Wide Web browsers. If a browser does not support a tag, it will simply ignore it. Any text placed between a pair of unknown tags will still be displayed, however.

Every HTML document should contain certain standard HTML tags. Each document consists of head and body text. The head contains the title, and the body contains the actual text that is made up of paragraphs, lists, and other elements. Browsers expect specific information because they are programmed according to HTML and SGML specifications.

4.3 DHTML Architecture

DHTML stands for Dynamic HTML and is not a W3C standard. It was defined by Netscape and Microsoft as a "marketing term" describing a new technology that the generation 4.x (and following) of browser must support. It is a combination of "HTML/XHTML, style sheets and scripts that allows documents to be animated". With DHTML a web developer can control how to display and position dynamically the HTML elements in a window. This is possible by intermediate of HTML DOM, a W3C standard that defines a standard set of objects for HTML and a standard way to access and manipulate these HTML objects. The HTML DOM specifies the objects together with their associated properties (or attributes, generally the equivalent to tag attributes such as *id*, *name*, *alt*, *title* etc), and where appropriate, methods that can be invoked for the object (such as *blur()*, *focus()*, *click()* etc).

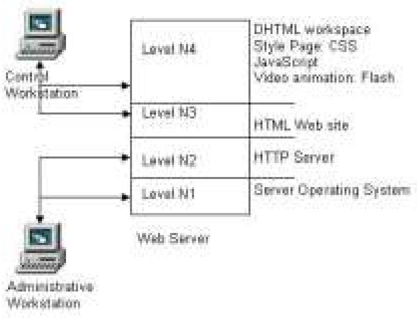


Figure 4.4 The functional structure of a site with DHTML architecture

DHTML represents the usage of a lot of languages that allows realizing animated pages and presentations, computations, and the control and validation of primary fields in fill-in forms, such as:

- HTML/XHTML;
- Style Pages (Cascading Style Sheets);
- Scripts (JavaScript, VBScript etc).

The animation inside of the site can be realized using different specialized applications from which Flash is the one most frequently used. The functional architecture of the site with DHTML is shown in figure 4.4.

The site content is similar to those previously described the difference being given by the way in which the presentation take place to the end user (represented by client). In this architecture the new level introduced N4 contains:

- Cascading Style Sheets;
- Scripts;
- Video animation (Flash).

These features can help to make the site easier to read, navigate, and react. They must be used in such ways that preserve or rise the search engine visibility (remember that search engine crawlers look for text on a web page and index and rank the page according to that text). Below is a brief presentation of these features. They accompanied by an extended introduction containing examples of usage in different circumstances in separate chapters.

4.3.1 CSS - Cascading Style Sheets

CSS is a language that allows defining the way in which a document displayed referring to the usage of parental levels of pages, of the fonts (name, color, size) and font family, or in other words, CSS is a style language that defines layout of HTML documents. CSS is used for formatting structured content. By using styles we can change the definition once and the change affects every element using that style. The styles provide an easy means to update document formatting and maintain consistency across a site. Styles are grouped together in style sheets and are normally stored in external files with a *.css* extension. The external style sheet allows define and change just once and apply that to more pages. The addition of CSS style in a document can be realized:

a) in the current line (inline style, the style defined directly in the line specifying the tag);
b) global, by specifying the document style at his beginning (in the heading part);
c) by linking the style page (defined in a separate document and stored in a separate file, too) to the document by using tags with the general syntax:
   <LINK REL=”stylesheet” TYPE=”text/css” HREF=”styldocumentname.css”>
d) browser default.

Example:
<link rel="stylesheet" type="text/css" href="styles/sitestyl.css">
where the sitestyl.css file contains:

The CSS syntax is very simple and is made up of three elements *selector* {*property*: *value*; *property*: *value* …} where the *selector* is normally the HTML element/tag you whish define the style (such as BODY, A:link, DIV.intro etc), and the *property* is the attribute you whish change for the selector (such as BACKGROUND, COLOR, FONT-FAMILY for BODY selector, for example),

```
BODY { BACKGROUND: white; COLOR: black; FONT-FAMILY: sans-serif }
A:link { BACKGROUND: none transparent scroll repeat 0% 0%; COLOR: #00e }
A:active { BACKGROUND: none transparent scroll repeat 0% 0%; COLOR:
#00e }
A:visited { BACKGROUND: none transparent scroll repeat 0% 0%; COLOR:
#529 }
A:hover{background:transparent none repeat scroll 0% 0%;color:#999999}
DIV.intro { MARGIN-LEFT: 5%; MARGIN-RIGHT: 5%; FONT-STYLE: italic }
PRE { FONT-FAMILY: monospace }
A:link IMG { BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none;
BORDER-LEFT-STYLE: none; BORDER-BOTTOM-STYLE: none }
A:visited IMG { BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none;
BORDER-LEFT-STYLE: none; BORDER-BOTTOM-STYLE: none }
A IMG { COLOR: white }
@media All { A IMG { } }
UL.toc { LIST-STYLE-TYPE: none }
DIV.issue { PADDING-RIGHT: 0.5em; PADDING-LEFT: 0.5em; PADDING-
BOTTOM: 0.5em; BORDER-TOP-STYLE: none; MARGIN-RIGHT: 5%; PADDING-
TOP: 0.5em; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
BORDER-BOTTOM-STYLE: none }
.hideme { DISPLAY: none }
.avbkgtop{border:medium none;background-position:center;BACKGROUND-
IMAGE:url('http://www.avrams.ro/imgs/tt077.jpg');FONT-FAMILY:'Times
New Roman';BACKGROUND-COLOR:transparent}
#menub{padding:0;margin:0;height:1em;list-style-type:none;border-left:1px
solid #c0c0c0;color: #bbbbbb;font-family:Verdana;font-weight: normal;font-
size: xx-small}
#menub li{float:left;width:8em;height:1em;line-height:1em;border-right:1px
solid #c0c0c0;position:relative;text-align:center;color:#efbb22;font-family:
Verdana;font-weight:normal;font-size:xx-small}
#menub li a, #menub li a:visited{display:block;text-
decoration:none;color:#efbb22}
#menub li a span, #menub li a:visited span{display:none}
#menub li a:hover{border:0px none;color:#c0c0c0}
#menub li a:hover span{display:block;width:8em;height:1em;text-
align:center;position:absolute;left:-1px;top:-
4px;color:#efbb22;cursor:pointer;font-family:Verdana;font-
weight:normal;font-size:x-small}
@media Print  { TABLE { page-break-inside: avoid } }
ul, li { margin-left:0px}
.marybkg
{  border: medium none;
   BACKGROUND-IMAGE: url('http://www.avrams.ro/brad-0167.jpg');
   FONT-FAMILY: 'Times New Roman';
   BACKGROUND-COLOR: transparent;
   BACKGROUND-REPEAT: repeat; }
```

and the *value* specifies the value you whish assign to the property (such as white for BACKGROUND, black for COLOR in the BODY selector).

CSS supports the following metrics for property values [W3C; SS05]:

- CSS keywords and other properties, such as thin, thick, transparent, ridge, and so forth
- Real-world measures:
  Inches (in)
  Centimeters (cm)
  Millimeters (mm)
  Points (pt) (1/72 of an inch)
  Picas (pc) (1 pica=12 points)
- Screen measures in pixels (px)
- Relational to font size (font size (em) or x-height size (ex))
- Percentages (%)
- Color codes (#rrggbb or rgb(r,g,b))
- Angles:
  Degrees (deg)
  Grads (grad)
  Radians (rad)
- Time values (seconds (s) and milliseconds (ms)) — Used with aural style sheets
- Frequencies (hertz (Hz) and kilohertz (kHz)) — Used with aural style sheets
- Textual strings

CSS gives many benefits to site designers such as:

- realizing control layout of many documents (web pages) from one single style sheet;
- a more precise control of layout;
- the possibility to define and apply different layout for different media type (display, print etc) to the same document;
- the availability of numerous advanced and sophisticated techniques.

There are three levels of CSS the main differences between them are as follows:

- CSS1 defines basic style functionality, with limited font and limited positioning support;
- CSS2 adds aural properties, paged media, and better font and positioning support;
- CSS3 adds presentation-style properties, allowing you to effectively build presentations from Web documents (similar to Microsoft PowerPoint presentations).

All styles defined for a document “cascade in a virtual” style following this rules (from a high to low priority):

1. Inline style;
2. Global style sheet;

3. External style sheet;
4. Browser default.

4.3.2 Scripts

The scripts are source programs expressed in a scripting language and can act at client side or at server side. Not all scripting languages allows writting scripts for both sides. In the paragraphs below we talk about client side scripting and scripting languages alowing defining those scripts.

The client side scripts are small source programs, expressed in a scripting language, embeded in the HTML page and that the browser, when loading and displaying the page, interprets and executes them. A scripting language is a lightweight programming language designed to add interactivity to HTML pages. A scripting language gives to HTML designers a programming tool with a easy to use simple syntax.

The scripts inside the web page can change dinamically the page (for example, can change the font size) when some events appears (for example, clicking the mouse), can verify if data in a fill-in form are correct or not, etc. To include a script in a HTML page place his content between the tags <script> and </script> or between the additional tags <noscript> and </noscript> for the browsers that do not understand scripts. The scripting languages situated at an intermediary level between HTML and high level programming languages such as JAVA, C++ and Visual Basic. If HTML used generally for formatting the text and link creation and the programming languages for complex instruction the script languages used for specifying of complex instructions whose syntax is more flexible than those of programming languages. Meanwhile, the script languages can format the text, and in that way they realize the interaction between the web page and the user.

4.3.2.1 DOM - Document Object Model

The scripts allow restructuring an entire HTML/XHTML document for which we can add, remove, change, or reorder items on a page. In order to change anything on a page, the script needs access to all elements in the HTML/XHTML document. This access, along with methods and properties to add, move, change, or remove HTML/XHTML elements, is given through the Document Object Model (DOM). The Document Object Model view HTML/XHTML documents as a collection of objects (having attributes/properties and methods) and provides access to every element in a document. Every element is modeled in a web browser as a DOM *node*, and the nodes make up the DOM *tree* describing the relationships between elements in a *child-parent* fashion. The children of the same node (having the same parent) referred to as *siblings*. A node can have multiple children but only one parent. Because of that access, any element may be modified by a snippet of JavaScript. Elements are easily accessed by use of an *id* attribute (that must be unique within a given document) and a method of the document object. The *document* object is the parent of all the other objects in an HTML/XHTML document, for example *document.title* represents the <title> element of the HTML/XHTML document as in the figure 4.5.

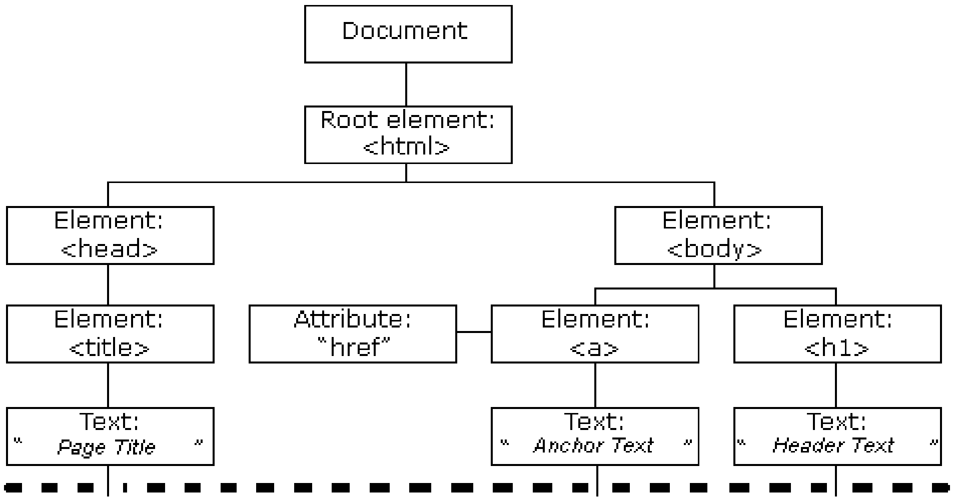


Figure 4.5 The tree structure of HTML documents (partly; theoretical)

In 1998, W3C published the Level 1 DOM specification. This specification allowed access to and manipulation of every single element in an HTML page. All browsers have implemented this recommendation, and therefore, incompatibility problems in the DOM have almost disappeared. The DOM can be used by JavaScript to read and change HTML, XHTML, and XML documents. The DOM is separated into different parts (Core, XML, and HTML) and different levels (DOM Level 1/2/3):

- Core DOM - defines a standard set of objects for any structured document;
- XML DOM - defines a standard set of objects for XML documents;
- HTML DOM - defines a standard set of objects for HTML documents.

Every object can have his own Collections, Attributes (Properties) and Methods. The table 4.1 gives the objects in DHTML DOM.

Table 4.1 The objects in DHTML DOM

Object	Description
window	The top level object in the DHTML DOM. It contains information about the window and the frames. The objects listed below are the children

	of the window object.
document	Represents the HTML document, and is used to access the HTML elements inside the document
frames	Represents the frameset
history	Keeps track of the sites visited by the browser object.
navigator	Contains information about the user's browser
location	Contains the URL of the rendered document
event	Contains information about events that occurs
screen	Contains information about the computer screen for the computer on which the browser is

Table 4.2 shows the properties (attributes) of a Document object as implemented in Microsoft Script Editor (Windows).

Table 4.2 The properties of a HTML Document object

Property	Description
aLink	Sets the color of hyperlinks as they are clicked.
Background image	Provides the path to a background image for the page.
bgcolor	Sets a background color for the page.
bgProperties	Sets whether the background for the page will be scrolling (default) or fixed (watermark)
bottomMargin	Sets the height of the blank margin at the bottom of the page.
charset	Selects the character set for the page.
defaultClientScript	Sets the default client scripting language.
dir	Sets the reading order of page objects.
keywords	Adds keywords to the META KEYWORDS tag in the <HEAD> of your document
leftMargin	Sets the width of the blank margin on the left side of the page.
link	Sets the default color of hyperlinks before they are clicked.
pageLayout	Selects whether page components added in design view will be positioned in-line as they occur on the page or positioned at specified locations (enables the positioning grid).
rightMargin	Sets the width of the blank margin on the right side of the page.
showGrid	Determines whether the positioning grid will appear in Design View.
targetSchema	Sets the minimum version of HTML required

	to view this page, and (in some cases) the preferred document object model (DOM) for client scripts on the page.
text	Sets the default color for foreground text on the page.
title	Provides the text string inserted between the <TITLE> and </TITLE> tags in the page HEAD.
topMargin	Sets the height of the blank margin at the top of the page.
vLink	Sets the default color of hyperlinks that have been clicked.

Table 4.3 shows the methods of a Document object.

Table 4.3 The methods of Document object

Method	Description
close()	Closes an output stream opened with the document.open() method, and displays the collected data
getElementById("id")	Returns a reference to the first object (node) with the specified "id"
getElementsByName("name")	Returns a collection of objects with the specified "name"
getElementsByTagName("tag name")	Returns a collection of objects with the specified "tagname"
open()	Opens a stream to collect the output from any document.write() or document.writeln() methods
write()	Writes HTML expressions or JavaScript code to a document
writeln()	Identical to the write() method, with the addition of writing a new line character after each expression

4.3.2.2 JavaScript

JavaScript is a platform independent scripting language that gives HTML designers a programming tool and that can be used for easy management of user interface: it can put dynamic text into a HTML page, it can make the page react to events or it can create and easy manipulate cookies. A JavaScript consists of lines of executable computer code usually embedded directly into HTML pages. JavaScript is an interpreted language and can be used without purchasing a license. JavaScript is a client-based language, a scripting language with definite limitations, and code visible in the document it appears. Can be used for tasks such as: forms verification, document animation and automation, and basic document intelligence (changes in the document based on other dynamic criteria by exploiting the DOM model).

A JavaScript inserted in the HTML document allows a local recognition and processing (that means at client level) of the events generated by the user such as those generated when the user scans the document or for management of fill-in forms (for example, we must recuperate the information referencing the client such as name, address, payment etc). By inserting a JavaScript in the HTML page we can validate the data filled by the client (for example we can validate the Credit Card Account, we can check for solvability, we can see transactions history, etc) before it is submitted to the server.

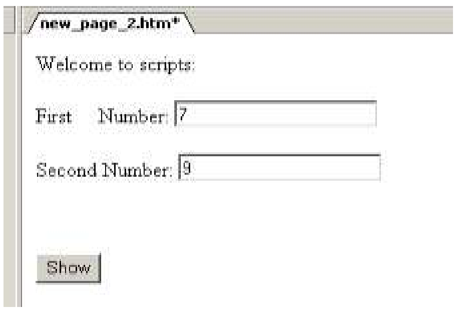
A JavaScript can:
- put dynamic text into a HTML;
- react to events;
- read and write HTML elements;
- be used to validate data;
- be used to detect the visitor's browser;
- be used to create cookies.

JavaScript allows restructuring an entire HTML document for which we can add, remove, change, or reorder items on a page. In order to change anything on a page, JavaScript needs access to all elements in the HTML document through the Document Object Model (DOM).

JavaScript is hardware and software platform independent. Within a JavaScript inserted in the HTML page we can validate the data supplied by the client (for example, to validate the card account, financial availability, history regarding previous transactions etc.).

For an inserted JavaScript the <script type="text/javascript"> and </script> tags tells where the JavaScript starts and ends:

```
<html>
<body>
<script type="text/javascript">
<!--
... // put here the script body
//-->
</script>
</body>
</html>
```



Welcome to scripts:

First Number: 7

Second Number: 9

A JavaScript example - what the browser displays

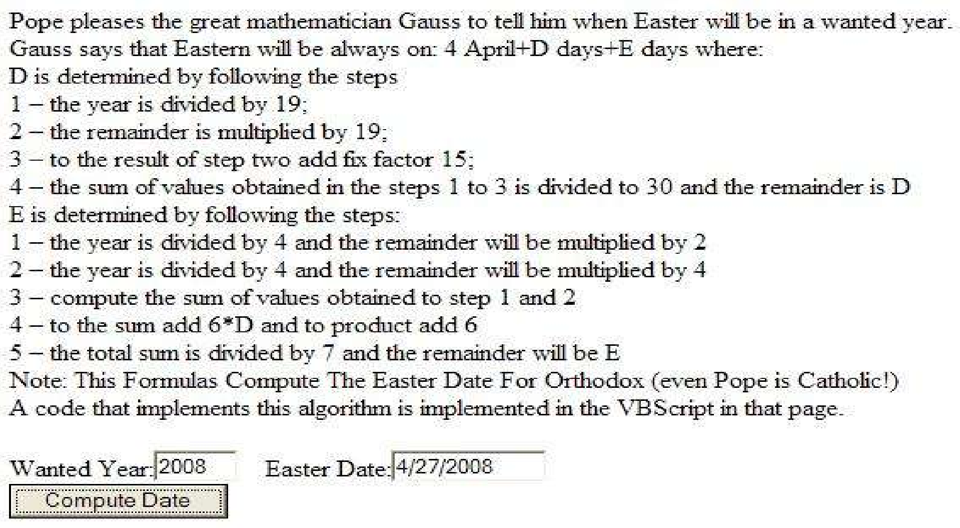
```
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
<title>New Page 2</title>
<script type="text/javascript" language="javascript">
       <!--
         function calc(a, b){ return (a*b);}
        -->
      </script>
      <script id=clientEventHandlersVBS language=vbscript>
      <!--
      Sub Validate_onclick
         document.write("You
      Type:"+cstr(text1.value)+":"+cstr(text2.value))
      End Sub
      -->
</script>
</head>
<body>
   <script type="text/javascript" language="javascript">
          var welcmess="Welcome to scripts:";
          document.write(welcmess)
   </script>
   <p></p> First&nbsp;&nbsp;&nbsp;&nbsp; Number: <INPUT
type="text" ID=Text1 value="0" name="text1" size="20">
    <p></p> Second Number: <INPUT type="text" ID=Text2 value="0"
name="text2" size="20">
    <p></p>&nbsp;<p><INPUT type="button" value="Show"
id="Validate"></p>
</body>
</html>
```

We can validate the data supplied by the client by inserting of a Java script in the HTML page (for example, to validate the card account, financial availability, history regarding previous transactions etc.).

In the example that follows is defined a function that compute when easter will be for a wanted year. The algorithm is the same as the one introduced in the figure 4.6. The example is also an illustration of the usage of parenthesis to change the order of evaluation of the terms of expressions. The button labeled "JavaScript Solution", in the HTML code that follows, calls the function every time the user clicks the button onclick="easter_datejs(wantedYear.value)". The algorithm check if the supplied year is a positive year or not, case in which an error signaled to the user. If the year is a positive one it computes the values for D and E, as defined in Gauss algorithm in figure 4.6, reconstitute the easter date computed for the year filled by user, and returns the easter date in the form.

```
. . .
<script type="text/javascript" language="javascript">
 <!--
 function easter_datejs(Wanted_Year){
   var D;
   var E;
   if (Wanted_Year<0){
     alert("The value for Year must be a positive number!");
     return -1;
   }

   D = ((Wanted_Year % 19) * 19 + 15) % 30;
   E = (((Wanted_Year % 4) * 2 + (Wanted_Year % 7) * 4) + 6 * D +
6) % 7;
   D=D+E+4;

   if(D>30){
     easterDate.value='5/'+(D-30.)+'/'+Wanted_Year;
     }
     else
     {
     easterDate.value='4/'+(D)+'/'+Wanted_Year;
  }

 }
 -->
</script>

:
:
<div style="border:1px solid #c0c0c0">
Wanted Year:<INPUT type="text" id="wantedYear" name="wantedYear"
size="4" align="right" value="2008" >
&nbsp;&nbsp;&nbsp;
Easter Date:<INPUT type="text" id="easterDate" name="easterDate"
size="11" align="right" readonly="readonly"><br/>
<br/>

<INPUT type="button" value="VBScript Solution" ID="vbcompdat"
onclick="vbvalidComp()">

<INPUT type="button" value="JavaScript Solution" ID="jscompdat"
onclick="easter_datejs(wantedYear.value)">
</div>
. . .
```


Pope pleases the great mathematician Gauss to tell him when Easter will be in a wanted year.
Gauss says that Eastern will be always on: 4 April+D days+E days where:
D is determined by following the steps
1 – the year is divided by 19;
2 – the remainder is multiplied by 19;
3 – to the result of step two add fix factor 15;
4 – the sum of values obtained in the steps 1 to 3 is divided to 30 and the remainder is D
E is determined by following the steps:
1 – the year is divided by 4 and the remainder will be multiplied by 2
2 – the year is divided by 4 and the remainder will be multiplied by 4
3 – compute the sum of values obtained to step 1 and 2
4 – to the sum add 6*D and to product add 6
5 – the total sum is divided by 7 and the remainder will be E
Note: This Formulas Compute The Easter Date For Orthodox (even Pope is Catholic!)
A code that implements this algorithm is implemented in the VBScript in that page.

Wanted Year: 2008 Easter Date: 4/27/2008

Figure 4.6 The algorithm to determine Easter date

For more details about JavaScript see the chapter "Introduction to JavaScript – theory and examples" in http://www.avrams.ro site.

4.3.2.3 VBScript

VBScript defined by Microsoft as a scripting language deduced from Visual Basic (more exactly a subset of Visual Basic for Application). A VBScript works (runs) with the browser Internet Explorer of Microsoft (in that way is software platform dependent). Figure 4.6 shows an example of an execution of a VBScript placed in the body of the HTML page together with the corresponding VB code, as shown in the implementation bellow.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
<title>Determine Easter Date</title>

<script language=”vbscript”>
<!--
function Easter_Date(Wanted_Year)
        Dim D, E
        D = ((Wanted_Year Mod 19) * 19 + 15) Mod 30
        E = (((Wanted_Year Mod 4) * 2 + (Wanted_Year Mod 7) * 4)
+ 6 * D + 6) Mod 7
        Easter_Date = DateAdd("d", (D +
E+0.),CDate("04/04/"+Trim(Wanted_Year)))
```

```
end function
function validComp()
        if not isnumeric(trim(wantedYear.value)) then
           msgbox "Err.1. Wrong value for year: is not a
number!",,"Easter Date"
           wantedYear.focus
           exit function
        end if
        if len(trim(wantedYear.value))<3 or
len(trim(wantedYear.value))>4 then
           msgbox "Err.2. Wrong value for year: is a to little/large
number (min 3 and max 4 digits allowed)!"+Chr(10)+chr(13)+"[This
limitation is given by the way in which Microsoft implements the
functions]"+Chr(10)+chr(13)+"[DateAdd(...) and CDate(...)].
Eliminate this test and enjoy computing!",,"Easter Date"
           wantedYear.focus
           exit function
        end if
        easterDate.value=Easter_Date(trim(wantedYear.value))
end function
-->
</script>
</head>

<body>
   Pope pleases the great mathematician Gauss to tell him when
Easter will be in a wanted year.<br/>
   Gauss says that Eastern will be always on:  4 April+D days+E days
where:<br/>
D is determined by following the steps<br/>
1 – the year is divided by 19;<br/>
2 – the remainder is multiplied by 19;<br/>
3 – to the result of step two add fix factor 15;<br/>
4 – the sum of values obtained in the steps 1 to 3 is divided to 30 and
the remainder is D<br/>
E is determined by following the steps:<br/>
1 – the year is divided by 4 and the remainder will be multiplied by 2
<br/>
2 – the year is divided by 4 and the remainder will be multiplied by
4<br/>
3 – compute the sum of values obtained to step 1 and 2<br/>
4 – to the sum add 6*D and to product add 6<br/>
5 – the total sum is divided by 7 and the remainder will be E<br/>
Note: This Formulas Compute The Easter Date For Orthodox (even
Pope is Catholic!)<br/>
A code that implements this algorithm is implemented in the VBScript
```

```
in that page.<br/><br/>
<div style="border:2px">
Wanted Year:<INPUT type="text" id="wantedYear"
name="wantedYear" size="4" align="right" >
&nbsp;&nbsp;&nbsp;Easter Date:<INPUT type="text" id="easterDate"
name="easterDate" size="11" align="right" readonly="true"><br/>
<INPUT type="button" value="Compute Date" ID="compdat"
onclick="validComp()">
</div>
</body>
</html>
```

If the user types a year value in the “Wanted Year” box then by pressing the “Compute Date” button (the *onclick* event) is called the function validComp() that realize some validity checks for the year. If the value for year you supply conforms to the validity checks requirements the function Easter_Date() is invoked having as argument the year for which you whish know when Easter will be. The return of the function is stored in the text box “Easter Date” and in that way the answer is delivered to the user.

Being a subset of VBA VBScript allows the usage of most familiar functions of these one. For more details about VBScript see the chapter “Introduction to VBScript – theory and examples” in http://www.avrams.ro site.

4.3.3 Flash

Macromedia Flash is a powerful tool for realizing and deploying a wide range of media content. The animation in the web pages can be done, in the simplest way, by intermediate of vectorial representation. Flash is an application that allows vectorial definition of graphics. The vectorial description uses mathematical expressions for describing the dimension (size), form (shape), positioning, and any other characteristics. In that way the loading of a figure becomes a simple transfer of formulas (defined and transferred in text format) from which the image can be reconstituted. To have an idea about the size of an image described with Flash you must think that is 10 times smaller than his representation GIF format. Like the applets, the Flash files (with the default extension “.swf”), can be displayed on Web pages by referencing them from HTML files. According to Macromedia Company, that produces Macromedia Flash, 95% of installed browsers can interpret Flash. The application Macromedia Flash allows creating Flash modules by dialogs. After creation the module can be exported in an external file (for example, *film.swf*) that can be referenced (as object) from the HTML page in that way:

```
<object width="400" height="300">
<param name="movie" value="film.swf">
<!-- additional tag for Netscape -->
  <embed src="film.swf" width="400" height="300">
  </embed>
</object>
```

In this sample the tag <object> is recognized by Internet Explorer but non-recognized by Netscape. For Netscape we can add the tag <embed>. Macromedia Flash allows generating the HTML pages provided with the tags required to include the Flash modules in the page and the alternatives links to the Flash interpretor on Macromedia site (if browser that display the page do not know Flash). There is an example (from ASE portal, home page, once upon) of such usage:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://
download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0"
width="191" height="35">
<param name="movie" value="Banner/preadmitere.swf" />
<param name="quality" value="high" />
<embed              src="Banner/preadmitere.swf"              quality="high"
pluginspage="http://www.macromedia.com/go/getflashplayer"    type="application/x-
shockwave-flash" width="191" height="35">
</embed>
</object>
```

Macromedia definition of Flash is: "Flash is an authoring tool that designers and developers use to create presentations, applications, and other content that enables user interaction."

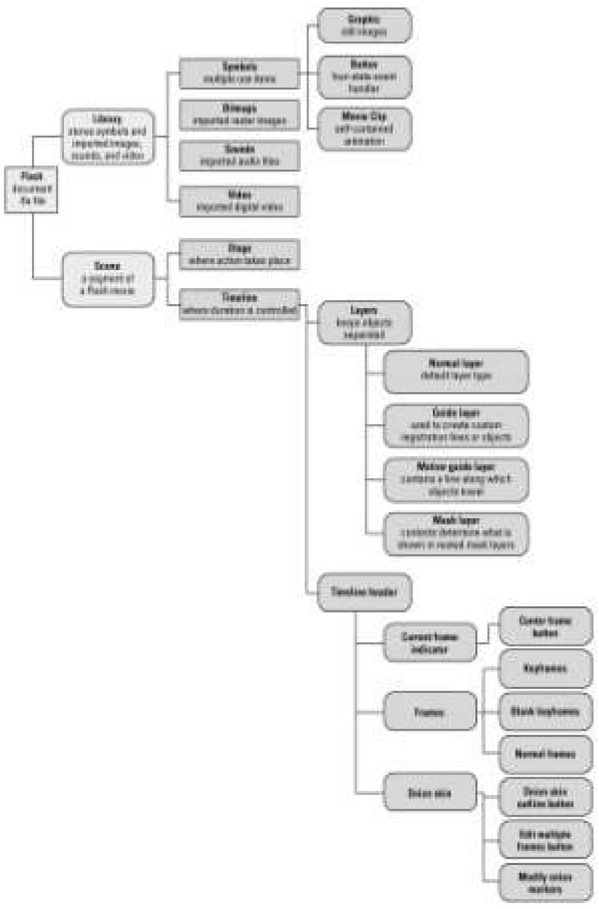
The individual pieces of content created with Flash, in a process denoted by the notion "authoring document", are called applications and are stored in files having a *.fla* extension. When you have finished authoring and publish the file a compressed version (with a very small size: a JPEG graphic can be represented in a Flash document 11 times less in size of the file) of file with the extension *.swf* (SWF) will be uploaded. This file can be played by Flash Player in a browser or as a standalone application.

A Flash application is build generally by mixing graphics created with the Flash drawing tools (visually) together with imported additional media and defining how and when you to use each of those elements.

A Flash document has four main parts:

1) **Stage** - specifying **where** graphics, videos, buttons, and so on appears during playback;
2) **Timeline** - is the part where we specify **when** we want the graphics and other elements of the project to appear;
3) **Library panel** - is where Flash **displays a list** of the media elements in the Flash document;
4) **ActionScript code** - that allows **adding interactivity** to the media elements in the document.

Figure 4.7 shows **t**he structure of a **.fla** file [RRSD] and how Flash documents are composed of individual scenes that contain keyframes to describe changes on the Stage. The figure shows also the efficiency of sharing Flash Libraries among several Flash documents by loading other Flash movies into a parent, or "master," Flash movie using the *loadMovie()* action, or creating interactive elements with scripting methods.



Flash document .fla file
Library stores symbols and imported images, sounds, and video
Symbols multiple use items
Graphic still images
Button four-state event handler
Movie Clip self-contained animation
Bitmaps imported raster images
Sounds imported audio files
Video imported digital video
Scene a segment of a Flash movie
Stage where action takes place
Timeline where duration is controlled
Layers keeps objects separated
Normal layer default layer type
Guide layer used to create custom registration lines or objects
Motion guide layer contains a line along which objects travel
Mask layer
Timeline header
Current frame indicator
Center frame button
Frames
Keyframes
Blank keyframes
Normal frames
Onion skin
Onion skin outline button
Edit multiple frames button
Modify onion markers

Figure 4.7 Elements of a Flash document (.fla) in the authoring environment (Source: Macromedia Flash 8 Bible by Robert Reinhardt and Snow Dowd, John Wiley & Sons © 2006, [RRSD])

The figure 4.8 outlines the characteristics of a published Flash file (.swf file) such as:

- **portability**, meaning that is compatible with most operating systems and browsers applications;
- **extensibility**, that allows adding new features in subsequent versions;
- **scalability**, allowing movies be played at multiple resolution with preventing loose in quality.

The heart of the Flash application is a vector-based drawing program that draws shapes by defining points that are described by coordinates. Lines that connect these points are called paths, and vectors at each point describe the curvature of the path. Because this scheme is mathematical, there are two distinct advantages the vector content is significantly more compact, and it's thoroughly scalable without image degradation, advantages that are especially significant for Web use.

The vector animation component of the Flash application relies on the slim and trim vector format for transmission of the final work. Instead of storing megabytes of pixel information for each frame, Flash stores compact vector descriptions of each frame.

Flash quickly renders the vector descriptions as needed and with far less strain on either the bandwidth or the recipient's machine and this is, indeed, a huge advantage when transmitting animations and other graphic content over the Web.

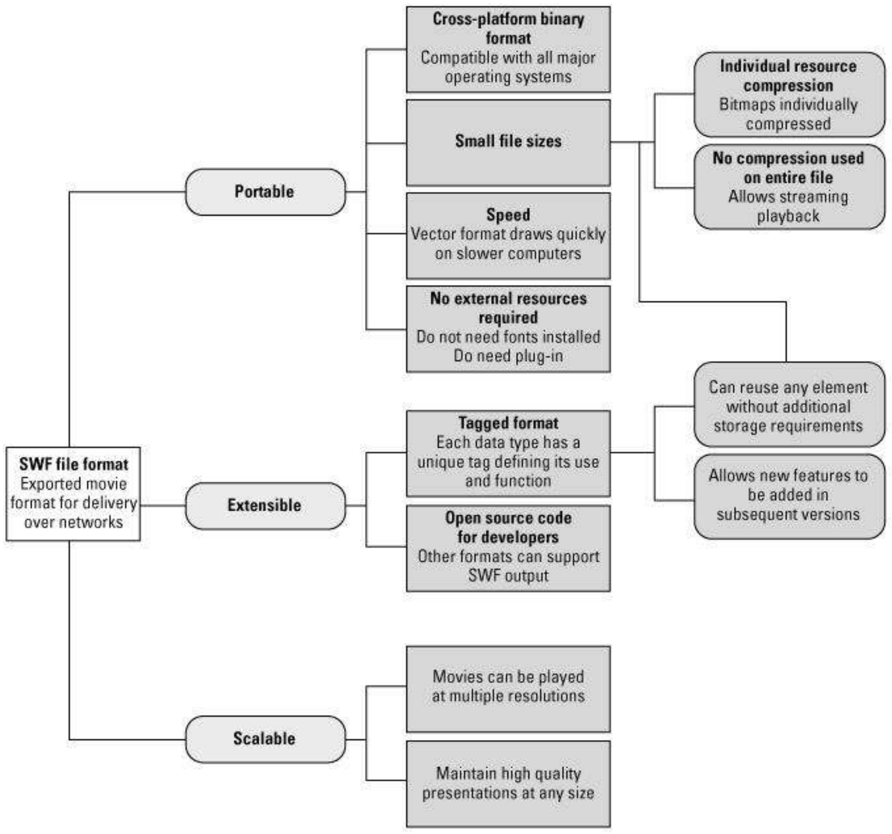

SWF file format
Exported movie format for delivery over networks
Portable
Extensible
Scalable
Cross-platform binary format
Compatible with all major operating systems
Small file sizes
Speed
Vector format draws quickly on slower computers
No external resources required
Do not need fonts installed
Do need plug-in
Individual resource compression
Bitmaps individually compressed
No compression used on entire file
Allows streaming playback
Tagged format
Each data type has a unique tag defining its use and function
Open source code for developers
Other formats can support SWF output
Can reuse any element without additional storage requirements
Allows new features to be added in subsequent versions
Movies can be played at multiple resolutions
Maintain high quality presentations at any size

Figure 4. 8 Overview of the Flash movie (.swf) format (Source: Macromedia Flash 8 Bible by Robert Reinhardt and Snow Dowd, John Wiley & Sons © 2006**)**

4.3.4 Ajax

Asynchronous JavaScript and XML – uses the JavaScript-based XMLHttpRequest object to fire requests to web server asynchronously (or without having to refresh the page). Figure 4.9 shows the usage of traditional server request/response model, the most used technology in Internet, in which the web server responds with a new content for the page at the user request, together with the use of Ajax asynchronous methodology in which the server responds with changes within the web page as answer to the user requests.

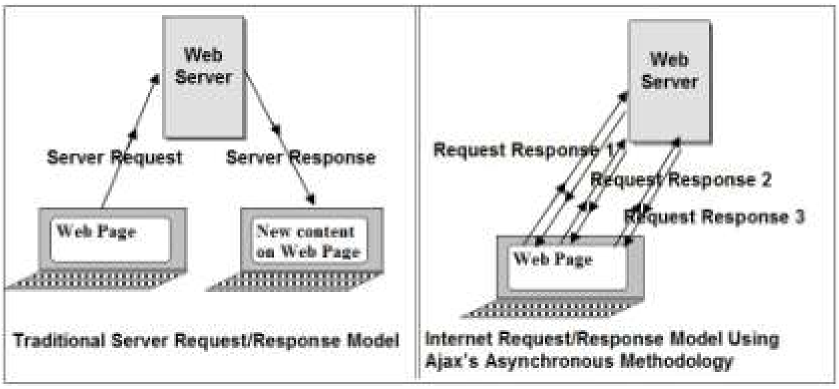
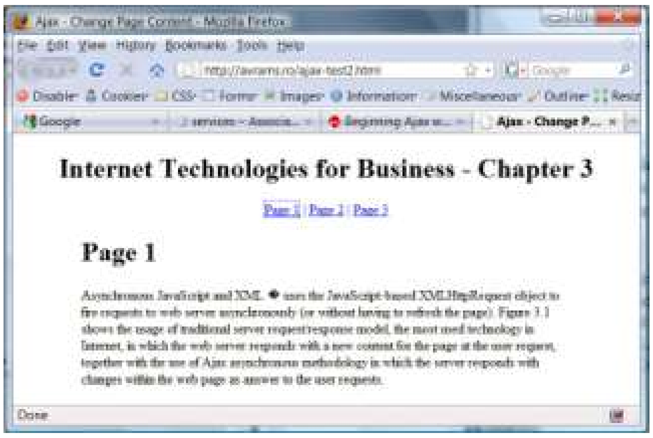


Figure 4. 9 Traditional Server Request/Response Model vs. Ajax Methodology

Ajax is not a really technology at all. It is a term to describe the process of using JavaScript-based XMLHttpRequest object to retrieve information from web server in a dynamic manner (asynchronously). The only requirement to use Ajax is to enable JavaScript within the used browser to surf on Internet. Even whether is based on JavaScript and seem be difficult the structure of an Ajax-based server request is quite easy to understand and invoke: you must simply create an object of the XMLHttpRequest type, validate that it has been created successfully, point where it will go and where the result will be displayed, and then send it.

Figure 4.10 shows an Ajax page as displayed in Mozilla Firefox browser. By pressing on one of the outlined hyperlinks will start the event "onclick" that calls a function named *makerequest(url)* having as argument an URL address.

The function loads the resource at the indicated address and replaces the part indicated from the page with the text response from the server where that resource resides. Figure 4.11 shows the source code of that page including the Ajax code.


Ajax - Change Page Content - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://avrams.ro/ajax-test2.htm
Google
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resiz
Google
services - Associa...
Beginning Ajax w...
Ajax - Change P...
Done

Internet Technologies for Business - Chapter 3

Page 1 | Page 2 | Page 3

Page 1

Asynchronous JavaScript and XML ● uses the JavaScript-based XMLHttpRequest object to fire requests to web server asynchronously (or without having to refresh the page). Figure 3.1 shows the usage of traditional server request/response model, the most used technology in Internet, in which the web server responds with a new content for the page at the user request, together with the use of Ajax asynchronous methodology in which the server responds with changes within the web page as answer to the user requests.

Figure 4. 10 An Ajax page example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=windows-1252" http-equiv="Content-Type"
/>
<title>Ajax - Change Page Content</title>
<script type="text/javascript">
<!--
var xmlhttp;
function state_Change()
{
if (xmlhttp.readyState==4)
   {// 4 = "loaded"
   if (xmlhttp.status==200)
    {// 200 = "OK"
    document.getElementById('repme').innerHTML=xmlhttp.responseText;
    }
   else
   {
    alert("Problem retrieving data:" + xmlhttp.statusText);
    }
  }
}
function makerequest(url){
xmlhttp=null;
if (window.XMLHttpRequest){// code for Firefox, Opera, IE7, etc.
  xmlhttp=new XMLHttpRequest();
  } else if (window.ActiveXObject){// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
if (xmlhttp!=null)  {
  xmlhttp.onreadystatechange=state_Change;
  xmlhttp.open("GET",url,true);
  xmlhttp.send(null);
  }else {
  alert("Attention! Your browser does not support Ajax. To view properly that
page Enable usage of JavaScript / ActiveX");
}}
//-->
</script>
</head>
<body onload="makerequest('pgs/ajax-1.html')">
 <a name="none"></a>
 <div align="center">
   <h1>Internet Technologies for Business - Chapter 3</h1>
<!-- This links defines a menu like on which you click and new content replaces
the division
identified by ‘repme’ -->
```

```
   <a href="#none" onclick="makerequest('pgs/ajax-1.html');return false;">Page
1</a>
 | <a href="#none" onclick="makerequest('pgs/ajax-2.html');return false;">Page
2</a>
 | <a href="#none" onclick="makerequest('pgs/ajax-3.html');return false;">Page
3</a>
   <p></p>
   <div id="repme" name="repme" style="width: 99.9%">This content will be
replaced by the requested page</div>
 </div>
</body>
</html>
```

Figure 4. 11 The source code corresponding to the page in figure 4.10

4.4 High Level Languages based Architecture

In this architecture a new level N5 introduced (figure 4.12) where the communication client-server take place, in a real client-server relationship, and producing a significant reduction of data traffic (comparing with CGI). The most used high level languages represented by Java and XML.

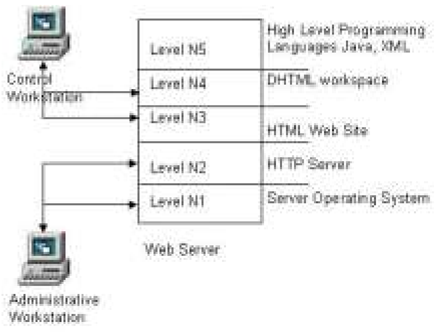


Figure 4.12 High level languages based architecture

4.4.1 Java

Java is a full programming language (in the same way C or C++ are) that offers the same services like other programming languages. His main advantage is represented by portability (the independence relatively to the used processor and operating system). The portability is ensured to both source and binary (executable) code. For the programs written in other programming languages even the source has portability the binary code generated by the compiler is specific to a processor type (figure 4.13).

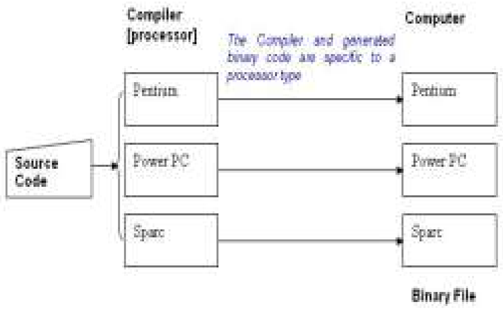


Figure 4.13 Programs "classic" compiled

The Java programs source composed by primary data of the same dimension does no matter the used development platform. The binary files are portable because they executed without requiring recompiling and this quality given by the pseudo-code (called *byte codes*) used to describe the procedures (figure 4.14). The pseudo-code is represented by a multitude of instructions closed to the machine language but without being dedicated to a specific processor type.

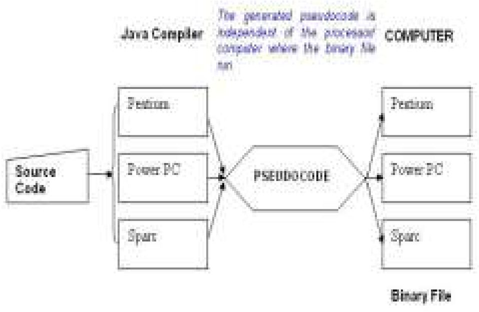


Figure 4.14 Compiled Java programs

The *bytes codes* interpreter is incorporated (embedded) in the compatible navigation programs/applications (browsers). Java language is an object oriented programming language and the creation of programs is modular and the modules reusable. Java has class libraries that supply the basic data types, realizes the system input/output operations, and other utility functions. Java programs, called applet, are executed (run) to the client machine (computer) and the server is involved only in the phase of initial loading of the applet (when it downloads to the client the requested page containing the applet). This behavior is very important in intranet applications.

By his conception and behavior (the language itself and the compiler) Java ensure an increased security. Java has a syntax of instructions and sentences as the C language with except of that Java do not uses pointers (the source of many problems) and registry data types but keeps the references to objects (realized by intermediate of pointers with an increased control).

The pseudo-code sequences are tested before execution to see if they do not violate the access constraints. A Java applet cannot read or write the local drives, cannot execute programs on the local computer and cannot connect to other web machines with except of the server from where taken. The compiler and interpreter verify the source code and the pseudo-code. A Java applet can be executed in multiple tiers but the management of the time allowed to each tier is his responsibility (and not of the host operating system as is happening for normal executables; the operating system allows the execution time to the applet and this, in turn, ensure the sharing of this by the execution tiers that it spawn).

Unlike an application, applets cannot be executed directly from the operating system and a well-designed applet can be invoked from many different applications. Web browsers, which are often equipped with Java virtual machines, can interpret applets from Web servers. Because applets are small in files size, cross-platform compatible, and highly secure (can't be used to access users' hard drives), they are ideal for small Internet applications accessible from a browser.

Generally is not necessary for end users to know Java to install applets in their pages. There are thousands of free applets available on the Internet for almost any purpose and most of them can be customized without programming. An applet can be embedded into a webpage and usually has several settings that allow personalize it. For instance, if you insert an applet that will work as a menu, you can specify which options should be in the menu, and which pages should be loaded upon click on an option. Since Java is a real programming language there aren't many limitations to it. Any program running on your computer could possibly have been made as an applet. When an applet is put on a page the applet and the HTML page the applet is embedded in must be saved on the server. When the page is loaded by a visitor the applet will be loaded and inserted on the page that embedded it. Applets have the file extension "class". Some applets consist of more than just one class file, and often other files need to be present for the applet to run (such as JPG or GIF images used by the applet). When we use existing applets we must check the documentation for the applet to see if we have all files for it to run, and before embedding an applet on a page, we need to upload the required files to the server.

Example:

```
<Html>
<Head>
<Title>A Java Example</Title>
</Head>

<Body>
This is a page with applet<br>
Below you see an applet<br>
<br>
<Applet Code="anapplet.class" width=200 Height=100>
</Applet>
</Body>
</Html>
```

The following attributes can be set for the <Applet> tag:

Attribute	Explanation	Example
Code	Name of class file	Code="anapplet.class"
Width=n	n=Width of applet	Width=200
Height=n	n=Height of applet	Height=100
Codebase	Library where the applet is stored. If the applet is in same directory as your page this can be omitted.	Codebase="applets/"
Alt="Text"	Text that will be shown in browsers where the ability to show applets has been turned off.	alt="Menu Applet"
Name=Name	Assigning a name to an applet can be used when applets should communicate with each other.	Name="starter"
Align= Left	Justifies the applet according to the text and images	Align=Right

Right **Top** **Texttop** **Middle** **Absmiddle** **Baseline** **Bottom** **Absbottom**	surrounding it.	
Vspace=n	Space over and under the applet.	Vspace=20
Hspace=n	Space to the left and right of applet.	Hspace=40

4.4.2 XML – eXtensible Markup Language

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

XML is not a language; it is actually a set of syntax rules for creating semantically rich markup languages in a particular domain. In other words, you *apply* XML to create new languages [DOS_03].

XML was developed by the XML Working Group (initially called SGML Editorial Review Board) from W3C (World Wide Web Consortium) in 1996. The initial specification of the language establishes the following objectives for this:

- XML must be directly usable in Internet;
- XML must support a variety of applications;
- XML must be compatible with SGML;
- The XML pages creation must be as simple as possible and rapidly done;
- XML may not contains facultative functions;
- The XML must have a high readable degree;
- The syntax must be formal and concise;
- The code concision is an element of little importance.

XML (Extensible Markup Language) has become far more than just a way of delimiting comma or tab separated files. XML has become an entire ecosystem of declarative languages and tools to process them. XML Schema are commonly used to efficiently validate form and structure. XML is now the most common way to express domain specific languages (DSLs). The new standard for HTML, XHTML, is a DSL expressed in XML. XML is a meta-language that allows defining specialized languages, extensible, that will be used for describing data structures specific to an applicative domain. XML is a markup language without having predefined tags (as HTML have): the author (user) defines (creates) all the markup tags it believe will be used, eventually qualified by attributes, and define the document structure (how the tags interacts with one another). For example, lets consider a Student record (table) having the fields outlined in figure 4.15).

Field Name	Data Type
StudID	Double
First_Name	String, 30 characters
Last_Name	String, 30 characters
Birth_Date	Short Date
Gender	String, 8 characters
Notes	String, 60 characters

Figure 4.15 Fields in Student record

The XML schema description of the Student is shown in figure 4.16.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema                    xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns:od="urn:schemas-microsoft-com:officedata">
<xsd:element name="dataroot">
<xsd:complexType>
<xsd:choice maxOccurs="unbounded">
<xsd:element ref="Student"/>
</xsd:choice>
</xsd:complexType>
</xsd:element>
<xsd:element name="Student">
<xsd:annotation>
<xsd:appinfo/>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:element        name="StudID"        minOccurs="0"        od:jetType="double"
od:sqlSType="float" type="xsd:double"/>
<xsd:element       name="First_Name"       minOccurs="0"       od:jetType="text"
od:sqlSType="nvarchar">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="30"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element       name="Last_Name"       minOccurs="0"       od:jetType="text"
od:sqlSType="nvarchar">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="30"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element      name="Birth_Date"      minOccurs="0"      od:jetType="datetime"
od:sqlSType="datetime" type="xsd:timeInstant"/>
<xsd:element         name="Gender"         minOccurs="0"         od:jetType="text"
od:sqlSType="nvarchar">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="8"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element         name="Notes"         minOccurs="0"         od:jetType="text"
```

```
od:sqlSType="nvarchar">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="60"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Figure 4.16 The XML schema description of Student record

The associated XML document (containing data) to the Student schema description is illustrated in figure 4.17.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmlns:xsi="http://www.w3.org/ 2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Student.xsd">
<Student>
<StudID>1001</StudID>
<First_Name>Ion</First_Name>
<Last_Name>IONESCU</Last_Name>
<Birth_Date>1986-09-26T00:00:00</Birth_Date>
<Gender>Male</Gender>
</Student>
<Student>
<StudID>1003</StudID>
<First_Name>Ana</First_Name>
<Last_Name>POPESCU</Last_Name>
<Birth_Date>1987-03-21T00:00:00</Birth_Date>
<Gender>Female</Gender>
</Student>
</dataroot>
```

Figure 4.17 The XML document containing data

Each application must give a semantic to markers by associating of a behavior to each element type. XML simplifies the language syntax and easiest considerably the implementation of different application. It offers the possibility for producing, changing, and processing of “well formed” (syntactically correct) documents without requiring from their part to include compulsory an explicit and strong declaration of their structure.

XML offers an elegant and efficient mechanism for defining namespaces allowing, in that way, applications to recognize (distinguish) them in a data multitude, to identify the structures that must be processed as such, ignored or processed in another manner. The namespace allows realizing the distinction between the elements created by different authors. Because authors can create custom elements, it leads to the possibility of naming collisions - different elements that each have the same name. An XML namespace is a collection of element and attribute names and has a unique name. The namespace can be used as namespace prefixes within a markup tag to indicate the associated namespace (for example

<student:first_name> or <person:first_name >, where *student* and *person* are namespace prefixes). The keyword *xmlns* is used to declare a default namespace for an XML document. The default namespace is declared within the root element.

XML, by simplifying the data exchange between heterogeneous applications and by unifying the data processing with the HTML documents, generated major changes in IT sectors such as electronic data exchange (EDI), "company's memory" and datamining.

4.4.2.1 Differences between XML, HTML, and SGML

XML - SGML Comparison. SGML was build for the purpose of storing documents in large centralized libraries while XML is designated to be used in a distributed environment and for storing documents while ensuring the interoperability by intermediate of data exchange. XML allows defining his elements and sub-elements. A document has a strong tree structure integrally accessible starting from his top element (root). This data structure type is more general and powerful than the relational data structure. A tree describes the composition of each element (his substructures); an element can be a hyperlink pointing any kind of informatic element located anywhere, or an element of a XML document (located in the same document or in another one).

XML - HTML Comparison. HTML defines only the page formatting of a document while XML defines the structure, content and semantic independent of the page formatting. The XML documents have a type definition given by DTD (Document Type Definition) while the HTML documents have not. The HTML grammar is fixed, standard defined and the keywords and structures are enclosed between tags. XML allows defining any structure that can be modeled as tree and the description is done by using the tags defined and wanted by user. In that way can be build standard data structures at profession level, for example, similarly to the definition of standard formats for EDI. The HTML documents have a sequential static structure with a heading and a body while the HML documents are hierarchies. Is another difference regarding the way the hyperlinks used in the advantage of XML.

4.4.2.2 XSL: the formatting language of XML

For displaying, printing, and voice synthesizing control XML provides two categories of style sheets:

- CSS, cascading style sheets used beginning with HTML 4 and applicable to XML too;
- XSL, style sheets based on an extensible language for style sheets - eXtensible Stylesheets Language, language representing developments SGML in accordance to ISO 10179. The XSL processor completes the technology of XML navigators (figure 4.1,85) and the XSL processing take place in two phases:

- in the first step is generated a new XML tree, based on the source document, expressed in XML or HTML;
- in the second phase the generated tree allows the fusion of XML data together with the static fragments of the document and a table of contents (that allows easy navigation) or a graphic element is generated.

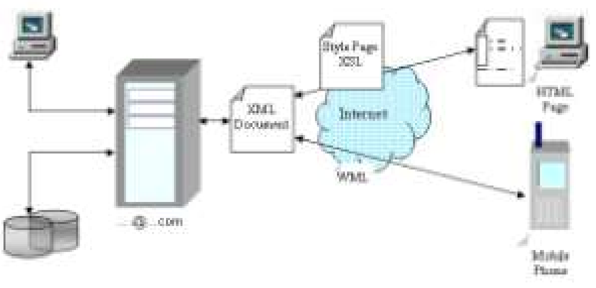


Figure 4.18 The steps of an XSL processing

The passing from a form to another one is realized simply by passing from a XSL page to another. The XSL processor can be used at server side in different ways.

Example:

The xsl page for Student (partly) is:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl" language="vbscript">
<xsl:template match="/">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=UTF-8" />
<TITLE>Student</TITLE>
<STYLE TYPE="text/css">
.Style0 { BORDER-STYLE: solid; COLOR: #000000; BACKGROUND-COLOR: #ffffff;
BORDER-WIDTH: 1px; BORDER-COLOR: #333399; TEXT-ALIGN: general;
WRITING-MODE: lr-tb; VISIBILITY: visible; FONT-WEIGHT: 400; FONT-SIZE: 9pt;
FONT-FAMILY: Tahoma; FONT-STYLE: normal; TEXT-DECORATION: none;
PADDING-TOP: 0in; PADDING-BOTTOM: 0in; PADDING-RIGHT: 0in; PADDING-
LEFT: 0in }
.Style1 { BORDER-STYLE: none; COLOR: #333399; BACKGROUND-COLOR:
transparent; BORDER-WIDTH: 1px; BORDER-COLOR: #000000; TEXT-ALIGN:
general; WRITING-MODE: lr-tb; VISIBILITY: visible; FONT-WEIGHT: 700; FONT-
SIZE: 9pt; FONT-FAMILY: Tahoma; FONT-STYLE: normal; TEXT-DECORATION:
none; PADDING-TOP: 0in; PADDING-BOTTOM: 0in; PADDING-RIGHT: 0in;
PADDING-LEFT: 0in }
</STYLE>
</HEAD>
<BODY link="#0000ff" vlink="#800080" style="BACKGROUND-
IMAGE:url('Images\Student.bmp'); BACKGROUND-POSITION: center center;
BACKGROUND-REPEAT: repeat">
<xsl:for-each select="/dataroot/Student">
<xsl:eval>AppendNodeIndex(me)</xsl:eval>
</xsl:for-each>
<xsl:for-each select="/dataroot/Student">
<xsl:eval>CacheCurrentNode(me)</xsl:eval>
<xsl:if expr="OnFirstNode">
<DIV style="BORDER-STYLE: none; WIDTH: 4.5416in; BACKGROUND-COLOR:
#ece9d8; VISIBILITY: visible; HEIGHT: 0in; POSITION: relative">
```

```
</DIV>
</xsl:if>
<DIV style="BORDER-STYLE: none; WIDTH: 4.5416in; BACKGROUND-COLOR:
#ece9d8; VISIBILITY: visible; HEIGHT: 1.6041in; POSITION: relative">
<SPAN class="Style0" style="TEXT-ALIGN: right; LEFT: 1.3333in; TOP: 0.0833in;
WIDTH: 1.6041in; HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
<xsl:eval no-entities="true">Format(GetValue("StudID", 5),"" ,"")</xsl:eval>
</SPAN>
<SPAN class="Style1" style="LEFT: 0.0416in; TOP: 0.0833in; WIDTH: 1.2916in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
StudID
</SPAN>
<SPAN class="Style0" style="LEFT: 1.3333in; TOP: 0.3333in; WIDTH: 1.6041in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
<xsl:eval no-entities="true">Format(GetValue("First_Name", 202),""
,"")</xsl:eval>
</SPAN>
<SPAN class="Style1" style="LEFT: 0.0416in; TOP: 0.3333in; WIDTH: 1.2916in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
First Name
</SPAN>
<SPAN class="Style0" style="LEFT: 1.3333in; TOP: 0.5833in; WIDTH: 1.6041in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
<xsl:eval no-entities="true">Format(GetValue("Last_Name", 202),""
,"")</xsl:eval>
</SPAN>
<SPAN class="Style1" style="LEFT: 0.0416in; TOP: 0.5833in; WIDTH: 1.2916in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
Last Name
</SPAN>
<SPAN class="Style0" style="TEXT-ALIGN: right; LEFT: 1.3333in; TOP: 0.8333in;
WIDTH: 0.7187in; HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
<xsl:eval no-entities="true">Format(GetValue("Birth_Date", 7), "Short Date",
"")</xsl:eval>
</SPAN>
<SPAN class="Style1" style="LEFT: 0.0416in; TOP: 0.8333in; WIDTH: 1.2916in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
Birth Date
</SPAN>
<SPAN class="Style0" style="LEFT: 1.3333in; TOP: 1.0833in; WIDTH: 0.6458in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
<xsl:eval no-entities="true">Format(GetValue("Gender", 202),"" ,"")</xsl:eval>
</SPAN>
<SPAN class="Style1" style="LEFT: 0.0416in; TOP: 1.0833in; WIDTH: 1.2916in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
Gender
</SPAN>
<SPAN class="Style0" style="LEFT: 1.3333in; TOP: 1.3333in; WIDTH: 3.1666in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
<xsl:eval no-entities="true">Format(GetValue("Notes", 202),"" ,"")</xsl:eval>
</SPAN>
<SPAN class="Style1" style="LEFT: 0.0416in; TOP: 1.3333in; WIDTH: 1.2916in;
HEIGHT: 0.1875in; OVERFLOW: hidden; POSITION: absolute">
Notes
</SPAN>
</DIV>
<xsl:if expr="OnLastNode">
<DIV style="BORDER-STYLE: none; WIDTH: 4.5416in; BACKGROUND-COLOR:
```

```
#ece9d8; VISIBILITY: visible; HEIGHT: 0in; POSITION: relative">
</DIV>
</xsl:if>
<xsl:eval>NextNode()</xsl:eval>
</xsl:for-each>
</BODY>
</HTML>
<xsl:script>
<![CDATA[
 …………
```

4.4.2.3 XQL – the extended query language

XQL (eXtended Query Language) allows searching in the XML tree and is realized as an extension of XLS for pattern matching that allows the description of searching criteria (for example, *book/author* means searching of elements *author* contained by the elements *book*). XQL allows hyperlinks to all nodes that satisfy some criteria and the string defining the search can be embedded in URL. XQL is declarative as SQL is the difference is that his result is a tree or an XML graph

4.4.2.4 Database Links

Structured Data Exchange. The first XML specifications in 1998 almost aim the definition of a specialized language for describing structured data with the scope of realizing the data exchange between applications in an open system network environment. In that sense XML was defined as a language for serializing the imported/exported records in/from relational databases. The usage of XML in that scope argued by:

- standardization of syntactical analyzers (efficient and free);
- the representation of relational schemas without data loss in XML is trivial;
- the language coupled with the style sheets (XSL) well matches to data fusion for heterogeneous schemas and sources and schema transformation.

This working manner allows (for example, in an e-commerce application starting with a HTTP request) bringing out from the database the information regarding a product (availability, price etc), coding in XML and fusion with the static part (product imagine, description etc) for generating “by fly” a personalized commercial offer.
Storage of XML documents in databases. XML is adapted to the storage of any kind of documents such as illustrated technical manuals, e-mail, programs, reports etc. The XML stored data are independent on the hardware, software, and used access methods, on the programming languages and the page layouts and formatting, being in fact a universal standard for storage.
Document Object Model (DOM). The XML/HTML navigator implements an application interface API that offers a programmable access to displayed data. The standard W3C defines an object-oriented API allowing an application program to access the tree formed by an XML object. This API can be implemented in any

object oriented programming language (generally in Java). The XML document can contains other forms (such as, for example, a menu form) and the code defining the changes to be done dynamically in the document as reaction to user actions and filled values (not necessarily typed). The document becomes in that way interactive and can be changed (modified), in content and presentation, without server interaction.

The figure 4.19 shows an example of using the document object and his sub-objects and properties of that in code (you can copy and paste in the HTML view in a new page in FrontPage or MS Script Editor or Netscape Composer etc and preview in browser to see the result). The script inside changes the document background when the user clicks somewhere on his surface.

```
<HTML>
<HEAD>
<TITLE>Color Switch</TITLE>
<META NAME="GENERATOR" Content="Microsoft Visual Studio">
<META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-8">
<script type="text/javascript">
function ChangeColor()
{
        /* to the variable curBGColor assigned the current background color
           that originally is set to red (#ff1001)
        */
        curBgColor = document.body.style.background;
        /* the switch sequence */
        if (curBgColor == "#ff1001" || curBgColor == "")
                { document.body.style.background="yellow";}
          else
                { document.body.style.background="#ff1001";
        }
}
</script>
</HEAD>
<BODY onclick="ChangeColor()" bgcolor="#ff1001">
  <p>Click anywhere on this document to switch the background color from red to
yellow!
  </p>
</BODY>
</HTML>
```

Figure 4.19 An example of using document object in a JavaScript inside of a HTML page

Figure 4.20 illustrates the usage of VBScript together with the DOM model to realize many operations:

- the function validateValues(), is called by the procedure Coresp_Temp() to validate the form fields values passed in by the user by checking if they are numbers. If not numbers an error message returned to the caller and displayed in a message box and the processing stops. The fields, such as minTemp.value or maxTemp for example, are objects whose property called value stores the text box content;
- if all typed values are numbers the Coresp_Temp() procedure produces the HTML tags for a table containing the correspondence value between Celsius an Fahrenheit degrees. First the heading of table stored in the

memory variable called repx and after that a cycle of transformation started from minTemp.value to maxTemp.value that transforms values at each hop given by stepTemp.value . Each pair Fahrenheit value and corresponding Celsius value will generate a new row in the table **"<tr><td width=120px>" & Right(Space(24) & round(fahrenheit,2),12) & "</td><td width=120px>" & Right(Space(24) & round(celsius,2),16) & "</td></tr>"**

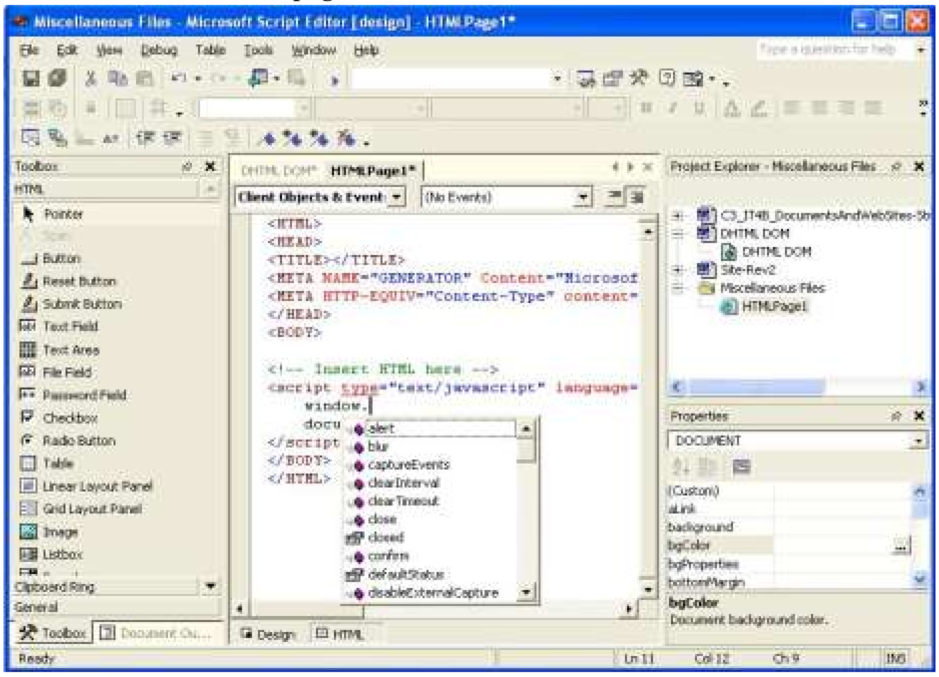
- after the conversion cycle ends the generated table closed by his end tag and the function displays the table (stored in the memory variable repx) by replacing the division called “replaceMe” in the web page **document.getElementById("replaceMe").innerHTML=repx**.
- the button called ClearPage clears the forms fields by assigning the empty string to the value property of each field. The content generated in the page is erased too by the command **document.getElementById("replaceMe").innerHTML=""**.

```
<html>
<head>
<title>VBScript solution</title>
<script type="text/vbscript" language="vbscript">
<!--
 function validateValues(min,max,pas)
  dim ret
  ret="Yes"
  if isnumeric(trim(minTemp.value)) then
    min=trim(minTemp.value)
   else
    if ret="Yes" then ret=" "
    ret=ret & "<<From>>"
  end if
  if isnumeric(trim(maxTemp.value)) then
    max=trim(maxTemp.value)
   else
    if ret="Yes" then ret=" "
    ret=ret & "<<To>>"
  end if
  if isnumeric(trim(stepTemp.value)) then
    pas=trim(stepTemp.value)
   else
    if ret="Yes" then ret=" "
    ret=ret & "<<Step>>"
  end if
  validateValues=ret
 end function
 Sub Coresp_Temp()
    Dim min,max,pas,fahrenheit, celsius,conr, repx
    ' Computation of the correspondence Co- Fo
```

```
      if (validateValues(min,max,pas))<>"Yes" Then
         msgbox "Err. 01. The value you typed in " &
validateValues(min,max,pas) & " box is not a number ? Correct
them and press again."
         exit sub
      end if
      fahrenheit=0 :  celsius=0
      repx="<table border=1 style='text-align:right'><tr><td
width=120px>Fahrenheit  </td><td width=120px> Celsius
</td></tr>"
      For fahrenheit = min To max Step pas
         celsius = (5/9)*(fahrenheit-32)
         repx=repx & "<tr><td width=120px>" & Right(Space(24) &
round(fahrenheit,2),12) & _
                  "</td><td width=120px>" & Right(Space(24) &
round(celsius,2),16) &  "</td></tr>"
      Next
      repx=repx & "</table>"
  document.getElementById("replaceMe").innerHTML=repx
  End Sub
  function ClearFields()
    minTemp.value=""
    maxTemp.value=""
    stepTemp.value=""
    document.getElementById("replaceMe").innerHTML=""
  end function
-->
</script>
</head>
<body>
<h1>VBScript based solution.&nbsp;</h1><br/>
Type the values you want to compute the correspondence and then
press the Show button<br/>
<div style="border:1px solid #c0c0c0">
From:<INPUT type="text" id="minTemp" name="minTemp"
size="4" align="right" value="0" >
&nbsp;&nbsp;&nbsp;To:<INPUT type="text" id="maxTemp"
name="maxTemp" size="5" align="right" value="300">
&nbsp;&nbsp;&nbsp;Step:<INPUT type="text" id="stepTemp"
name="stepTemp" size="5" align="right" value="20">
<br/>
<br/><INPUT type="button" value="Show" ID="vbcompdat"
onclick="Coresp_Temp()">
<INPUT type="button" value="ClearPage" ID="vbclr"
onclick="ClearFields()">
</div>
```

```
<p></p>
 <div id="replaceMe"></div>
</body>
</html>
```

Figure 4.21 An example of using document object model in a VBScript inside of a HTML page



```
<HTML>
<HEAD>
<TITLE></TITLE>
<META NAME="GENERATOR" Content="Microsof
<META HTTP-EQUIV="Content-Type" content=
</HEAD>
<BODY>

<!-- Insert HTML here -->
<script type="text/javascript" language=
    window.
    docu
</script
</BODY>
</HTML>
```

Figure 4. 20 The access to DOM objects in Microsoft Script Editor

Figure 4.21 shows how document object and his component objects are accessed and manipulated in Microsoft Script Editor.

Access to DHTML documents. For an object oriented programming language the document is an object composed by other objects. When API accessed by the XML interpreter it exists a gateway object that allows access another object, the document, whose elements are objects too. In that conditions any conversion of a XML document can be done under the control of a Java or Visual Basic program (for example).

4.5 Dynamic Pages Architecture

The e-commerce applications must interfaced with the enterprise's databases for data retrieval purposes. The programs for ensuring the link between the web server and databases are written in different languages such as Python, Perl, PHP, ASP, C++, etc.

The generic name that defines such links is CGI – Common Gateway Interface – described in the chapter 1.2.6 (figure 4.22, 4.23, and 4.24). These applications allows, depending on their objectives and characteristics, generating dynamical pages. Figure 4.22 shows the content of a PHP script stored at server side that access a database table to check if a username/password combination exists or not and to allow or deny the access. Figure 4.23 shows what is delivered to the user by the server when this one access the page.

```
<?php
session_start();
// Check if he wants to login:
if (!empty($_POST[username]))
{         require_once("connect.php");
          $query = mysql_query("SELECT * FROM members
                                        WHERE username = '$_POST[username]'
                                        AND password = '$_POST[password]'")
          or die ("Error - Couldn't login user $_POST[username].");
          $row = mysql_fetch_array($query)
          or die ("Error - Couldn't login user $_POST[username].");
          if (!empty($row[username]))        {
                    $_SESSION[username] = $row[username];
                    echo "Welcome $_POST[username]! You've been successfully logged
in.";
                    exit();    }
          else // bad info.
          {
          echo "Error - Couldn't login user $_POST[username].<br /><br />
                              Please try again.";
                    exit();
}}
?>
<html>
  <head>
          <title>Login</title>
  </head>
  <body>
    <form action="login.php" method="post">
      <table border="1" cellpadding="3" cellspacing="1" style="width: 37%">
          <tr>
            <td style="width: 93%; text-align:center; height: 1.2em;"><h2 style="font-
size: small; height: 10px; margin-bottom:1px">Login</h2></td>
          </tr>
          <tr>
            <td style="width: 93%; font-size: small; height: 1em;"><label>Username:
<input type="text" name="username" size="29" value="<? echo $_POST[username];
?>"></label></td>
          </tr>
          <tr>
            <td style="width: 93%; font-size: small; height: 1em;"><label>Password:
<input type="password" name="password" size="25" value=""></label></td>
          </tr>
          <tr>
            <td style="width: 93%"><input type="submit" value="Login"></td>
          </tr>
    </table>
    </form>
  </body>
```

```
</html>
```

Figure 4. 22 The page content at server side containing access to database tables

```
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <form action="login.php" method="post">
      <table border="1" cellpadding="3" cellspacing="1" style="width: 37%">
        <tr>
          <td style="width: 93%; text-align:center; height: 1.2em;"><h2 style="font-
size: small; height: 10px; margin-bottom:1px">Login</h2></td>         </tr>
        <tr>
          <td style="width: 93%; font-size: small; height: 1em;"><label>Username:
<input type="text" name="username" size="29" value=""></label></td>         </tr>
        <tr>
          <td style="width: 93%; font-size: small; height: 1em;"><label>Password:
<input type="password" name="password" size="25" value=""></label></td>
</tr>
        <tr>
          <td style="width: 93%"><input type="submit" value="Login"></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

Figure 4. 23 The page content at client side as generated and downloaded to the user

The dynamic page is not stored to the server as the static one is. She is generated (manufactured) according to the client request and the information content is prepared to the web server (an image of that) and send to client that can read them on his computer and used browser (figure 4.24). In figure 4.24 is represented a functional architecture of the site containing dynamical pages and figure 4.25 the synthetic representation of the process of generating pages.

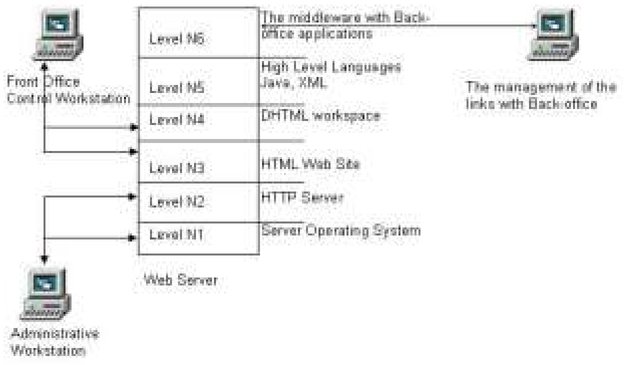

Front Office Control Workstation
Level N6
Level N5
Level N4
Level N3
Level N2
Level N1
The middleware with Back-office applications
High Level Languages Java, XML
DHTML workspace
HTML Web Site
HTTP Server
Server Operating System
The management of the links with Back-office
Web Server
Administrative Workstation

Figure 4.24 Functional architecture for dynamic pages

The communication process involved by the new level N6 introduced by that architecture take place as indicated in the chapters 2.2.6, paragraph

Client/Server Technology, and chapter 3.1 paragraphs Intranet, Extranet, and The e-business – e-commerce relationships.

SSI (Server-Side Include). The server can generate dynamic (“by fly”) the HTML file, that it sends to client for displaying, in accordance with the data send/requested by client with SSI. SSI is a language that can create dynamically the page content send to client. Not all Web servers offer support for SSI. How SSI works? For example, for including a header (stored in a file named *header.txt*) in every page that will be displayed by the browser we must write in the HTML file, to the beginning of the body:

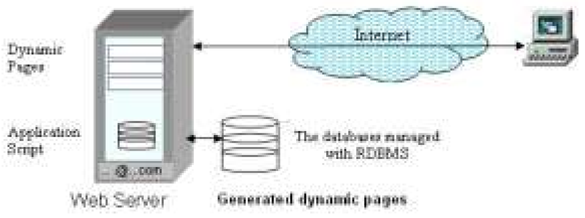


Figure 4.25 The generation of a dynamic page

```
<pre>
<!--#include file="header.txt" -->
</pre>
```

When a client request that page the server processes first the command and send to client the HTML file containing the content of the file “header.txt” substituted to the command (the header.txt content is substituted to the comment). The SSI commands are invisible to the client this one receiving (his browser) only the result of the processing of that commands.

ASP - Microsoft ® Active Server Pages. ASP is a Web environment for developing server scripts and used for dynamic execution of Web server interactive applications. As idea ASP is similar to SSI, but is more complex. ASP is

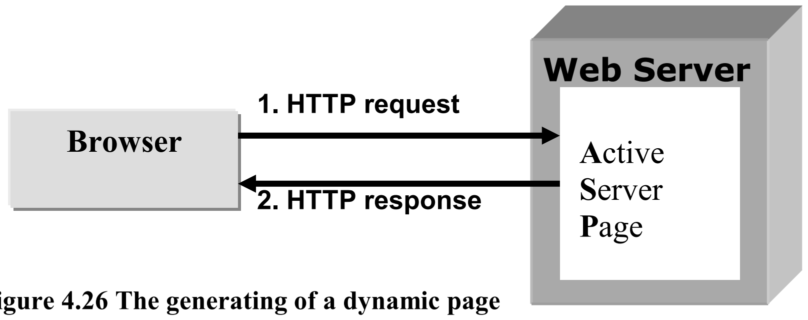


Figure 4.26 The generating of a dynamic page

characteristic to Microsoft Web servers and is nothing than a way to process the scripts at server side. The difference is that the client side scripts are send to client (usually embedded in the HTML documents) and this one processes locally the script while the ASP is processed by the server and the result is included and send in the HTML file to client (figure 4.26).

PHP. Is a language similarly to C and used in the same way as ASP. Is a development of Apache and is completely free. For PHP it exist interpreters on a variety of Web servers (including Microsoft) and can run under a variety of operating systems. As a rule the files containing PHP have the extension ".php". The PHP code is enclosed in special start (<?php) and end (?>) tags that allow you to jump into and out of "PHP mode" (figure 4.22).
Example:

```
</html>
<?php
   echo "Text afisat de script PHP.<br>";
?>
</body>
</html>
```

PHP [OLS07], which stands for "PHP: Hypertext Preprocessor" is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but you can do much more with PHP.

There are three main areas where PHP scripts are used:

- Server-side scripting, this is the most traditional and main target field for PHP and can do anything any other CGI program can do (such as collect form data, generate dynamic page content, send and receive cookies etc). This type of usage requires three things to make this work: a PHP parser (CGI or server module), a web server and a web browser.
- Command line scripting, the PHP script can be run without any server or browser but PHP parser;
- Writing desktop applications, even PHP is not the very best language to create a desktop application with a spectacular graphical user interface can be used in combination with PHP-GTK to write cross-platform client application programs.

PHP can be used on all major operating systems (including Linux, many Unix variants, Microsoft Windows, Mac OS X, RISC OS etc) has support for most of the web servers today (including Apache, Microsoft Internet Information Server, Personal Web Server, Netscape and iPlanet servers, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, and many others).

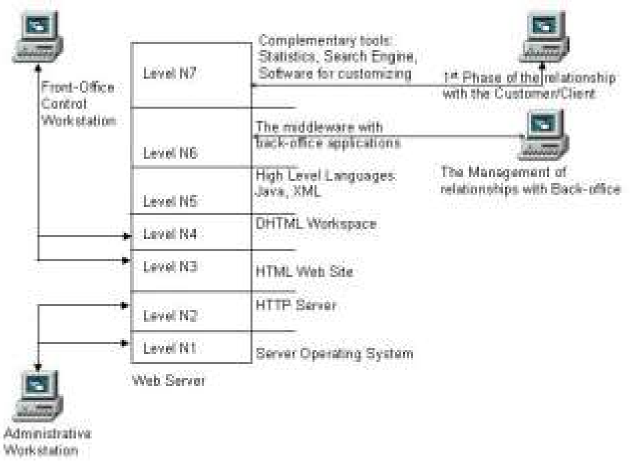
PHP scripts can be realized using procedural programming or object oriented programming, or a mixture of them. PHP has the ability to output different kinds of documents such as HTML, XHTML, XML, images, PDF files and even

Flash movies (using *libswf* and *Ming*) generated on the fly. PHP can auto-generate these files, and save them in the file system, instead of printing it out, forming a server-side cache for your dynamic content.

A strongest and significant feature of PHP is its support for writing a database-enabled web page for a wide range of databases: Adabas D, InterBase, Postgre, SQL, dBase, FrontBase, SQLite, Empress, mSQL, Solid, FilePro (read-only), Direct MS-SQL, Sybase, Hyperwave, MySQL, Velocis, IBM DB2, ODBC, Unix dbm, Informix, Oracle (OCI7 and OCI8), Ingres, Ovrimos.

4.6 Advanced Management Architecture

For better usage of information produced by the site or to offer complementary services to their clients (figure 4.27) we must provide the site with specific utilities (applications or programs) that answers to that requirements, such as, tools for site activity analysis or those for obtaining statistic information.



Front-Office Control Workstation
Level N7
Level N6
Level N5
Level N4
Level N3
Level N2
Level N1
Web Server
Complementary tools: Statistics, Search Engine, Software for customizing
1st Phase of the relationship with the Customer/Client
The middleware with back-office applications
High Level Languages: Java, XML
The Management of relationships with Back-office
DHTML Workspace
HTML Web Site
HTTP Server
Server Operating System
Administrative Workstation

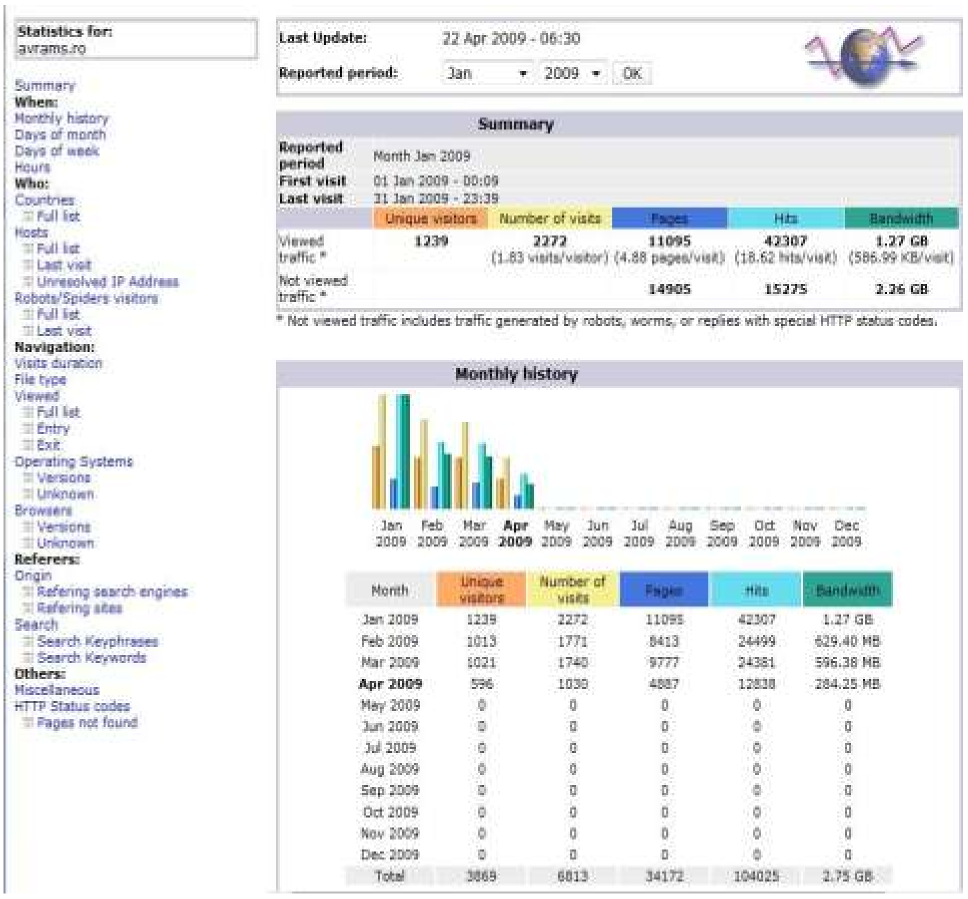
Figure 4.27 The Advanced Management Functional Architecture

4.6.1 Statistic utilities

All statistic information are collected and analyzed in a form of database schemas. Analytics is software that generates metrics for example about how many times files are accessed, how many unique IP addresses access the site, how many pages are served, and so on. Analytics can calculate the most popular pages, how long the typical person stays on the typical page, the percentage of people who "bounce" or leave the site from a particular page, and thus the percentage of people who explore the site more deeply.

In the following paragraphs two examples of analytics packages introduced: AWStats as an open source site side solution and Google Analytics as

AWStats. AWStats (Advanced Web Statistics) is an open source log analyzer written in Perl that can use a variety of log formats and runs on a variety of operating systems. AWStats is primarily a site statistics program, residing at host server side, a log analyzer, that counts more than it calculates. Figure 4.28 shows the first screen of the Web pages report given by AWStats where the left panel is a menu and list of topics reported.



Last Update:	22 Apr 2009 - 06:30
Reported period:	Jan 2009 OK

Summary

Reported period	Month Jan 2009
First visit	01 Jan 2009 - 00:09
Last visit	31 Jan 2009 - 23:39

	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Viewed traffic *	1239	2272 (1.83 visits/visitor)	11095 (4.88 pages/visit)	42307 (18.62 hits/visit)	1.27 GB (586.99 KB/visit)
Not viewed traffic *			14905	15275	2.26 GB

* Not viewed traffic includes traffic generated by robots, worms, or replies with special HTTP status codes.

Monthly history

Month	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Jan 2009	1239	2272	11095	42307	1.27 GB
Feb 2009	1013	1771	8413	24499	629.40 MB
Mar 2009	1021	1740	9777	24381	596.38 MB
Apr 2009	596	1030	4887	12838	284.25 MB
May 2009	0	0	0	0	0
Jun 2009	0	0	0	0	0
Jul 2009	0	0	0	0	0
Aug 2009	0	0	0	0	0
Sep 2009	0	0	0	0	0
Oct 2009	0	0	0	0	0
Nov 2009	0	0	0	0	0
Dec 2009	0	0	0	0	0
Total	3869	6813	34172	104025	2.75 GB

Figure 4. 28 AWStats reports (main screen)

Google Analytics. "Google Analytics (figure 4.29) helps you find out what keywords attract your most desirable prospects, what advertising copy pulled the most responses, and what landing pages and content make the most money for you (http://www.google.com/analytics/feature_benefits.html). "

Google Analytics uses a snippet of JavaScript code to track the traffic on your web site as the following one:

```
<script            src="http://www.google-analytics.com/urchin.js"
type="text/javascript">
</script>
```

```
<script type="text/javascript">
  <!--
    _uacct = "UA-1653633-1"; // domain unique code given by Google
when registered
    urchinTracker(); // call the tracker for the specified domain
  //-->
</script>
```

The code introduced in the body section of every page you want monitorize. The left panel in the figure 4.29 is a navigational menu to a variety of reports included in one of the fourth broad categories: Visitors, Trafic Sources, Content, and Goals.

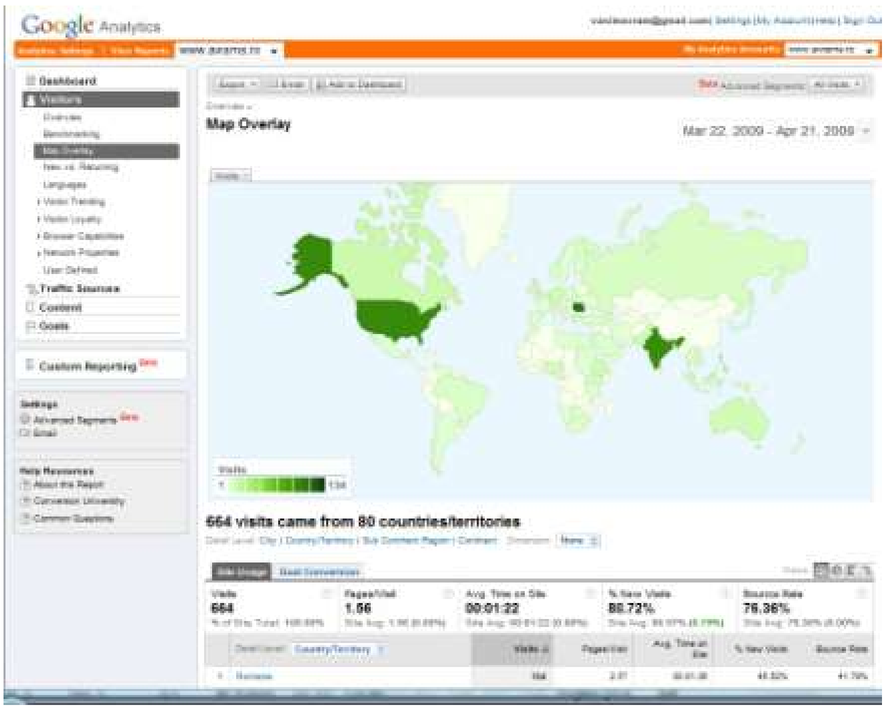


Figure 4. 29 The Google Analytics reports

4.6.2 Cookie

Cookie offers complementary means for identifying the visitors/users and distinguished of other solutions in three strong points:

- a cookie is stored always on the user's computer;
- a cookie is accessible uniquely to the server that generating it;
- a cookie is editable only in the moment the user visits the site that generated this.

Being build as a character string a cookie represents a powerful tool for developing web sites. The creation of a cookie can be done in two ways (figure 4.30):

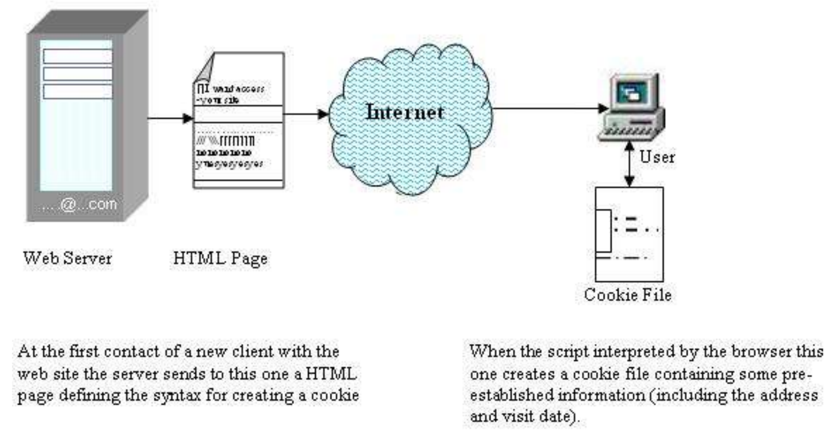

Internet
....@...com
User
Web Server
HTML Page
Cookie File
At the first contact of a new client with the web site the server sends to this one a HTML page defining the syntax for creating a cookie
When the script interpreted by the browser this one creates a cookie file containing some pre-established information (including the address and visit date).

Figure 4.30 The implementation of a cookie

1. by transmitting from server to navigator the order for creation (instruction set_cokie);
2. by executing the instructions for creating cookies on the client computer by intermediate of one of programming languages whose instructions embeds in HTML pages.

In both cases the starting of the command for generating cookie is the corollary of the client request of/access to a HTML page. Figure 4.31 shows the evolution way of a cookie.

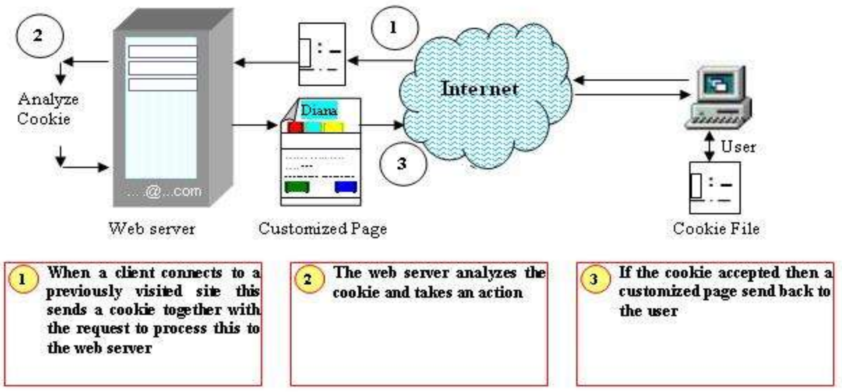

1
2
3
Analyze Cookie
Internet
Diana
....@...com
User
Web server
Customized Page
Cookie File
When a client connects to a previously visited site this sends a cookie together with the request to process this to the web server
The web server analyzes the cookie and takes an action
If the cookie accepted then a customized page send back to the user

Figure 4.31 The evolution of a cookie

A cookie can be used for different purposes such as:

1) Restricting access to some web pages (for services that requires subscription, for example). In that case the access requires as a rule the pair username/password given to the user when registering in the site. The script for this case can transmit a cookie to client containing his name or an access code to the page. For this purpose (restricted access) the webmaster must write every page having restricted access as a CGI script and later on must verify

dynamically the existence of cookie (for limit the duplicates of this and repeating the authentication by username/password);

2) Building purposefulness forms. Some sites presents to every visit the same form that requires the same information from the client that can embarrassing this and can produce redundant information. Solving of that problem can be done by a CGI script that searches the presence of a generic cookie created after the user fills the minimal set of acceptable data in the registering form. If the cookie present the registration page will not be created and the access is given to the next page. If the cookie is missing then the registering form will be generated.
3) Web pages personalization. This is one of most judicious usage of cookies. Many sites (generally informational) allow users to configure/customize the home page. By intermediate of the associated (assigned) cookie it recuperates the personalization at any login.
4) Piloting a virtual kadi (cart). The absence of the management of a memory (primary or secondary) by the HTML protocol makes this inadequate for e-commerce. By the appearance of cookies becomes possible the creation of a virtual Kadi that follows the client in his virtual travel. The cookie memorizes the integrality of buying session that allows displaying on each new page the selected articles (name, quantity, price etc). The concurrent solutions developed in JavaScript do not eliminates cookie because by intermediate of this one the nominative information shared with the server do not requires retyping to the next login.

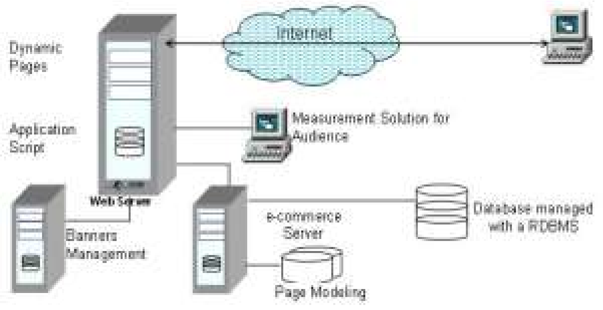


Figure 4.32 Network traffic analysis tools

4.6.3 Network traffic analysis

The quality of services offered by a selling site can be measured by intermediate of traffic analysis tools (figure 4.32).

4.7 Multi-tier (three tiers) Architecture

In the integration process of company's e-commerce site with his current activities appears the necessity, in mean or long term, to realize the link with the existing management informatic systems (to integrate the site with these ones).

The manufacturing process, enterprise management, and the ERP (Enterprise Resource Planning) utilities are permanently linked to the selling architecture (if not embedded in this one). In the case in which the organization disposes of heterogeneous application to automate his activities these must be interfaced with the selling site. The first function that must be accomplished by the application server is to ensure that interfacing and integration (figure 4.33).

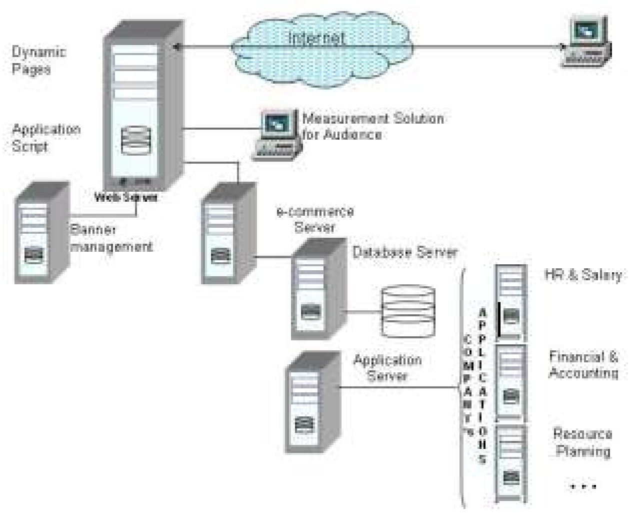


Figure 4.33 Three-tier Architecture

4.7.1 Client-Server Infrastructure

The client-server infrastructure allows generally the support for interfacing and integrating (the description of client-server technology introduced in chapter 1.1).

The client-server technology can be categorized in the following architectures:

- Client-server 1st generation: this architecture consists of a client that uniquely manages the presentation layer and a server running the entirely application;
- Client-server 2nd generation: this architecture developed on Windows platform and PC. The basic idea was that of using the processing capabilities of PC (at client side) and the server manages the database access by the SQL queries of the client.
- Client-server 3rd generation: this generation was born due to Internet technology. It consists of a client managing the presentation layer, a server layer for application whose task represented by applications, and a third layer containing the database.

The client-server architecture has three simple layers:

- Client workstation;
- Communication intermediaries;
- Access to standardized services.

The client workstations, provided with browsers from which the client components, such as Java applets and ActiveX controls, are automatically updated.

This solves the software distribution problem and the software versions management to the client workstation. More than the application logic migrates to server side authorizing, in that way, an efficient control of his deployment.

4.7.2 Application Server

The main function of the application server is to deserve the information system applications. The enterprise's information system concentrates in three poles (figure 4.34) – human resources organization, information and technology – that allows a centralization of enterprise's intelligence so that we can qualify this assembly be the enterprise "know how".

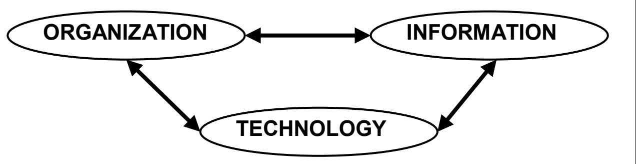


Figure 4.34 Organization of Information System

In order to develop this architecture the enterprises must choose between the "as it is possible" in-house development and buying an offer "ready to use" from third parties, generally, less flexible. Now is possible to realize a mixed approach: the development on the basis of existing components of J2EE (Java 2 Enterprise Edition) in association with EJB (Enterprise Java Beans) components, as an emergency solution.

The enterprise's intelligence and the information system can be divided in three layers:

- information capital, that is a prime material of the enterprise;
- processes, that models the enterprise activity;
- applications, that constitutes the functional part, the graphic interface, and processing processes.

The information base is shared between all enterprise processing processes, and each processing process is sharable between many applications. In that way, by preserving this architecture until the implementation, we can obtain a maximal rate of usage of the information system components.

The logical structures of the architecture of information system can be represented by using a layered approach as in figure 4.35. In that logical structure:

- the application layer decomposed in two (sub)layers presentation and coordination;
- the layer information database composed by the domain layer and persistency layer;

In that way the logical architecture composed by five layers:

1. **Presentation**: manages the visual domain;
2. **Coordination**: his tasks are to invoke the processes of the inferior layer, to manage the user workspace and working session;

3. **Services**: his main purpose is to supply services specific to the domain activities. The main tasks of this layer represented by the component distribution (deployment), transaction control, and security;
4. **Domain**: is concerned on the enterprise activities common to all applications and his main task is to guarantee the domain model by applying the processing rules;
5. **Persistency:** the last layer in the architecture but the most important by that all persistency of the system is reported to it. At this layer we found basic functionalities that allow creation, updating, searching, retrieval, and deletion of components of the entity processing processes.

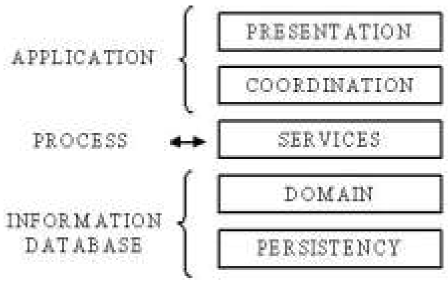


Figure 4.35 The logical layers of the information system

5 DEFINING AND STRUCTURING WEB PAGES USING HTML

5.1 HTML – An introduction

HTML, acronym for HyperText Markup Language, represents the most commonly used language for presenting information over the Internet. It is not a programming language, because it does not allow the implementation of algorithms. It is a description language which uses different elements to format, arrange and place text, images and other objects on a webpage. HTML code can be written using any simple text editor, like Notepad, in the form of HTML elements.

The HTML elements are defined using tags represented through predefined keywords, stating the name of the HTML element, surrounded by angle brackets. Most of the HTML elements have both a start tag, represented by the name of the element surrounded by angle brackets (`<html>, <head>, <b>`), and an end tag, represented by the name of the element preceded by symbol `/`, surrounded by angle brackets (`</html>, </head>, </b>`). The start tag sets the place where the html element begins and the end tag sets the place where the html element ends. The content of the HTML element, between the start tag and the end tag, might be nothing or it could contain plain text and other HTML elements, called children. Usually, the children HTML element tags are defined within the parent HTML element tags, meaning that child start tag follows parent start tag and child end tag is written before parent end tag:

```
    <parent_start_tag> [HTML] <child_start_tag> [HTML]
<child_end_tag> [HTML] </parent_end_tag>
```

[HTML] is optional HTML code which could contain simple text and/or other HTML elements. When writing HTML, a special attention should be paid to respecting this rule and exceptions must be treated very carefully.

There are also HTML elements which do not require an end tag. For instance, when you insert a horizontal line across the page, you cannot specify both where this line should start and where it should end because it is drawn, by default, from the left edge to the right edge. This way, only the opening tag `<hr />` is used for inserting the horizontal line. These are called empty HTML elements and they are closed in the start tag. The same thing applies for other HTML elements like new line `<br />` or image `<img ... />`. HTML is case insensitive which means that there is no difference between writing tags with capital letters, like <HTML>, <BODY>, <BR /> or with small letters like <html>, <body>, <br />. The browsers will interpret them the same way.

Most of the HTML elements can be customized by setting values to their attributes. In order to modify an HTML element, the name of the attribute and the value it takes are written in the start tag, like name_of_attribute = "value". More attributes can be inserted in the opening tag of one HTML elements and the changes they provide are applied in the order they are written. For instance, in order to insert a centered red colored line with a length of 200 pixels, the three attributes along with their values must be written inside <hr> tag, separated by at least a simple space:

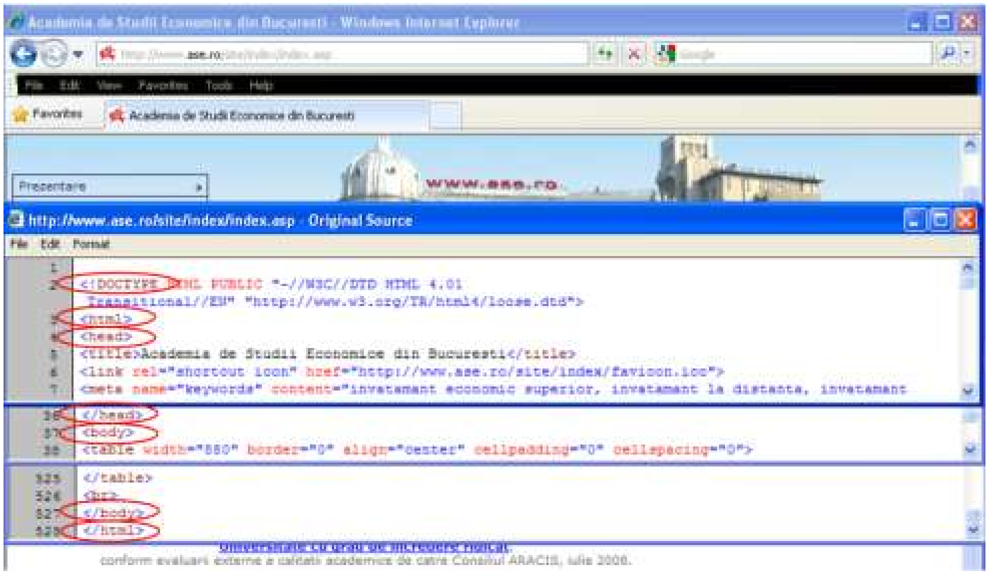
```
<hr align = "center" color = "red" width = "200" />
```

The values of the attributes should be written between double quotes or single quotes. If the value itself contains a word written between double quotes then it should be enclosed between single quotes. Though attribute names and attribute values are case insensitive, the World Wide Web Consortium (W3C) recommends them to be written with lowercase in their HTML 4 recommendation.

There are some attributes that can be applied to most HTML elements with very few exceptions. These attributes are divided into *core attributes, internationalization attributes* and *scripting events*:

- core attributes:
  - id – the value it takes uniquely identifies the HTML element it belongs to, throughout the entire webpage. The value, which is case sensitive, should begin with a letter and might contain letters(a-z, A-Z), digits (0-9), hyphens ("-"), underscores ("_"), colons (":") and periods (".").
  - class – specifies that the HTML element is member of one or more classes. Unlike the case of the ID attribute, more HTML elements may belong to the same class and one HTML element may belong to several classes. The value of the class attribute is case-sensitive.
  - style – allows the specification of style rules for the HTML element it belongs to.
  - title – provides a title to the HTML element, which can be viewed as a tooltip when the user visiting the webpage goes with the mouse pointer over the HTML element.
- internationalization attributes
  - lang – specifies the language in which the values of the attributes are written and helps search engines to index the document. The

values are usually predefined (en for English, en-US for American English, ro for Romanian).
    - dir – specifies in which direction the text is written: ltr for left to right and rtl for right to left.
- scripting events – the attribute value is a script (function call) which is executed whenever one of the following events occurs on the HTML element:
    - onclick – the mouse button is clicked on the element;
    - ondblclick – the mouse button is double-clicked on the element;
    - onmousedown – the mouse button is pressed on the element;
    - onmouseup – the mouse button is released over the element;
    - onmouseover – the mouse pointer is moved over the element;
    - onmousemove – the mouse pointer is moving over the element;
    - onmouseout – the mouse pointer is moved outside the element;
    - onkeypress – a key is pressed and released when the element is selected;
    - onkeydown – a key is pressed down when the element is selected;
    - onkeyup – a key is released when the element is selected.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Academia de Studii Economice din Bucuresti</title>
<link rel="shortcut icon" href="http://www.ase.ro/site/index/favicon.ico">
<meta name="keywords" content="invatamant economic superior, invatamant la distanta, invatamant
</head>
<body>
<table width="880" border="0" align="center" cellpadding="0" cellspacing="0">
</table>
<br>
</body>
</html>
```

Figure 5.1 - Source code for www.ase.ro homepage

Although the content of HTML pages is very different from one page to another, there is a basic structure that all HTML pages should contain. You can deduct this structure by accessing a webpage using a browser like *Microsoft Internet Explorer* and viewing its HTML code by running *Source* command from the *View* menu. For instance, if you open www.ase.ro with *Microsoft Internet Explorer* and go to *View-Source*, you can view the source of the index page, Figure 5.1.

5.1.1 The Structure of a HTML Page

The first HTML element is !DOCTYPE which states what version of HTML is used in this document and what rules it complies to. This element also communicates to the Web server what kind of document it is delivering. The !DOCTYPE element has only a start tag and is found always at the very beginning of the HTML document. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> is the HTML 4.01 Transitional Document Type Definition (DTD) and declares that the code on the page includes presentation attributes and elements that W3C expects to phase out as support for style sheets matures.

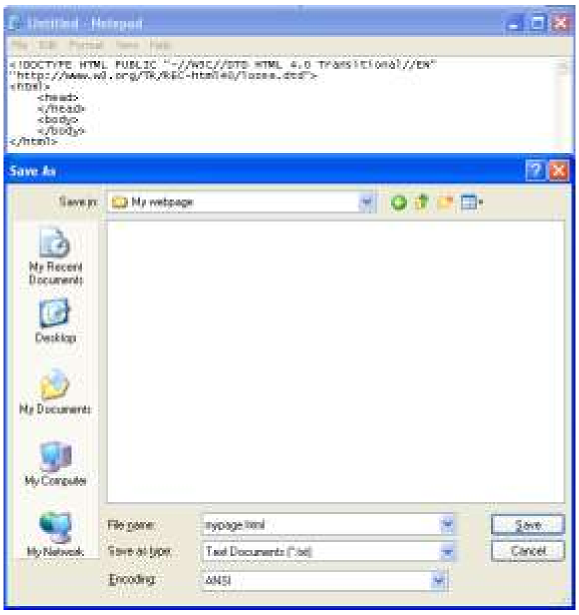
The next element is HTML, defined by start tag <html> and end tag </html> which closes all the tags on a page. It actually defines the boundaries of the HTML page, specifying that its content is text with HTML markup tags. The <head> start tag follows the <html> tag and defines the beginning of the document head which contains information about the document. The HEAD element ends with </head> which is immediately followed by <body> start tag, defining the beginning of the document body. The BODY element contains all the visible information on the webpage, along with structuring and formatting elements. Its ending tag </body> is just before the ending tag of the HTML document </html>.

These are the main HTML elements that define the structure of an HTML page and which can be found in almost any HTML webpage on the Web. Thus, in order to start building a page using HTML, first we have to define its basic structure as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
   <head>
   </head>
   <body>
   </body>
</html>
```

The text above can be written in any text editor like Microsoft Notepad. In order to be viewed using an internet browser like Microsoft Internet Explorer, Mozilla Firefox or Opera, the file must be saved with .html or .htm extension,

Figure 5.2. There is no difference between the two extensions except that older operating systems do not accept extensions made up of four letters.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

Save in: My webpage

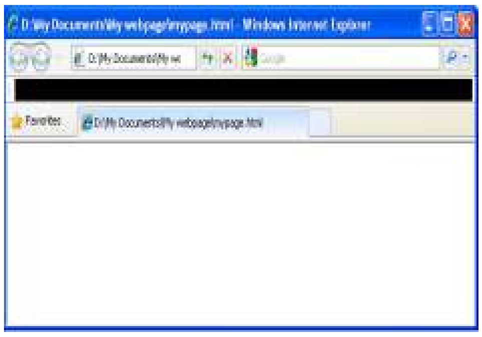
File name: mypage.html

Save as type: Text Documents (*.txt)

Encoding: ANSI

Figure 5.2 - Saving HTML code written in Notepad as webpage with .html extension

Opening the file with the default web browser, by simply double clicking on it in Windows Explorer, will result in displaying an empty internet browser window which shows the full path to this file in the title bar, Figure 5.3. Since we have written only HTML code to define the structure of our webpage, there is nothing displayed by the browser as the page is opened.



D:\My Documents\My webpage\mypage.html - Windows Internet Explorer
Favorites
D:\My Documents\My webpage\mypage.html

Figure 5.3 - Empty browser window

When building a webpage by writing HTML code in a text editor like Notepad, it is recommended to keep opened both the Notepad and the internet browser displaying the page. After changing the HTML code in Notepad, press Ctrl+S to save the document, press Alt+Tab to switch between Notepad and Internet Browser and then press F5 to refresh the content of the browser so that the changes done in Notepad are updated. Then, after viewing the refreshed content of the webpage, in order to make further changes, switch back to Notepad, by pressing Alt+Tab.

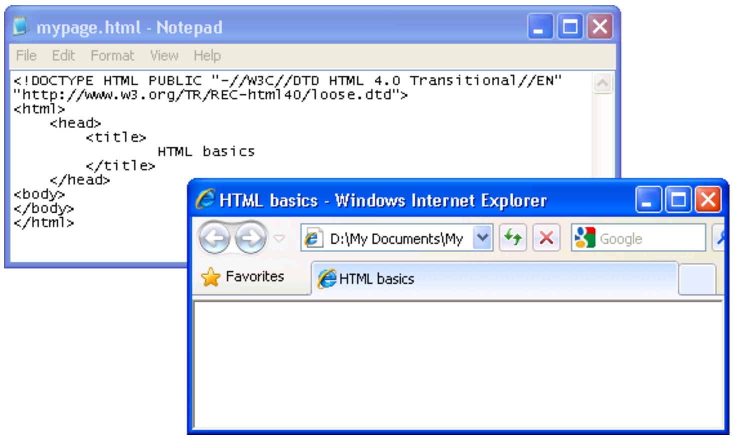
5.1.2 The HTML Page Head Tag

The head element, enclosed between <head> and </head>, contains only other HTML elements which describe documents characteristics. It cannot contain plain text since the content of this element is not displayed on the webpage. The elements contained in the head element cannot contain, at their turn, other HTML elements, and there is no specific order to write them.

The title element is the only HTML element whose content is shown in the browser displaying the webpage. The text written between <title> and </title> tags is displayed in the title bar of the browser. This title should be both meaningful for the webpage so that everyone understands what the page is about, and not too long so it fits into the title bar. Adding

<title> HTML basics </title>

inside the head element, between <head> and </head> tags, and previewing the saved file in Internet Explorer will result in displaying a page whose title bar has changed, Figure 5.4.


mypage.html - Notepad
File Edit Format View Help
HTML basics - Windows Internet Explorer
D:\My Documents\My
Google
Favorites
HTML basics

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
    <head>
        <title>
                HTML basics
        </title>
    </head>
<body>
</body>
</html>
```

Figure 5.4 - Webpage with title was set through the TITLE element

The other HTML elements that can be inserted in the head element are not visible from the webpage but are very useful for providing information about the content of the webpage is about to search engines crawlers, web applications or people interested in details about the webpage.

The meta element contains general information (meta-information) about the document and includes pairs of attributes (name, content). The name attribute specifies what kind of information is written in the content attribute.

For instance, the author and the description of the document can be specified in the meta element as:

```
<head>
    <title>  This  is  the  title  of  my  webpage  </title>
    <meta name = "Author" content = "Dragos Vespan" name
    = "Description" content =  "Test page">
</head>
```

The head element can also contain the following HTML elements:

- base – defines a default URL or a default target for all the links in the page. For instance, if you want to force all hyperlinks on the page to open in a new window, you can insert the following statement inside the head element:

```
<head>
    [HTML]
    <base target = "_blank" />
</head>
```

- link – sets up a relationship between the webpage and an external file. Usually, this HTML element is used with external style sheets and will be discussed later in this chapter.

The body element encapsulates all the content that has to be displayed on the webpage. It might contain a wide variety of nested HTML elements used for structuring and formatting the text, images and other objects.

5.2 Text emphasizing elements

For anyone using document editors like Microsoft Office Word or Star Office Writer, it is very easy to understand and use HTML elements for formatting the text displayed in a webpage. The HTML elements used for formatting the text practically do most of the things that usual document editors can do. The only difference is that the text you want to format in a certain way in HTML must be contained inside the HTML element that does the formatting, between its start and end tags.

5.2.1 Headings

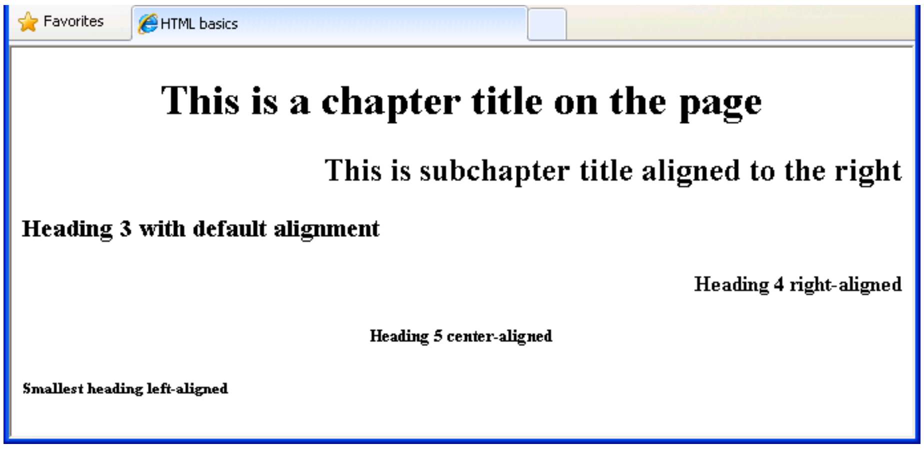
For instance, suppose you want to write a chapter title for your page. In Word, you would use headings to write chapter and sub-chapter titles: Heading 1 for chapter title, Heading 2 for subchapter title, Heading 3 for sub-subchapter title and so on. The same thing happens in HTML: you can use h1 element to write chapter titles, h2 for subchapter titles and so on. There are 6 headings available in HTML: h1 displaying default 24 point text size, h2 – 18 pints, h3 – 14 points, h4 –

12 points (which is the default text size in HTML), h5 – 10 points and h6 – 8 points. Beside the different size of the text, the headings are always bold and separated from the text above and below through an empty line. Thus, there is no difference between text written using h5 and normal bold text, only that the former has empty lines above and beyond.

The HTML heading elements may contain in their start tag the align attribute, which can take three possible values: left (which is the default alignment to the left margin of the page), center and right. Usually, the titles on a page are center-aligned.

```
<body>
    <h1 align = "center"> This is a chapter title on the page </h1>
    <h2 align = "right"> This is subchapter title aligned to the right </h2>
    <h3> Heading 3 with default alignment </h3>
    <h4 align = "right"> Heading 4 right-aligned </h4>
    <h5 align = "center"> Heading 5 center-aligned </h5>
    <h6 align = "left"> Smallest heading left-aligned </h6>
</body>
```

The presence of the align attribute in the start tag of an HTML element forces the content to be aligned according to the attribute's value. If the heading is part of another HTML element which is center aligned and there is no align attribute specified in heading's start tag then the heading will be, by default, center aligned. If there is an align attribute in the opening tag then the heading will be aligned according to the value it has. The HTML code above displays centered heading 1 and heading 5, right-aligned heading 2 and heading 4, left aligned heading 6 and default aligned (left if there is no other alignment specified) heading 3, Figure 5.5.



This is a chapter title on the page

This is subchapter title aligned to the right

Heading 3 with default alignment

Heading 4 right-aligned

Heading 5 center-aligned

Smallest heading left-aligned

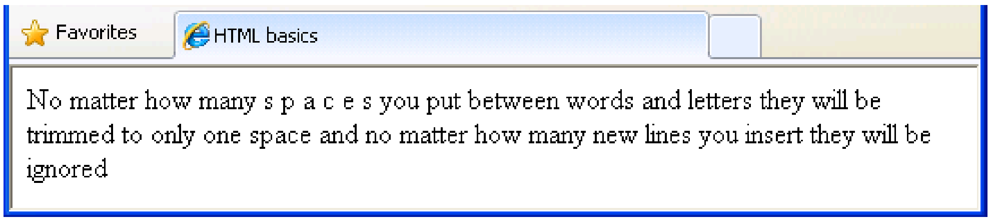
Figure 5.5 - Headings in HTML

5.2.2 Spaces

When typing text in a document editor, you don't have to worry about new line and spaces: if you want to get to the new line simply press Enter and if you want to add more space between two words, simply type space several times. In HTML things are different: no matter how many spaces you put between two words or letters, there will be only one space displayed and all the consecutive new lines you insert with Enter will count as one space.

```
<body>
   No   matter    how many  s    p    a    c    e    s    you put
   between words and letters    they will be trimmed to only one
   space

   and
   no matter
   how many new lines you insert
   they will be ignored
</body>
```

The result of the code written above is presented in Figure 5.6. As you can see, all multiple spaces were treated as only one space and all consecutive new lines were treated as one space. This way, in most cases it doesn't matter if you write the HTML code for the entire page on a single row or on several rows. Still, writing code on a single row is not recommended because it is very difficult to read and understand the HTML code in order to update or change it.



No matter how many s p a c e s you put between words and letters they will be trimmed to only one space and no matter how many new lines you insert they will be ignored

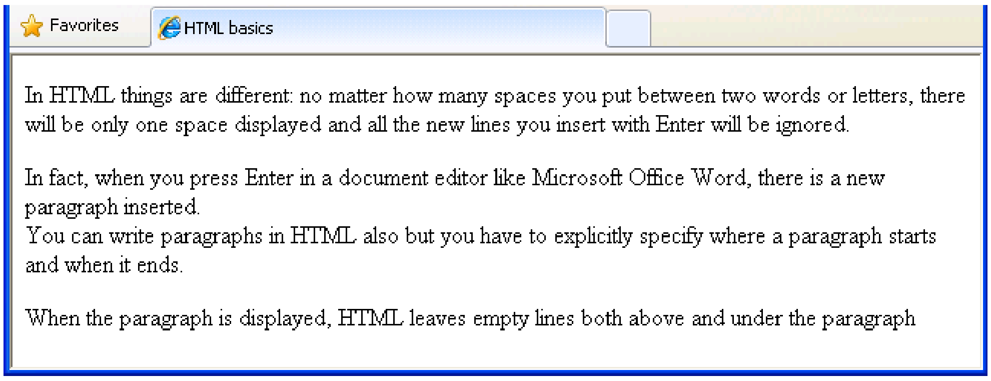
Figure 5.6 - HTML ignores multiple spaces and new lines

5.2.3 Paragraphs

In fact, when you press Enter in a document editor like Microsoft Office Word, there is a new paragraph inserted. You can write paragraphs in HTML also but you have to explicitly specify where a paragraph starts and when it ends, with start tag `<p>` and end tag `</p>`. When the paragraph is displayed, HTML leaves empty lines both above and under the paragraph. Also, if you want the text from a sentence inside the paragraph to begin at the left margin, similarly to inserting a new line in Word by pressing Shift+Enter, you have to use the <br> tag. Break HTML element has only a start tag <br />, without end tag. It is not necessary to write tags on different lines like the <p> and </p> tags in the HTML code below, but it is a good practice to structure the information in the HTML code according to the way that it should be displayed, so that the code is updated easier.

```
<body>
   <p>
        In HTML things are different: no matter how many spaces you put
        between two words or letters, there will be only one space
        displayed and all the new lines you insert with Enter will be ignored.
   </p>
   <p>
        In fact, when you press Enter in a document editor like Microsoft
        Office Word, there is a new paragraph inserted.
        <br /> You can write paragraphs in HTML also but you have to
        explicitly specify where a paragraph starts and when it ends.
   </p>
   <p>
        When the paragraph is displayed, HTML leaves empty lines both
        above and under the paragraph
   </p>
</body>
```

Putting text in paragraphs, between <p> and </p> tags, will insert empty lines above and below the text, Figure 5.7. Note that the second sentence of the second paragraph starts from the left edge, due to the <br /> tag inserted in the text in HTML code.



In HTML things are different: no matter how many spaces you put between two words or letters, there will be only one space displayed and all the new lines you insert with Enter will be ignored.

In fact, when you press Enter in a document editor like Microsoft Office Word, there is a new paragraph inserted.
You can write paragraphs in HTML also but you have to explicitly specify where a paragraph starts and when it ends.

When the paragraph is displayed, HTML leaves empty lines both above and under the paragraph

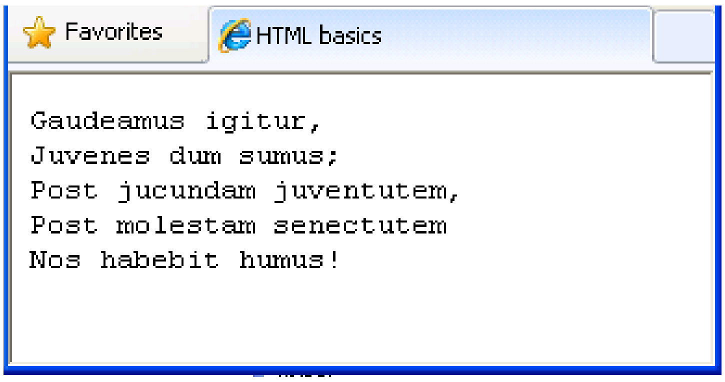
Figure 5.7 – Paragraphs and new lines in HTML

5.2.4 Preformatted Text

Still, if there is a need to insert text that would require many new lines, like rhymes of a poem or lyrics of a song, then the preformatted HTML element may be used, with <pre> start tag and </pre> end tag. This element displays the new lines from the text contained but trims multiple spaces to only one.

```
<body>
    <pre>
        Gaudeamus igitur,
        Juvenes dum sumus;
        Post jucundam juventutem,
        Post molestam senectutem
        Nos habebit humus!
    </pre>
</body>
```

The result for the code above is shown in Figure 5.8.



```
Gaudeamus igitur,
Juvenes dum sumus;
Post jucundam juventutem,
Post molestam senectutem
Nos habebit humus!
```

Figure 5.8 - Use of preformatted HTML element

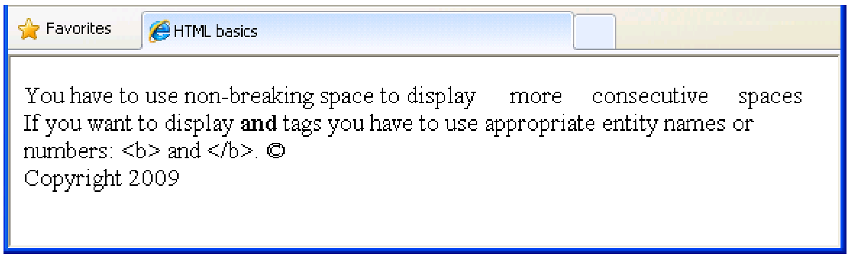
5.2.5 Character Entities

Some characters are reserved in HTML and their simple use might have unexpected results. For instance, as mentioned before, several spaces written in HTML are trimmed to only one space. In order to display more consecutive spaces, you can use either the non-breaking space entity name &nbsp; or its entity number &#160;.

```
<body>
    <p>
```

```
        You have to use non-breaking space to display &nbsp;&nbsp; &nbsp;
        more &nbsp;&nbsp;&nbsp; consecutive &nbsp;&nbsp;&nbsp;
        spaces. <br>
        If you want to display <b> and </b> tags you have to use
        appropriate entity names or numbers: &lt;b&gt; and &#60;/b&#62;.
        &copy; Copyright 2009
    </p>
</body>
```

The code above, exemplifies the use of character entity names and spaces and Figure 5.9 shows how the Internet Explorer browser interprets them.



You have to use non-breaking space to display   more   consecutive   spaces
If you want to display **and** tags you have to use appropriate entity names or numbers: <b> and </b>. ©
Copyright 2009

Figure 5.9 - Use of character entity names and numbers

Table 5.1 presents the main character entities used in HTML:

Table 5.1 – Character entity descriptions, names and numbers[*1]

Character to display	Description	Entity name	Entity number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;
€	euro	&euro;	&#8364;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
±	plus or minus	&plusmn;	&#177;

*1 Annex 3 contains a full list of this character entity description

5.2.6 Text Formatting

Word processors allow parts of text to be emphasized by applying different changes to the fonts used: type, size, color, and also bold, italic, underline, subscript, and superscript. HTML provides also elements which format the text they contain. The most used text emphasizing elements in HTML are presented in Table 5.2.

Table 5.2- HTML elements for text formatting

HTML tags	Action
<b>...</b>	Bold text
<i>...</i>	Italic text
<big>...</big>	Big text (increases the font size from its current value by 1)
<em>...</em>	Emphasized text (most of the browser display the text in italics)
<strong>...</strong>	Strong text (Internet Explorer displays the text in bold)
<small>...</small>	Small text (decreases the font size from its current value by 1)
<sub>...</sub>	Subscripted text (The text appears slightly below the baseline in a smaller font)
<sup>...</sup>	Superscripted text (The text appears slightly above the top of the preceding text in a smaller font)
<del>…</del>	Deleted text (The text is strikethrough)
<ins>…</ins>	Inserted text (In Internet Explorer the text is displayed underlined)

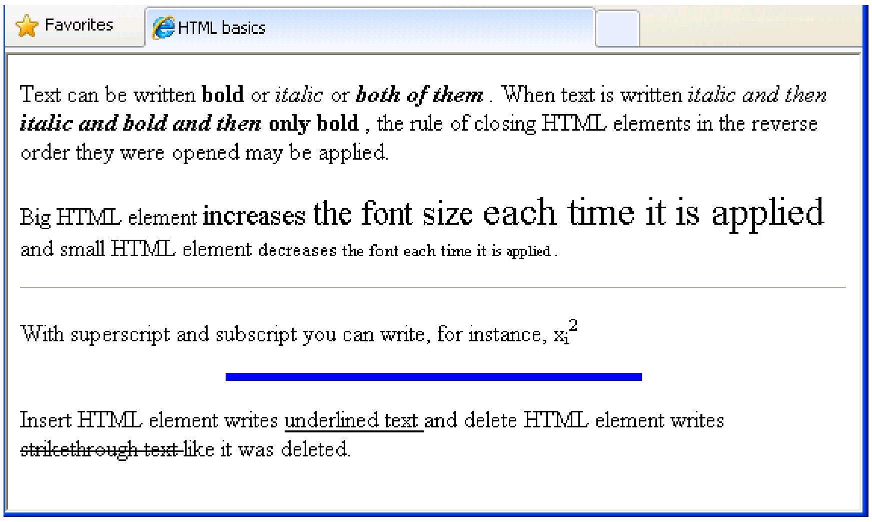
You can use more HTML elements for the same text by simply nesting one element into the other. There is no restriction on how many elements can be nested but the rule that has to be strictly respected is that any start tag should have a corresponding end tag for all non-empty HTML elements.

5.2.7 Horizontal Rule

A HTML page can be divided into sections through horizontal rules, by using <hr /> tag. Without any attributes, this tag draws a black horizontal line from the left margin to the right margin of the text and inserts empty lines above and below it. The width attribute defines the length of the line: a number value sets the length in pixels and a percent value sets the length related to the width of the page. The size attribute sets the thickness in pixels of the rule and the color attribute sets the color. Also rules can be aligned to center, right or left by assigning these values to the align attribute. In Figure 5.11, the first rule is the default one and the second rule is blue, covers 50% of the HTML page width, is center aligned and has a thickness of 3 pixels.

```
<body>
   <p>
```

```
        Text can be written <b> bold </b> or <i> italic </i> or <b> <i>
        both of them </i> </b>. When text is written <i> italic and then
        <b> italic and bold and then </i> only bold </b>, the rule of
        closing HTML elements in the reverse order they were opened may
        be applied.
    </p>
    <p>
        Big HTML element <big> increases <big> the font size <big> each
        time it is applied </big> </big> </big> and small HTML element
        <small> decreases <small> the font each time it is <small>
        applied </small> </small> </small>.
    </p>
    <hr />
    <p>
        With superscript and subscript you can write, for instance,
        x<sub>i</sub><sup>2</sup>
    </p>
    <hr size = ”3” width = ”50%” color = ”blue” align = ”center”>
    <p>
        Insert HTML element writes <ins> underlined text </ins> and
        delete HTML element writes <del> strikethrough text </del> like it
        was deleted.
    </p>
</body>
```



Text can be written **bold** or *italic* or ***both of them*** . When text is written *italic and then* ***italic and bold and then*** **only bold** , the rule of closing HTML elements in the reverse order they were opened may be applied.

Big HTML element increases the font size each time it is applied and small HTML element decreases the font each time it is applied .

With superscript and subscript you can write, for instance, x_i^2

Insert HTML element writes underlined text and delete HTML element writes ~~strikethrough text~~ like it was deleted.

Figure 5.10 - Text formatting in HTML

The code above exemplifies the application of HTML text emphasizing elements presented in **Error! Reference source not found.** and the way that Internet Explorer displays it is presented in Figure 5.10 . Note that if the same HTML element is opened more than once then it has to be closed the same number of times it was opened. The first closing tag of an HTML element which was opened several times will actually close the last opened instance of that HTML element. When more HTML elements are opened then a good rule to follow is that they are closed in the reverse order they were opened. Still, in HTML this rule does not have to be always applied, like in the above example where the text is written italic, then both italic and bold and then only bold.

5.2.8 Lists

Like document editors, HTML allows you to create lists in order to enumerate different items. There are three types of lists available: unordered lists (<ul>...</ul>), ordered lists (<ol>...</ol>) and definition lists (<dl>...</dl>). In unordered lists the items are marked by a bullet (small black circle) and in ordered lists items are marked by numbers. The marking symbol for both types of lists can be changed using the type attribute with the following values: disc (●), circle (○) or square (▫) for unordered lists and 1 (arabic numbers - 1,2,3...), A (upper alpha - A,B,C,…), a (lower alpha - a,b,c,…), I (upper roman - I, II, III, IV,…), i (lower roman - i,ii,iii,iv,…) for ordered lists. In both types of lists, items are inserted using the list Item HTML element whose end tag may be omitted (<li>).

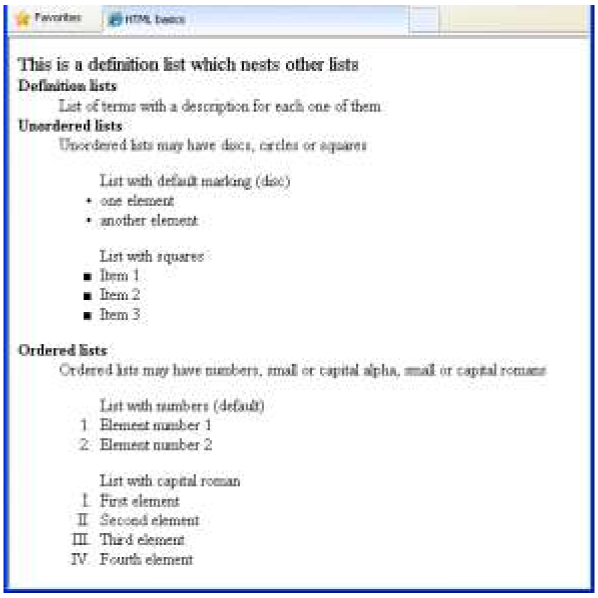
Definition lists are lists of terms with a description for each one of the terms. Each term is inserted using definition term HTML element (<dt>) and its correspondent definition is inserted using definition description HTML element (<dd>). For both elements, the end tag may be omitted.

The code below presents a definition lists with three elements: definition lists and what it represents, unordered lists and what kind of marking symbols they may have, ordered lists and what kind of numbering they may use. The way the browser interprets the code is exemplified in Figure 5.11.

```
<body>
   <p>
      <dl>
         <big>This is a definition list which nests other lists</big>
         <dt> <b> Definition lists </b>
         <dd> List of terms with a description for each one of them
         <dt> <b> Unordered lists </b>
         <dd>  Unordered lists may have discs, circles or squares
```

```
                  <ul> List with default marking (disc)
                       <li> one element
                       <li> another element
                  </ul>
                  <ul type = ”square”>
                       List with squares
                       <li> Item 1 <li> Item 2 <li> Item 3
                  </ul>
             <dt> <b> Ordered lists </b>
             <dd> Ordered lists may have numbers, small or capital alpha,
             small or capital romans
                  <ol> List with numbers (default)
                       <li> Element number 1 <li> Element number 2
                  </ol>
                  <ol type = ”I”> List with capital roman
                       <li> First element <li> Second element <li> Third
                       element <li> Fourth element
                  </ol>
          </dl>
       </p>
    </body>
```

The title of the definition list was formatted as big and each term in the definition list was formatted as bold, in order to be emphasized. <dd>, <dt> and <li> tags were not closed since HTML considers both that <dd> and <dt> tags are implicitly closed when another one of the two elements is opened or definition list is closed. <li> tag is implicitly closed when another list item element is opened or when the list it belongs to is closed. Note that if, by mistake, the <b> tag was not closed, then the text remaining until the end of the



This is a definition list which nests other lists
Definition lists
List of terms with a description for each one of them
Unordered lists
Unordered lists may have discs, circles or squares

List with default marking (disc)
- one element
- another element

List with squares
- Item 1
- Item 2
- Item 3

Ordered lists
Ordered lists may have numbers, small or capital alpha, small or capital romans

List with numbers (default)
1. Element number 1
2. Element number 2

List with capital roman
I. First element
II. Second element
III. Third element
IV. Fourth element

Figure 5.11 - Types of lists

webpage would have been displayed as bold. This is why it is very important to close each start tag that has a corresponding end tag, as soon as not closing it might generate errors throughout the entire page.

A good practice to use when writing HTML code is to close a tag just as it was opened and then to go and fill its content.

5.3 Hyperlinks and pictures

5.3.1 Hyperlinks

The main reason to build a web page is for publishing information over the Internet. Usually web pages are organized in web sites, representing collections of web pages related through hyperlinks.

Hyperlinks are references that allow users to quickly access over the internet resources like web pages, images, sound and video files. The anchor HTML element is used for creating hyperlinks inside a webpage. The basic syntax for this element is

```
<a href = "url_of_web_resource"> displayed_hyperlink </a>
```

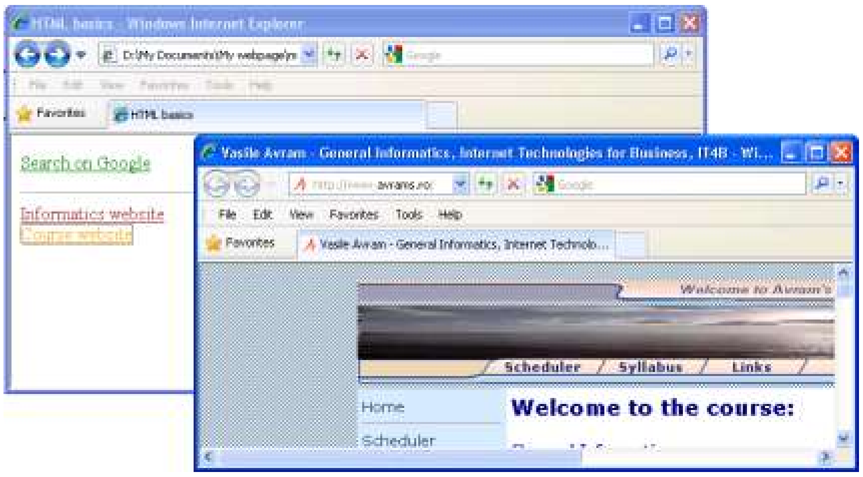
where url_of_web_resource is the value of the href property and represents the relative or the absolute address where the referred resource is found. displayed_hyperlink is the text or image that the visitor has to click in order to access the resource. If there is no other attribute specified inside the opening tag of the anchor then, by default, the referred resources is opened in the same browser window when the visitor clicks on the hyperlink. The target attribute sets the place where the resource will be opened. The _blank value forces the browser to open the resource into a new window.

By default, in Internet Explorer hyperlinks have the following colors: blue for unvisited hyperlink, purple for visited hyperlink and red for active hyperlink (an active hyperlink is a hyperlink that the visitor just clicked). The colors cannot be modified from inside the opening tag of the anchors. They may be set for the entire HTML page by changing the values of the following attributes in the <body> opening tag: link for default color, alink for active link and vlink for visited link.

```
<body link = "green" vlink = "brown" alink = "orange">
   <p>
      <a href = "http://www.google.ro/" target = "_blank"> Search on
      Google </a> <hr>
      <a href = "http://info.ase.ro/"> Informatics website </a><br>
      <a href = "http://www.avrams.ro/" target = "_blank"> Course
      website </a>
```

```
    </p>
</body>
```

In the code above, the attributes inserted in the body opening tag set up the colors for the hyperlinks all over the page: green for normal hyperlinks, brown for visited hyperlinks and orange for active hyperlinks.



HTML basics - Windows Internet Explorer
Search on Google
Informatics website
Course website
Vasile Avram - General Informatics, Internet Technologies for Business, IT4B - Wi...
Scheduler
Syllabus
Links
Home
Welcome to the course:

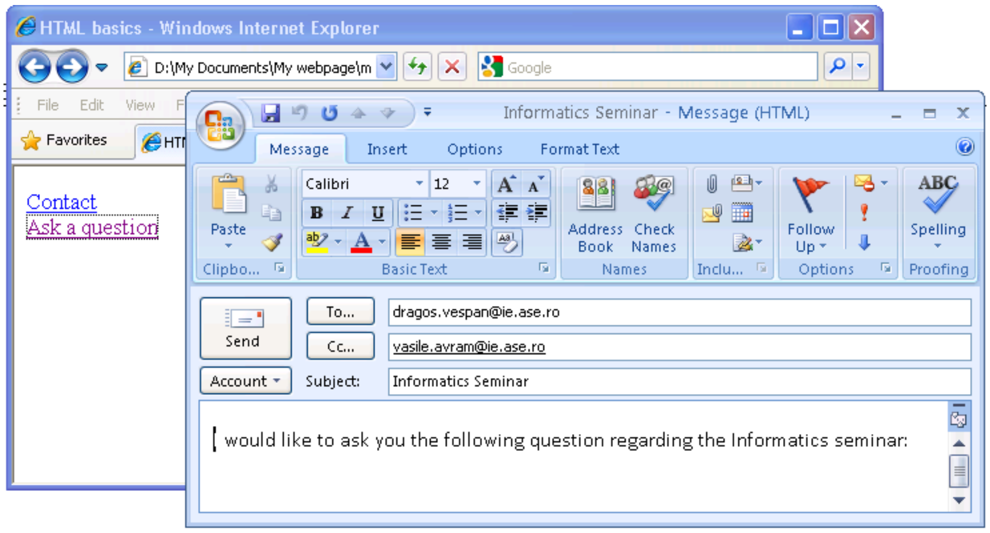
Figure 5.12 - Default, accessed and active hyperlinks

The _blank value of the target attribute in the first and third link will force the Google page to open in a new browser window, while the second link will open in the same page, Figure 5.1. Note that if the hyperlink refers a page which is not on the same website as the current webpage then the “http://” string must be used before the address of the website. If this is omitted, the browser interprets the URL as being on the same computer as the webpage and will return an error when the hyperlink is accessed. If there is no trailing slash to subfolder references, like “http://www.google.ro”, there will be two HTTP requests to the server because the latter will add the trailing slash and will generate a new request. This should be avoided, by adding trailing slash to all subfolder references, like “http://www.google.ro/”.

Hyperlinks can also be used to create e-mails with a specific subject and content, using the default mail editor installed on the visitor’s computer.

```
<body>
    <p>
        <a href = ”mailto:dragos.vespan@ie.ase.ro”> Contact </a> <br>
        <a href =
        ”mailto:dragos.vespan@ie.ase.ro?cc=vasile.avram@ie.ase.ro
        & subject=Informatics Seminar&body=I would like to ask you
        the following question regarding the Informatics seminar”>
        Ask a question </a>
    </p>
</body>
```

The first anchor from the code above will create a new e-mail having the recipient address already written in the To field. The second anchor creates an e-mail hyperlink that includes more recipient addresses, the subject and the content, Figure 5.13.



To... dragos.vespan@ie.ase.ro
Cc... vasile.avram@ie.ase.ro
Subject: Informatics Seminar

I would like to ask you the following question regarding the Informatics seminar:

Figure 5.13 - E-mail hyperlinks

The first parameter of mailto should always be preceded by ? and second and subsequent parameters should be preceded by &. Still, this method is not recommended for use on public web pages because there might be a lot of unwanted e-mails sent to the specified e-mail address.

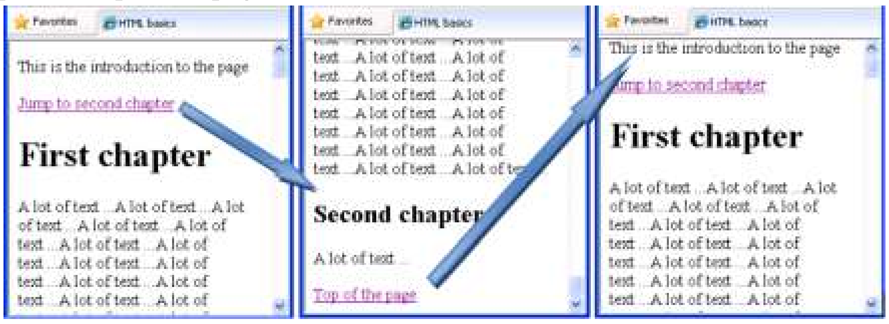
If the HTML page contains a lot of information and the user has to scroll a lot from top to bottom, anchors may be used to create links that jump to certain regions of the page. First of all, bookmarks have to be created by setting the name attribute to HTML elements that are going to be bookmarked. Suppose, for instance, that you want to bookmark the top of the page and the second chapter on the page so that when the visitor loads the page he can directly jump to the second chapter and when he finishes reading the page he can jump directly to the top of the page. First of all, the bookmarks have to be created: one just after the <body> tag and one just before the <h1> tag corresponding to the second paragraph. These bookmarks are created by inserting empty anchors elements with name attribute set.

```
<body>
    <a name = ”start”> </a>
```

```
        <p> This is the introduction to the page </p>
        <a href = "#second"> Jump to second chapter </a>
        <h1> First chapter </h1>
        <p> ... A lot of text ... </p>
        <a name = "second"> </a>
        <h2> Second chapter </h2>
        <p> ... A lot of text ... </p>
        <a href = "#start"> Top of the page </a>
    </body>
```

In the code above, <a name = "start"> </a> creates bookmark on the web page that can be referred as #start. Once the bookmarks defined in the page, we can create the links that will jump to these by setting the href property to the name of the bookmark preceded by #. Thus when the visitor clicks on the text displayed by <a href = "#start"> Top of the page </a>, the browser will jump directly to the anchor named start, Figure 5.14.

Usually, named anchors are used for creating table of contents at the beginning of large web pages. For each chapter inside the page, a bookmark is created and hyperlinks to all these bookmarks are added at the top of the page. If the browser cannot find a named anchor that was referred by a link, then it will jump at the top of the page when that link is accessed.



This is the introduction to the page
Jump to second chapter
First chapter
A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of text ... A lot of

Second chapter
A lot of text ...
Top of the page

Figure 5.14 - Using anchors inside the web page

5.3.2 Images

In order to make web pages more attractive and easier to read, the use of pictures is recommended. Pictures can be inserted in a web page using the image HTML tag whose basic syntax is:

```
        <img src="name_of_file" \>
```

where name_of_file represents the entire path and name of the file containing the picture. If the picture file is on the same folder as the webpage, then the path is not necessary. Usually, on websites, it is recommended that all pictures are stored in a separate folder, like “img”, so they are managed easier. If a picture called “sunset.gif” is stored in the “img” folder which is in the same folder as the webpage, then the src value would be “img/sunset.jpg”. If a picture from the web is used, then the value of src will be the entire URL of that picture, like “http://www.fabiz.ase.ro/siglase2.jpg”. Be aware that, in HTML, everything written between double quotes is case sensitive so there is a difference between “siglaase2.jpg” and “SIGLAASE2.JPG”. Many web servers interpret differently the two cases and may not send the resource if the URI string does not match binary with the real URI of the resource..

When using the basic syntax of the image HTML element, the browser will place the picture in its original size and in line with text, the same way as Word places an inserted picture. This way, only one line of text can be displayed on both sides of the picture. Usually, there should be more than on line of text written beside pictures and this can be solved by using the align attribute with left or right values. Also, if the original size of the picture is too big, then it will probably fill the entire screen, so it has to be resized. Resizing of the image HTML element is done by setting the number of pixels to width and height attributes. Attention should be paid to original dimensions of the picture in order to keep its proportions when it is resized.

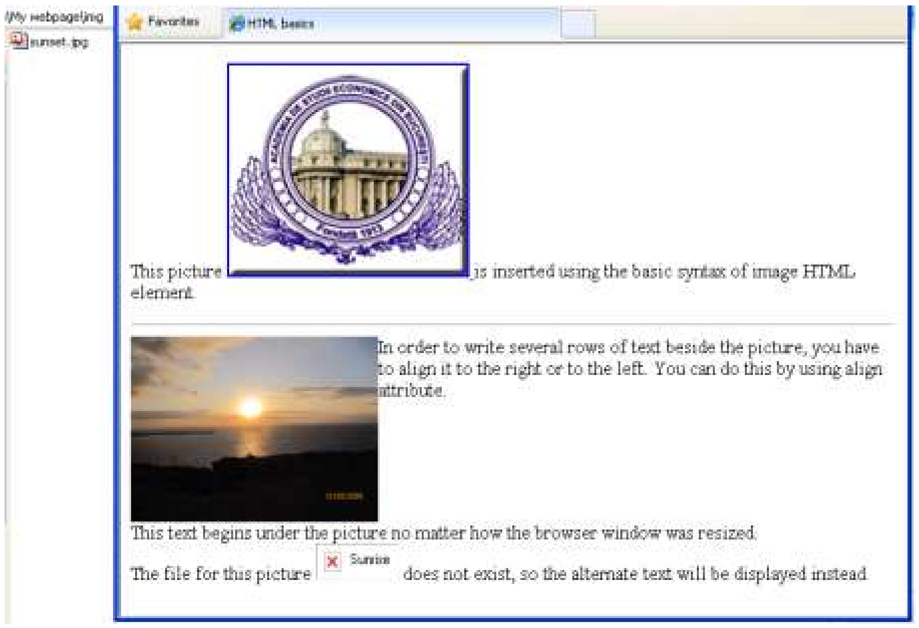
A picture will behave like a hyperlink if its tag is nested inside an anchor tag, like in the code below. Such a picture is surrounded by a square having the same colors as hyperlinks, in all their states: normal, visited and active. This way, if the visitor clicks on the first picture displayed by the code below then a new browser window will open and will load the homepage from “http://www.ase.ro”.

```
<body >
   <p>
      This picture
      <a href = "http://www.ase.ro" target = "_blank">
      <img src="http://www.fabiz.ase.ro/siglase2.jpg"> </a>
      is inserted  using the basic syntax of image HTML element. <hr>
      <img src="img/sunset.jpg" align = "left" width = "200" height =
      "150" alt = "Most beautiful sunset"> In order to write several rows of
      text beside the picture, you have to align it to the right or to the left.
      You can do this by using align attribute.
      <br clear="all">
      This text begins under the picture no matter how the browser
      window was resized. <br>
      The file for this picture <img src= "img/sunrise.jpg" alt="Sunrise">
      does not exist, so the alternate text will be displayed instead.
```

```
    </p>
</body>
```

Also, the source of the first picture is an URL, which means that, in order to be displayed when the webpage is accessed, the browser will first have to download it from the internet. The use of many pictures with sources on the Internet like this one is not recommended because the webpage will take time to load on slow internet connections.

The second picture is aligned to the left and resized from its original size of 4000x3000 pixels to 200x150 pixels, Figure 5.15. Although the displayed picture size is small, when the page is loaded the browser will download the entire file whose dimension is about 2MB. Usage of such big files is not recommended unless you want to create a photo gallery and let the visitor see the pictures in full size. The source of the picture is a file located in the "img" folder, so the src attribute contains only the relative path to this, "img/sunset.gif".



This picture is inserted using the basic syntax of image HTML element

In order to write several rows of text beside the picture, you have to align it to the right or to the left. You can do this by using align attribute.

This text begins under the picture no matter how the browser window was resized.

The file for this picture does not exist, so the alternate text will be displayed instead

Figure 5.15 - Using pictures in HTML

The use of the align = "right" attribute forces the browser to display all the text that follows beside the picture. The text will be written this way until the bottom of the picture is reached, no matter if HTML elements like paragraphs or headings are used. This might result in improper display of text if the window is resized. In order to avoid this and to be sure that a paragraph starts always below the picture, the break HTML element must be used with clear property set to "all".

The value of the alt attribute is displayed by the browser if the file of the picture is unavailable and cannot be loaded.

5.4 HTML elements for defining layout of web pages

5.4.1 Tables

HTML table is one of the most powerful HTML elements. This element can be used with a double purpose: one for representing data (true tables) and one for building the layout used to present information on a webpage (false tables).

In HTML, tables are built starting from their top left corner, row by row and, for each row, cell by cell. The HTML elements used for creating a table are:

- <table> ... </table> - define the beginning and the end of a table. Child HTML elements are table rows or table bodies;
- <tr> ... </tr> - define the beginning and the end of a table row. Child HTML elements are table cells or table headers;
- <td> ... </td> - define the beginning and the end of a table cell (table data). Its content can be formed of simple text and/or any HTML element that can be nested, including other tables;
- <th> ... </th> - define the beginning and the end of a table heading and replace <td> ... </td>. They represent, in fact, table cells with bold centered content;
- <thead> ... </thead> - define the table header, the section which usually contains comlumn headers. Child elements are table rows;
- <tbody> ... </tbody> - define the beginning and the end of a table section (table body). Child HTML elements are table rows;
- <tfoot> ... </tfoot> - define the table footer;
- <caption> ... </caption> - set up the title of the table.

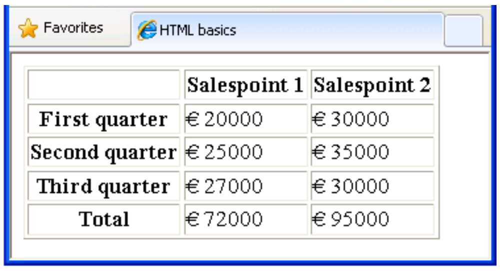
The simplest table that can be built in HTML contains only one row and one cell. The HTML code for building such a table is:

```
<table>
   <tr>
      <td> Cell content </td>
   </tr>
</table>
```

A regular table that presents data contains several rows and several columns. By default, a table built in HTML will not have any borders displayed in the browser but only its content. This is good for layout tables but when the table contains data its borders have to be displayed so that the visitor of the web page reads easier the values on rows and on columns. In order to accomplish this, the border attribute of the table opening tag has to be set to a value greater than 0.

```
<body >
    <table border = "1">
        <tr>
            <th> </th> <th> Salespoint 1 </th> <th> Salespoint 2 </th>
        </tr>
        <tr>
            <th> First quarter </th>
            <td> &euro; 20000 </td>
            <td> &euro; 30000 </td>
        </tr>
        <tr>
            <th> Second quarter </th>
            <td> &euro; 25000 </td>
            <td> &euro; 35000 </td>
        </tr>
        <tr>
            <th> Third quarter </th>
            <td> &euro; 27000 </td>
            <td> &euro; 30000 </td>
        </tr>
        <tr>
                <th> Total </th>
                <td> &euro; 72000 </td>
                <td> &euro; 95000 </td>
        </tr>
    </table>
body>
```

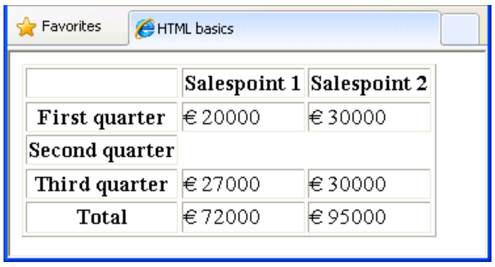
The table defined in the code above is presented in Figure 5.16. By default, the cells of the tables are auto-resizable so that they fit their content. The size of the columns will match the size of the cell containing the most text from that column. If the window does not allow each cell to be written on a single row then cells with more text will be vertically enlarged and so will be the rows they belong to.



	Salespoint 1	Salespoint 2
First quarter	€ 20000	€ 30000
Second quarter	€ 25000	€ 35000
Third quarter	€ 27000	€ 30000
Total	€ 72000	€ 95000

Figure 5.16 - Table with headers on both rows and columns

Because of the way the tables are constructed, by adding cells on rows, when building a table you have pay attention to how many cells you put in each row. Always the first cell on each row belongs to the first column; the second cell on each row belongs to the second column and so on. Cells missing on one row are considered to be the last cells of the rows and have no borders. For instance, if the HTML elements defining the second and third cells of the third row in table from Figure 5.16 are omitted, then the table will display some improper space inside, Figure 5.17.



	Salespoint 1	Salespoint 2
First quarter	€ 20000	€ 30000
Second quarter		
Third quarter	€ 27000	€ 30000
Total	€ 72000	€ 95000

Figure 5.17 - Cells omitted on one row of a table

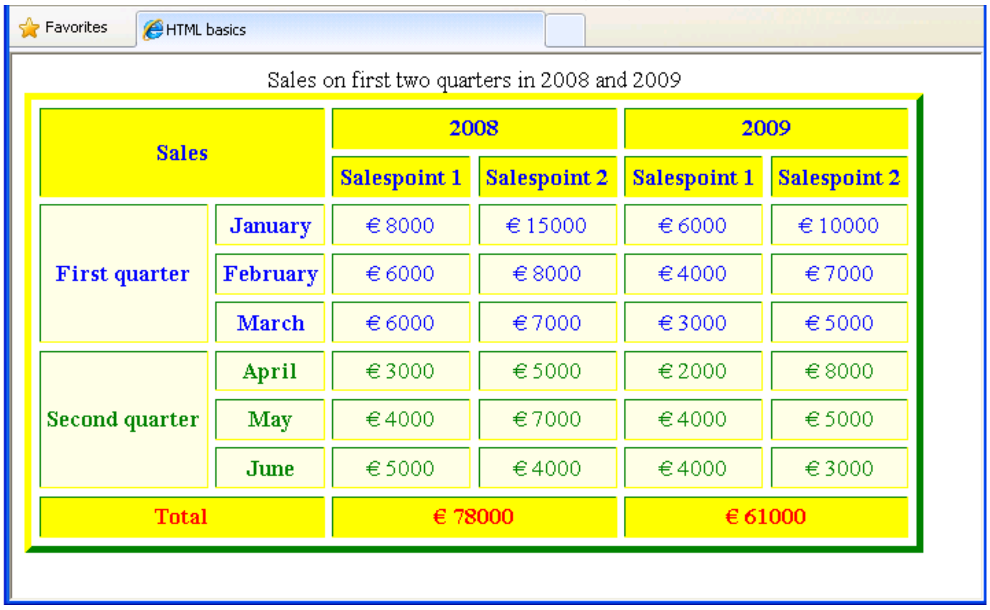
Still, there are cases when a row contains one or more cells spanning over two or more columns or a column contains one or more cells spanning over two or more rows. Spanning a cell over more columns is done using colspan attribute with the value indicating the number of columns that the cell spans over. Spanning a cell over more rows is done using rowspan attribute with the value indicating the number of rows that the cell spans over. Both colspan and rowspan attributes have to be written inside the opening tag of the spanned cell.

```
<body>
<table border = "5" bordercolorlight = "yellow" bordercolordark = "green"
cellpadding = "4" cellspacing = "5" style = "text-align:center">
 <caption> Sales on first two quarters in 2008 and 2009 </caption>
   <thead style = "color:blue; background:yellow; font-weight:bold">
         <tr> <td colspan = "2" rowspan = "2"> Sales </td> <td
            colspan = "2"> 2008 </td> <td colspan = "2"> 2009 </td>
         </tr>
         <tr> <td> Salespoint 1 </td> <td> Salespoint 2 </td> <td>
            Salespoint 1 </td> <td> Salespoint 2 </td>
         </tr>
   </thead>
   <tbody style = "color:blue; background:lightyellow">
         <tr> <th rowspan = "3"> First quarter </th> <th> January
            </th> <td> &euro; 8000 </td> <td> &euro; 15000 </td>
            <td> &euro; 6000 </td> <td> &euro; 10000 </td>
         </tr>
```

```
          <tr> <th> February </th> <td> &euro; 6000 </td> <td>
             &euro; 8000 </td> <td> &euro; 4000 </td> <td> &euro;
             7000 </td>
          </tr>
          <tr> <th> March </th> <td> &euro; 6000 </td> <td> &euro;
             7000 </td> <td> &euro; 3000 </td> <td> &euro; 5000
             </td>
          </tr>
    </tbody>
    <tbody style = "color:green; background:lightyellow">
          <tr> <th rowspan = "3"> Second quarter </th> <th> April
             </th> <td> &euro; 3000 </td> <td> &euro; 5000 </td>
             <td> &euro; 2000 </td> <td> &euro; 8000 </td>
          </tr
          <tr> <th> May </th> <td> &euro; 4000 </td> <td> &euro; 7000
             </td> <td> &euro; 4000 </td> <td> &euro; 5000 </td>
          </tr>
          <tr> <th> June </th> <td> &euro; 5000 </td> <td> &euro;
             4000 </td><td> &euro; 4000 </td> <td> &euro; 3000 </td>
          </tr>
    </tbody>
    <tfoot style = "color:red; background:yellow; font-weight:bold">
       <tr>
          <td colspan = "2"> Total </td>
          <td colspan = "2"> &euro; 78000 </td>
          <td colspan = "2"> &euro; 61000 </td>
       </tr>
    </tfoot>
</table>
</body>
```

The table generated from the code above is displayed in Figure 5.18. The code exemplifies the most commonly used table attributes and tags.

The border attribute sets up the thickness of the table's outside border. Its color can be set by using the bordercolor attribute or bordercolordark and bordercolorlight attributes. If bordercolor is used and the border size is greater than 1 then the browser will create a shadowing effect by setting a darker nuance of the color for the top and left margins and a brighter nuance for the right and bottom margins of the whole table and vice versa for the interior borders of the cells. This way it creates the effect of light coming from top left corner of the screen.. The two nuances can be set as different colors by using bordercolordark and bordercolorlight attributes in the <table> opening tag.



Sales on first two quarters in 2008 and 2009

Sales		2008		2009	
		Salespoint 1	Salespoint 2	Salespoint 1	Salespoint 2
First quarter	January	€ 8000	€ 15000	€ 6000	€ 10000
	February	€ 6000	€ 8000	€ 4000	€ 7000
	March	€ 6000	€ 7000	€ 3000	€ 5000
Second quarter	April	€ 3000	€ 5000	€ 2000	€ 8000
	May	€ 4000	€ 7000	€ 4000	€ 5000
	June	€ 5000	€ 4000	€ 4000	€ 3000
Total		€ 78000		€ 61000	

Figure 5.18 - A complex HTML table

The cellspacing attribute sets up the thickness of the table's inside borders and rules. It actually represents the distances between cells and between margin cells and outside borders.

By default, as exemplified in Figure 5.16, the cell with the most text on each column will fit exactly to its content, with no space between the text and the left and right margins of the cell. This can be annoying if it happens to most of the cells of the table so, an extra space between the content and the margins of the cell is required. This can be accomplished by using the cellpadding attribute whose value sets up the minimum space between the text contained in a cell and cell's margins, on columns and on rows.

All the attributes set up in the <table> start tag, including the style, will be applied throughout the entire table. If their values are changed inside the opening tag of a table element, then the values defined in the <table> tag will be overwritten for that element.

The caption HTML element defines the table title which will be displayed above the table. Table head, table body and table foot HTML elements define regions inside the table. They are useful because the attributes they contain inside the opening tag will be applied for the entire region, without the need of defining the attributes for each cell. In the example above, there are four regions defined in the table among which two table bodies having their content displayed with different colors so that it is easier for the reader to distinguish the two areas. For table head and table foot, the font was bolded and for first two columns of the table bodies, table heading element was used. Formatting the text of the table elements is done using styles, rule that may be applied to most of the HTML elements.

The layout of the table requires that the first cell spans over two rows and over two columns. This is done by using both colspan and rowspan attributes with values of 2. When spanning a cell over two rows, the cells of the second row are inserted from the left to the right, without affecting the space occupied on the row by the spanned cell. This way, if the spanned cell is the first one on the upper row then the first cell inserted in the second row will actually be on the second column. If a cell is spanned over two columns, then its width is resized according to the maximum widths of the cells belonging to each column.

```
<body>
    <table border = "1">
    <caption> Table with wrong spanning </caption>
      <tr>
          <td> Wrong spanning </td>
          <td rowspan = "2"> Row spanned cell </td>
      </tr>
      <tr>
            <td colspan = "2"> Column spanned cell </td>
      </tr>
    </table>
</body>
```

Using spanning requires a lot of attention. The code above exemplifies the misuse of spanning, by setting the second cell on the first row to span over two rows and the first cell on the second row to span over two columns.

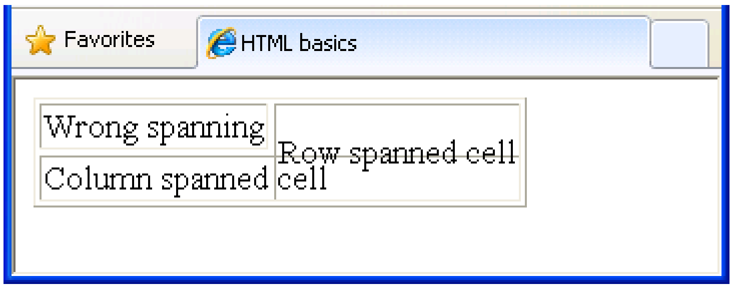


Figure 5.19 - Misuse of colspan and rowspan

Because of the way the tables are build, starting from the top left corner, row by row, by setting cells on each row, the second row of the second cell in the first row and the second column of the first cell in the second row will overlap, causing also their content to be overlapped, Figure 5.19.

One very common practice is to use HTML tables for formatting the layout of HTML pages. This helps the designer to divide the page in as many sections as he wants. The advantage of using tables derives from the fact that cells of the table are independent and adding a content to a cell will not affect the other cells.

Before actually building the webpage, the designer should first draw the layout of the page on a piece of paper in order to figure out what kind of table he has to create and what kind of spanning he has to apply. Suppose that we want to create a page with two navigation bars (an horizontal and a vertical one), one footer and with two columns content. The layout of the table would look like in Figure 5.20.

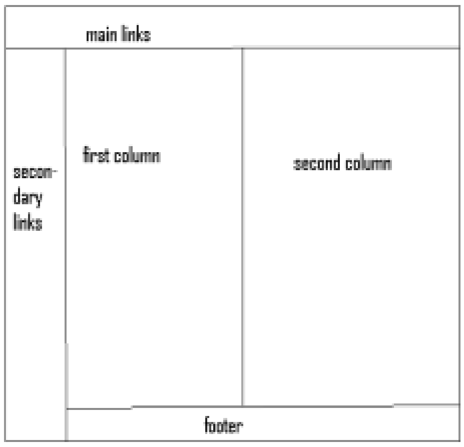


Figure 5.20 - Layout of HTML page

The table for this layout contains three rows and three columns: on the first row, one cell spanned over three columns; on the second row, one cell spanned over two rows and another two cells; on the third row, one cell spanned over two columns. The content of the cell in the first row should be center-aligned and the content of the cell spanned vertically (from the second row) should be vertically top-aligned. Also, the table must not be visible so the `border` attribute will be omitted.

When building a HTML table, its rows and columns are resized according to their content. It means that if there is no cell with much content on a column then that column would be very narrow and would lead to improper displaying. In order to fix this, the width of the columns in the table should be specified. Width can be specified in pixels or in percents. When specified in pixels, the width of the entire table will be the sum of largest cell widths per column and it will not vary depending on the size of the browser window or on the resolution. If the size of the window is smaller than the table, then scroll bars will be displayed to be used in order to view content outside the displayed area.

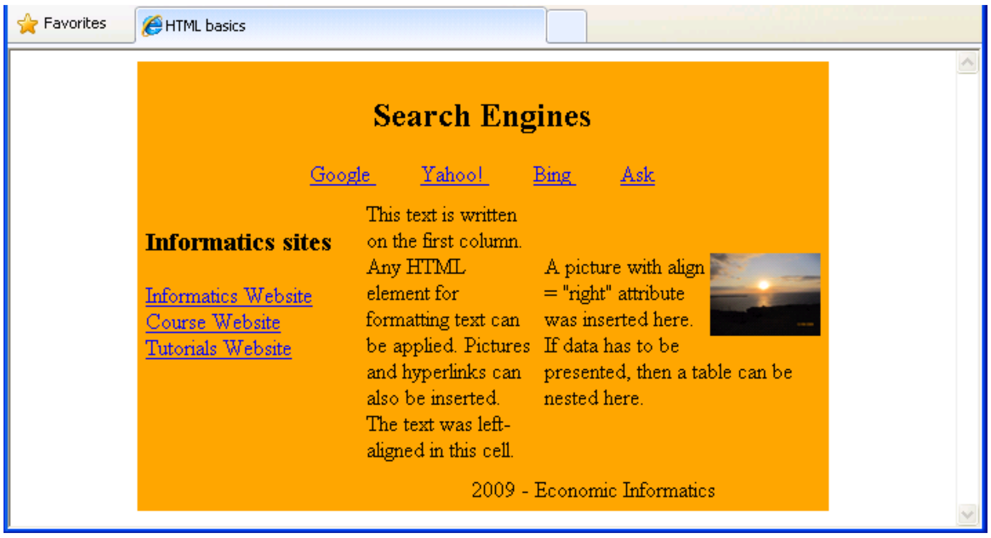
When width are specified in percents, then they are relative to the width of the entire window: a table with 100% percent width will extend over the entire window and a table with 50% width will always occupy half of the window.:

```
<body >
    <table align = "center" style = "width:75%; text-align:center" bgcolor
    = "orange" cellpadding = "4">
        <tr>
    <!-- This is the horizontal links bar -->
    <td colspan = "3" >
        <h2> Search Engines </h2>
        <a href = "http://www.google.ro"> Google </a> &nbsp;
          &nbsp; &nbsp; &nbsp;
        <a href = "http://www.yahoo.com"> Yahoo! </a>  &nbsp;
          &nbsp; &nbsp; &nbsp;
        <a href = "http://www.bing.com"> Bing </a> &nbsp;
          &nbsp; &nbsp; &nbsp;
        <a href = "http://www.ask.com"> Ask </a>
    </td>
</tr>
<tr>
    <!-- This is the vertical links bar -->
    <td valign = "top" rowspan = "2" style = "width:150px; text-
    align:left">
        <h3> Informatics sites </h3>
        <a href = "http://www.info.ase.ro"> Informatics Website
        </a> <br>
        <a href = "http://www.avrams.ro"> Course Website </a>
        <br>
        <a href = "http://www.w3schools.com"> Tutorials Website
        </a> <br>
    </td>
    <!-- This is the left column -->
    <td style = "text-align: left">
        This text is written on the first column. Any HTML element for
        formatting text can be applied. Pictures and hyperlinks can
        also be inserted. The text was left-aligned in this cell.
    </td>
    <!-- This is the right column -->
    <td style = "width:200px; text-align:left">
        <img src = "img/sunset.jpg" align = "right" width = "80"
        height = "60">
        A picture with align = "right" attribute was inserted here. If
        data has to be presented, then a table can be nested here.
    </td>
</tr>
<tr>
    <!-- This is the footer of the page -->
```

```
                <td colspan = "2" style = "height:10px">
                    2009 - Economic Informatics
                </td>
            </tr>
        </table>
    </body>
```

The code above is shown by the browser as in Figure 5.21: the table is set to occupy 75% of the window, to have center aligned text and orange background. The absence of the border attribute forces the browser to hide the margins of the table. Also, for a greater visibility, the distance between the text and the margins of the cells is increased through the cellpadding attribute.

The <!-- ... > tag is used for inserting comments into the HTML code. Everything written inside this HTML tag will not be displayed in the browser window and will help the author to remember what each section of the page represents.



Search Engines

Google Yahoo! Bing Ask

Informatics sites

Informatics Website
Course Website
Tutorials Website

This text is written on the first column. Any HTML element for formatting text can be applied. Pictures and hyperlinks can also be inserted. The text was left-aligned in this cell.

A picture with align = "right" attribute was inserted here. If data has to be presented, then a table can be nested here.

2009 - Economic Informatics

Figure 5.21 - Layout table

Basically, the table contains three rows and three columns. The first row contains only one cell which spans over all three columns which represents the horizontal links bar. If there is a width in pixels specified for this row, it will be ignored, being overwritten by the size of the table.

The first cell of the second row spans over the last two rows and contains the vertical links bar. There are two alignments set for this cell. the top vertical alignment will always display the contained links in the upper part so the visitor has quick access to them no matter how big is the web page. Also, for a better view, the horizontal alignment is set to left and overrides the default center alignment of the table, which is set up in the <table> opening tag. Due to the fact that the content of this cell is improbably to change over time, the size of this cell is set to 150 pixels so it just fits the text of the hyperlinks inside. This size will be kept no matter how much the browser window is enlarged.

The second and the third cell of the second row contain left-aligned text, overwriting the default center alignment of the table. The third cell has a specified width of 200 pixels mainly because of the picture contained. A lower width will force the improper display of single words beside the picture. Like the first cell, the width of this cell will be kept constant to this value no matter how much the window is extended.

Due to the fact that both of first and third cells have fixed size widths, the middle cell will be always resized depending on the size of the browser window or on the resolution of the screen so that the sum of the three cells sizes equals 75% of the window as the table is set to occupy. Note that if the browser window is shrunk too much, then all the columns will be dimensioned to smaller size so that their content is displayed as properly as possible.

The third row of the table contains only one cell spanning over two columns. Because of the first cell in the second row which spans over two rows, this cell will actually begin on the second column, so it will span over second and third columns. Its height is fixed to 10 pixels. There is no alignment specified in its opening tag, so it will keep the default center alignment of the table. Because of its spanning, the text contained in this cell will be aligned relatively to the second and the third column.

5.4.2 Horizontal Rule

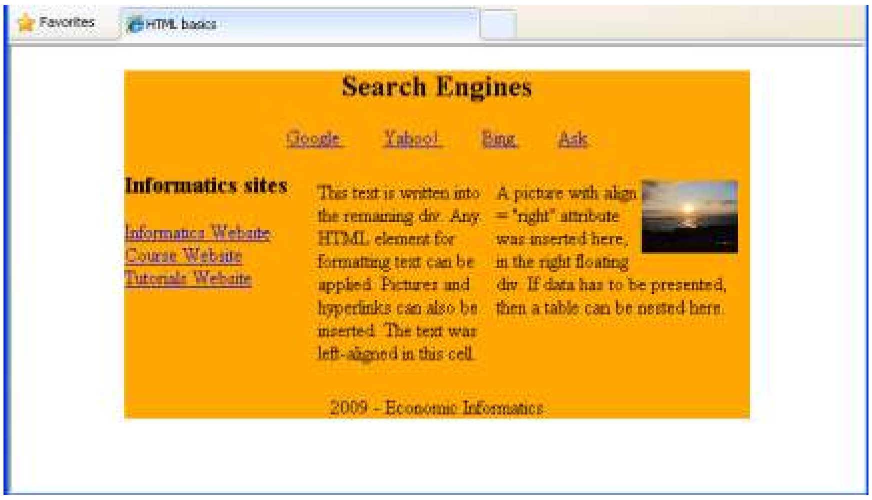
Almost the same layout can be obtained by using div HTML element with <div> and </div> opening and closing tags.

```
<body>
    <div style = " margin: auto; float: center; width: 75%; text-align: center;
    min-width: 400px; min-height: 230px; background-color: orange">
        <div>
            <!-- This is the horizontal links bar -->
            <h2> Search Engines </h2>
            <a href = "http://www.google.ro"> Google </a> &nbsp;
            &nbsp; &nbsp; &nbsp; <a href = "http://www.yahoo.com">
            Yahoo! </a>  &nbsp; &nbsp; &nbsp; &nbsp; <a href =
            "http://www.bing.com"> Bing </a> &nbsp; &nbsp; &nbsp;
            &nbsp; <a href = "http://www.ask.com"> Ask </a>
        </div>
        <div style = " float:left; width:150px; text-align:left; ">
            <!-- This is the vertical links bar -->
            <h3>Informatics sites</h3> <a href="http://www.info.ase.ro">
            Informatics    Website    </a>    <br>    <a    href    =
            "http://www.avrams.ro"> Course Website </a> <br> <a href =
            "http://www.w3schools.com"> Tutorials Website </a> <br>
        </div>
        <div style = "float: right; width: 200px; text-align:left; padding: 10px">
            <!-- This is the right column -->
            <p>
                <img src = "img/sunset.jpg" align = "right" width = "80"
                height = "60">
                A picture with align = "right" attribute was inserted here, in
                the right floating div. If data has to be presented, then a table
                can be nested here.
            </p>
        </div>
        <div  style  =  "margin-left:150px;  margin-right:  200px;text-
        align:left; padding:10px ">
            <!-- This is the middle column -->
            <p>
                This text is written into the remaining div. Any HTML element
                for formatting text can be applied. Pictures and hyperlinks can
                also be inserted. The text was left-aligned in this cell.
            </p>
        </div>
        <div style = "text-align: center;">
            <!-- This is the footer of the page -->
            2009 - Economic Informatics
        </div>
    </div>
</body>
```

The Internet Explorer browser displays the code above as in Figure 5.22. When using div HTML element, the approach is slightly different than when using tables for layout. First of all, there is only one HTML element used, div, and not more like in tables: table, tr, td, thead, tbody, tfoot and so on. Then, the most used columns layout based on div is made up of three columns: a div HTML element representing the left column, a div HTML element representing the right column and a div HTML element that fills the space between the two columns.

In our case, the page occupies 75% of the window and is center aligned. In order to accomplish this, a div HTML element was used just after the <body> opening tag to encapsulate the entire content of the body, which we will call container. Its closing </div> tag is just before the </body> closing tag. Properties for this div, as for the other div HTML elements too, are defined using style attribute and are separated through semicolon „;". The width can be set as an absolute value, in pixels, or as a relative value, in percents. In the example, the div content will always occupy 75% of the browser window. Float property defines the position of the div inside the HTML element that contains it, which is the body. When centered, it has to be used in conjunction with the margin property set to auto, required for adapting the right and left margin of the page so that the div is always center aligned. The content of the div is centered by using the text-aligned property.

The width property resizes the div content when the window is resized. If the window is resized to a small width, then 75% of this may not be enough to display the entire content. This is why the min-width property is used. In this case, it forces the browser to display horizontal rules when 75% of its width is less than 400 pixels. This ensures that the content of the div is properly displayed no matter how much the browser window is shrunk. The min-height property, correlated with the background-color property, ensures that the background is orange on at least 230 pixels height, no matter the height of the div content.



Search Engines

Google Yahoo! Bing Ask

Informatics sites

Informatics Website
Course Website
Tutorials Website

This text is written into the remaining div. Any HTML element for formatting text can be applied. Pictures and hyperlinks can also be inserted. The text was left-aligned in this cell.

A picture with align = "right" attribute was inserted here, in the right floating div. If data has to be presented, then a table can be nested here.

2009 - Economic Informatics

Figure 5.22 - Layout using div HTML element

Once the div container is defined, it can be divided into more areas by inserting nested div HTML elements. For the layout in Figure 5.22, there are five nested div HTML elements.

The first div has no parameters. This means that all its properties will be inherited from the container div: width, text alignment, background color. In fact, unless the properties are changed, all the nested div HTML elements will inherit the container properties. Still, the height of this div will be auto resized depending on its content.

The three columns are created by using three consecutive div HTML elements. The float property sets the second div to be the left column and the third div to be the right column. For both of these, the width properties set up their fixed width and the text-align properties overwrite the text alignment of the container and align their text to the left. The padding property used for the third div ensures a distance of 10 pixels between the content of the div and its margins.

The fourth div HTML element has no float property set, so its content will be displayed in the same line level as both left and right div HTML elements. In order to avoid its content being displayed over the content of the other two div HTML elements, its margins are set at a distance of 150 pixels from the left margin and 200 pixels from the right margin of the container through margin-left and margin-right properties.

Because there is no width specified for the fourth div, it will automatically resize when the browser window is resized. The minimum width of the container (400 pixels) and the distances of 150 pixels and 200 pixels from the left and right margins of the container will ensure that the width of this fourth div will not drop below 50 pixels no matter how much the browser window is shrunk. More, when the width of the page is reduced, so is the width of this div. This will result in the increase of its height so that the entire content is displayed. Because this is a nested div, the height of the container will also be increased.

The final div is a simple one, with center aligned content. It will always be displayed under the fourth div which is resizable with respect to the resolution of the screen or to the size of the browser window.

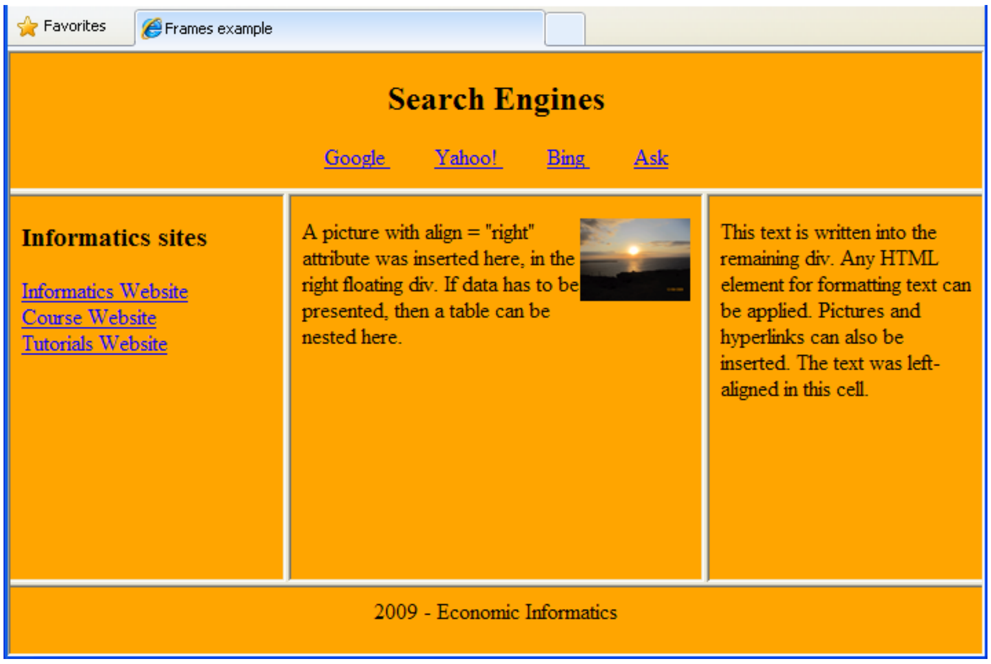
div HTML elements are more powerful and more flexible than layout tables. They are easier to use, to position and to replace. div elements support 3-D positioning and can be displayed one over another. Lately, it is recommended to use div HTML elements for layout instead of HTML tables.

5.4.3 Frames

Another method for defining the layout of a web page is the use of frames. Frames actually divide the web page in several sections, each of the section loading a different web page. The layout used as example in this chapter can be obtained by using the following code:

```
<html>
    <head>
        <title> Frames example </title>
    </head>
    <frameset rows = "100,*,50">
        <frame src = "div1.html">
        <frameset cols = "200   ,*,200">
            <frame src = "div2.html">
            <frame src = "div3.html">
            <frame src = "div4.html">
        </frameset>
        <frame src = "div5.html">
    </frameset>
</html>
```

The code above is displayed by the browser as in Figure 5.23. With frames, more web pages are displayed in the same browser window. Each section of the page is represented by a frame and independently displays a different web page. This is one of the reasons why frames are not used very widely: the web developer must manage more html pages. Also, pages containing frames are difficult to print.



Search Engines

Google Yahoo! Bing Ask

Informatics sites

Informatics Website
Course Website
Tutorials Website

A picture with align = "right" attribute was inserted here, in the right floating div. If data has to be presented, then a table can be nested here.

This text is written into the remaining div. Any HTML element for formatting text can be applied. Pictures and hyperlinks can also be inserted. The text was left-aligned in this cell.

2009 - Economic Informatics

Figure 5.23 - Frames layout

As you can see in the code above, the body HTML element was replaced by the frameset HTML element. This is because the two of them cannot exist on the same page. The frameset HTML element defines how to divide the parent HTML element into horizontal or vertical frames, depending on the attribute used in <frameset> opening tag. In the code, the parent of the first frameset HTML element is html. The attribute is rows and the value of this attribute has three components: 100, *, 50; which indicate the height of each row. This means that the entire window (content between <html> and </html>) will be divided into three rows: the first row will have 100 pixels height, the third row will have 50 pixels height and the second row will occupy the rest of the window. Each row will be defined by a frame tag or by a frameset element.

The frame tag defines which html page is loaded into the section it represents. The name of the page is defined in the value of the src attribute inside the tag. Each one of the pages displayed in the frames is independent of the others, have its own HTML structure and code and may be displayed separately. For instance, the entire code of the HTML page loaded in the first frameset, div1.html, is presented below:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
	<head>
		<title> HTML basics </title>
	</head>
	<body style = "text-align: center; background-color: orange; ">
		<!-- This is the horizontal links bar -->
		<h2> Search Engines </h2>
			<a href = "http://www.google.ro"> Google </a> &nbsp;
			&nbsp; &nbsp; &nbsp; <a href = "http://www.yahoo.com">
			Yahoo! </a>  &nbsp; &nbsp; &nbsp; &nbsp; <a href =
			"http://www.bing.com"> Bing </a> &nbsp; &nbsp; &nbsp;
			&nbsp; <a href = "http://www.ask.com"> Ask </a>
	</body>
</html>
```

Practically, the body HTML element has the same content as the first div from Figure 5.22. To get the same result as in the examples above, the body of div2.html will have the same content as the second div, the body of div3.html the same content as the third div and so on.

The first and the third elements of the frameset defined for the entire page display HTML pages div1.html and div5.html. The second element is represented by a nested frameset element, whose cols attribute value contains three elements: 200, *, 200. This means that this section will be divided into three columns where the first and the third have a 200 pixels width and the second occupies the rest of the section. Each column is defined by a frameset element which will load the webpage specified in its src attribute.

In order to keep consistency, all HTML pages displayed should have the same background color specified in the style attribute of the <body> opening tag: orange.

5.4.4 Colors

Colors in HTML are made up of the three basic components of light: RGB – red, green and blue. Computer, LCD TVs or mobile phone screens have different resolutions that are measured in pixels: 1024x760 resolution means that the screen can display 1024 pixels in width and 760 pixels in height. Each one of these pixels is composed of three subpixels: a red one, a green one and a blue one. When all the subpixels are shut off, the pixel will not lit which means that it will be black. When all subpixels are lit at maximum, the pixel will display the combination of the three colors, which is white. If only the red subpixel is lit and the other two are shut then the pixel will light red and so on.

Most of the screens can display 256 values of intensity for each subpixel: from 0, when the subpixel is completely shut off, to 255 when the subpixel is lit at maximum. It means that, taking in account the three kinds of subpixels they contain, screens can display a maximum of 256x256x256 = 16.777.216 variations of colors, from dark black to bright white. This is why you see that monitors, for instance, can display 16.7 million colors.

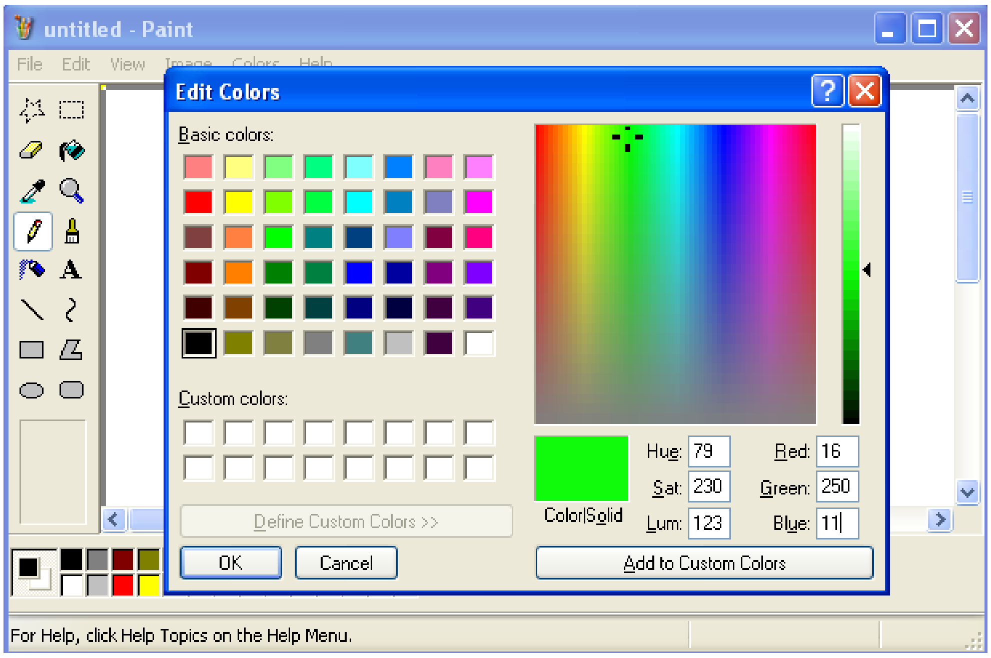
In HTML, colors are represented in three ways: by name, by hexadecimal value or by RGB value. W3C HTML and CSS standards have listed only 16 valid color names: aqua, black, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white and yellow. Most of the browsers though can recognize almost 150 color names: hotpink, darkred, lightcoral etc.

Still, defining a color by its name is very restrictive: less than 200 colors can be represented this way, out of 16.7 million. In order to represent any color, the combination of red, green and blue intensities may be used.

The hexadecimal representation is made up of 6 hexadecimal digits preceded by #: two digits for the intensity of each subcomponent. This way, colors vary from #000000 which is pure black to #FFFFFF which is pure white. #FF0000 represents pure red, #00FF00 is pure green and #0000FF is pure blue. Remember that FF_H decimal value is 255. You can use any combination you want: #10FA0B represents a color made from red with intensity 10_H (whose value is 16), green with intensity FA_H (which is 250) and blue with intensity $0B_H$ (whose value is 11).

Because hexadecimal values are difficult to compute quickly, the RGB representation of colors may be used. This way, pure black is represented as RGB(0,0,0), pure white as RGB(255,255,255), pure red as RGB(255,0,0), pure green as RGB(0,255,0) and pure blue as RGB(0,0,255). The color above, #10FA0B can be represented as RGB(16,250,11). Keeping the same value for all subcomponents will result in nuances of grey - for instance #3A3A3A or RGB(58,58,58).

The RGB representation is useful when you want to visually choose a color. For this, you can start *Paint*, access *Edit Colors* command from the *Colors* menu and press *Define Custom Colors* button. There you can visually pick any color you want and its red, green and blue values will be displayed in the textboxes below, Figure 5.24.



Hue: 79
Sat: 230
Lum: 123
Red: 16
Green: 250
Blue: 11

Figure 5.24 - Choosing colors in Paint

Color values are usually used for properties defined inside the style attribute of HTML elements opening tag.

5.5 Styles and CSS

The main purpose of HTML is to define the content of the document by using HTML elements like headings or paragraphs and not to format the way the web pages look. When the text on a webpage is formatted by defining HTML elements individually for each section of the page that has to look differently, it is very difficult to keep track of all these elements and to maintain their consistency especially when the web developer has to deal with large websites.

In order to solve this problem, W3C introduced CSS (Cascading Style Sheets) which are recommended to be used for changing different HTML elements properties, like color, face and size of the font used for displaying its content, alignment, width etc. CSS styles are combinations of `property:value` pairs separated through semicolons (;), encapsulated between { and }. Styles may be applied to one or more HTML elements.

There are three types of CSS styles that can be defined:

- Inline style, defined directly inside the opening tag of an HTML element. This type of style applies exclusively to the HTML element it was defined for and has the higher priority (it will be applied over the styles defined for the HTML element category and over the styles defined for parent HTML elements);
- Embedded style, defined in the head section or in a CSS file. This type of style will be applied as default to all HTML elements on the page that have the opening tag it was defined for;
- Named style, defined also in the head section or in a CSS file. This type of style will be applied to all HTML elements that belong to the class it represents, through the class="style_name" attribute written in their opening tag.

Inline styles can be defined for all HTML elements inside the body of the web page. The syntax for an inline style is:

<*tag* style="*property:value; property;value; … ; property:value*">

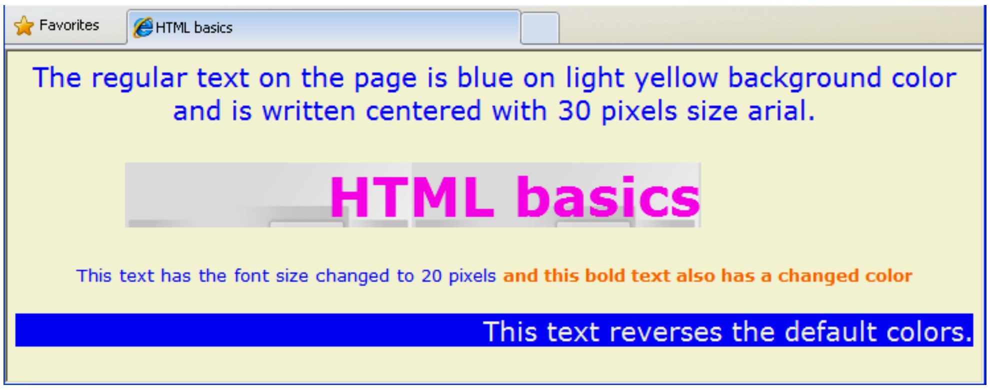
An inline CSS style will be applied exclusively to the HTML element in whose opening tag it was defined, affecting all its child elements. All the elements defined before the opening tag or after the closing tag of the HTML element containing the inline style remain unaffected by this.

```
<body style = "background-color: lightyellow; text-align: center; font-family:
verdana; font-size: 30px; color: blue">
    The regular text on the page is blue on light yellow background color and
    is written centered with 30 pixels size arial.
    <h1 style = "background-image: url('img/back.jpg'); text-align: right;
    width: 60%; margin-left: 120px; color: #FF00F0">
        HTML basics
    </h1>
    <p style = "font-size: 20px">
```

```
            This text has the font size changed to 20 pixels
            <b style = "color: RGB(255,128,10)">
                and this bold text also has a changed color
            </b>
    </p>
    <p style = "text-align: right; background-color: blue; color: lightyellow">
            This text reverses the default colors.
    </p>
</body>
```

The code above exemplifies the use of inline CSS styles and the way Internet Explorer interprets it is shown in Figure 5.25. First of all, an inline style is applied to the entire body, being defined in the <body> opening tag. All the settings defined by this style will be inherited by all the HTML elements used in the body: light yellow background color, center alignment of text contained, verdana font family, 30 pixels font size and blue font color.

The inline style defined in the heading overwrites the background setting by displaying an image over the light yellow background of the body. The image is used with its original size: if this is greater than the area used for the heading element then only the top left part of the image will be shown; if the original size of the image is smaller than the area it is used for then the image will be multiplied horizontally and vertically until the area is filled. The width property sets the percent from the pages' width that the heading occupies and the left-margin property sets the distance of the heading area from the left border. All inline style settings (background, font color) will be applied only to this area, including all nested HTML elements.



The regular text on the page is blue on light yellow background color and is written centered with 30 pixels size arial.

HTML basics

This text has the font size changed to 20 pixels **and this bold text also has a changed color**

This text reverses the default colors.

Figure 5.25 – Use of inline styles

The inline style defined in the first paragraph overwrites only the font size setting defined in the <body> opening tag, size which will be applied to b nested element also. The inline style defined in the <b> opening tag overwrites, by inheritance, the default font color. The inline style defined in the second paragraph overwrites the background color, font color and alignment of the text it contains.

Suppose there are twenty paragraphs on the web page and you want to apply the same style for all of them. Using inline CSS styles is not feasible since it takes a lot of time and a lot of code to write and complicates very much the procedure of changing the values of the properties used. In this case, embedded styles are recommended.

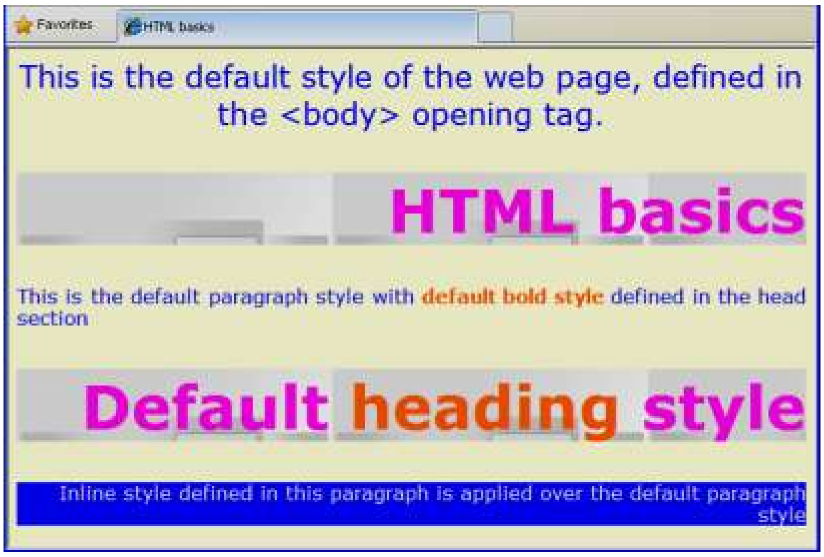
Embedded CSS styles set properties that will be applied as default for all tags they were defined for, throughout the entire web page. Embedded styles can be defined in the head section of the HTML page or in a separate Cascade Style Sheet (.css) file. When defined in the head section, the style HTML element is used with type attribute set to "text/CSS" inserted in the <style> opening tag. Each embedded style is defined using the following syntax:

```
tag_name
{ property:value; property:value;... property:value}
```

The order in which embedded styles are defined is not important. The code below defines embedded CSS styles for paragraph, bold and heading 1 HTML elements and exemplifies how inline styles properties overwrite embeded styles properties.

```
<html>
    <head>
        <title> HTML basics </title>
        <style type="text/css">
            p{
                font-size: 20px;
                text-align: justify; }
            b{
                color: RGB(255,128,10); }
            h1{
                background-image: url('img/back.jpg'); text-align: right;
                color: #FF00F0; }
        </style>
    </head>
    <body style = "background-color:lightyellow; text-align: center; font-
      family: verdana; font-size: 30px; color: blue">
        This is the default style of the web page, defined in the &lt;body&gt;
        opening tag.
        <h1> HTML basics </h1>
        <p> This is the default paragraph style with <b> default bold style
            </b> defined in the head section </p>
        <h1> Default <b> heading </b> style </h1>
        <p style = "text-align:right; background-color: blue;
            color:lightyellow"> Inline style defined in this paragraph is
            applied over the default paragraph style </p>
    </body>
</html>
```

The code above is displayed by Internet Explorer as in Figure 5.26. Note that the simple insertion of an HTML element that has an embedded CSS style defined (like p, b or H1) will display its content using that style. Still, if there is any inline style defined for an HTML element then it will be applied over the embedded one. The last paragraph, which contains an inline style too, uses the font size of 20 pixels defined in the embedded style. For the text-align property, the value right defined in the inline style overwrites the value justify defined in the embedded style and, therefore, the text is right aligned. The value blue for the background-color property defined in the inline style of the same paragraph overwrites the lightyellow value for the same property which defined in the inline style of the <body> opening tag. Therefore, this paragraph will display center aligned light yellow text on blue background.



This is the default style of the web page, defined in the <body> opening tag.

HTML basics

This is the default paragraph style with default bold style defined in the head section

Default heading style

Inline style defined in this paragraph is applied over the default paragraph style

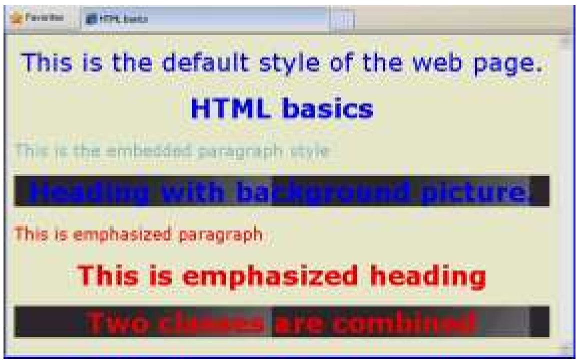
Figure 5.26 - Use of tag styles

The web developer can also define named CSS styles which can be applied to different html elements by simply using the name of the style. Like embedded styles, named styles can be defined either inside the style HTML element in the head section or in a separate CSS file. A named style can be general when its name starts with a simple dot (.name_of_style) in which case it may be applied to any HTML element inside the body, or restricted to a specific tag when its name starts with the name of the tag followed by dot (tag_name.name_of_style) in which case it may be applied only to HTML elements associated with the tag.

```
<html>
    <head>
            <title> HTML basics </title>
            <style type="text/css">
                p{
                    font-size: 20px;
                    text-align: justify;
                    color:lightblue; }
                .emphasize {
                    color: red;}
                h1.bgimage {
                    background-image: url('img/back.jpg');}
            </style>
    </head>
    <body style = "background-color:lightyellow; text-align: center; font-
        family: verdana; font-size: 30px; color: blue">
            This is the default style of the web page.
            <h1> HTML basics </h1>
            <p> This is the embedded paragraph style </p>
            <h1 class="bgimage"> Heading with background picture. </h1>
            <p class="emphasize"> This is emphasized paragraph </p>
            <h1 class="emphasize"> This is emphasized heading </h>
            <h1 class="emphasize bgimage"> Two classes are combined </h1>
      </body>
</html>
```

The code above exemplifies the use of all types of styles and the way that Internet Explorer displays it is represented in Figure 5.27. The default webpage style is defined as an inline style inside the <body> opening tag. In the style HTML element from the head, there are one embedded style, one general named style and one tag named style defined.

Applying named styles to HTML elements is done by using the class attribute in their opening tag, with the value given by the name of the style. The value of the class attribute may contain more names separated by space, which means that the HTML element will apply several styles in the order they are specified. The first heading uses the default page style and the first paragraph uses the embedded style applied over the default page style. The second heading applies the bgimage style and the second paragraph applies the emphasize style. Because bgimage is a tag restricted style, it will have no effect if inserted in a paragraph HTML element. emphasize is a general named style so it can be applied both to h1 and p HTML elements. The last heading applies two styles: the general emphasize style and the bgimage tag restricted style.



This is the default style of the web page.

HTML basics

This is the embedded paragraph style

Heading with background picture

This is emphasized paragraph

This is emphasized heading

Two classes are combined

Figure 5.27 - HTML styles

CSS styles can also be defined for HTML elements having the id attribute set. The value of this attribute represents actually a name given for the HTML element, which uniquely identifies this element over the entire web page. It is very useful when dynamic changes have to be applied to a certain HTML element identified by its unique id. For instance, the following paragraph

```
<p id = "firstparagraph"> First paragraph on the page </p>
```

may be identified by its id, which is "firstparagraph". None of the other HTML elements on the same page may have the same id.

A CSS style may be defined for a single element of the webpage with a certain id. The syntax for this type of style is:

```
#id {property:value; property:value;...property:value}
```

where id represents the id of the HTML element that the style was defined for. As an example, suppose that the first paragraph on the webpage should always be formatted with italic green text and indented with 30 pixels. The following style can be defined either in the head section of the document inside the style HTML element or in a separate CSS file:

```
<style>
    #firstparagraph
    {
        color: green;
        text-indent: 30 px;
        font-style: italic;
    }
</style>
```

This way, the paragraph with the id="firstparagraph" specified in the opening tag will apply the formatting element from the defined #paragraph style.

As mentioned above, styles can also be defined in a separate file having .css extension. This is useful when consistent styles have to be applied on all the web pages of a website.

```
p{ font-size: 20px; text-align: justify; color: blue;}
b{ color: RGB(255,128,10); font-size: 25 }
body{background-color: yellow;}
h1{ text-align: right; color: #FF00F0; }
.emphasize {color: red; font-weight:bold; border-style:solid; }
h1.bgimage {background-image: url('img/back.jpg'); text-align: left;}
#firstparagraph {color: green; text-indent: 30 px; font-style: italic; }
```

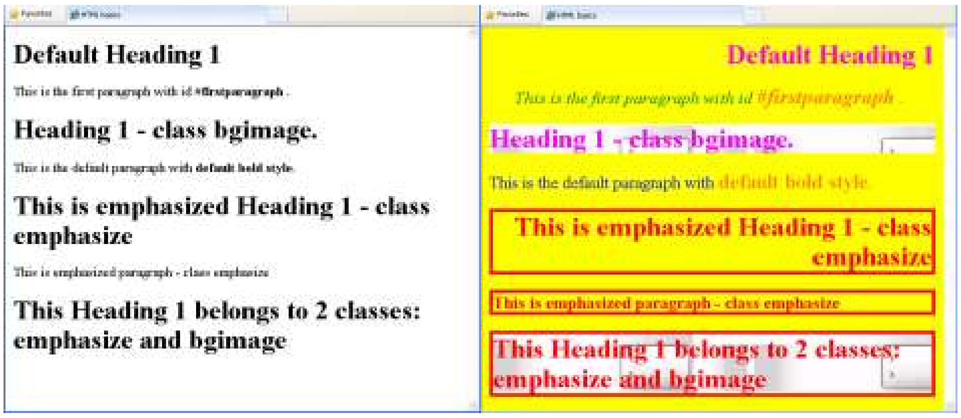
The code above represents an example of css content. In order to apply the styles defined in the code above, this code must be either included inside a style HTML element in the head section of the page or saved in a separate file, with .css extension. When styles are saved inside a style HTML element in the head section of a web page, they will be applied only to that page. If styles are saved in an external file then they will be applied to all web pages that include a reference to this file. Note that the CSS file will not contain any HTML tags (not even <html>, </html>, <head>, </head> or <body>, </body>). Suppose the code above is saved in the file *styles.css* . The reference to this external CSS file is created by inserting the <link> tag inside the head section of the page, with the attributes rel = "stylesheet", type = "text/css" and href = "styles.css" .

```
<html>
  <head>
        <link rel = "stylesheet" type = "text/css" href = "styles.css" />
        <title> HTML basics </title>
  </head>
  <body>
     <h1> Default Heading 1 </h1>
     <p id = "firstparagraph"> This is the first paragraph with id <b>
        #firstparagraph </b>. </p>
     <h1 class = "bgimage"> Heading 1 -  class bgimage. </h1>
     <p> This is the default paragraph with <b> default bold style</b>. </p>
     <h1 class = "emphasize"> This is emphasized Heading 1 - class
        emphasize </h1>
     <p class = "emphasize"> This is emphasized paragraph - class
        emphasize </p>
     <h1 class = "emphasize bgimage"> This Heading 1 belongs to 2
        classes: emphasize and bgimage </h1>
        </body>
    </html>
```

Figure 5.28 exemplifies the use of .css for a webpage containing the code above. The page represented on the left side of the figure misses the <link> tag from the head, which means that it is not linked to any css file, so will display everything with browser default settings. The page represented on the right side of the figure is linked to the css file, so it will apply the styles defined there.

In CSS (Cascading Style Sheets), multiple styles applied to the same element will cascade into one in the following order, where number 7 is the last style applied (has the highest priority):

1. Default styles of the browser;
2. Styles defined in external CSS file for a parent HTML element;
3. Styles defined in the head section of the page for a parent HTML element;
4. Styles defined in the opening tag of a parent HTML element;
5. Styles defined in external CSS file;
6. Styles defined in the head section of the page;
7. Styles defined in the opening tag of the HTML element (inline styles).


Default Heading 1

This is the first paragraph with id #firstparagraph .

Heading 1 - class bgimage.

This is the default paragraph with default bold style.

This is emphasized Heading 1 - class emphasize

This is emphasized paragraph - class emphasize

This Heading 1 belongs to 2 classes: emphasize and bgimage

Default Heading 1

This is the first paragraph with id #firstparagraph .

Heading 1 - class bgimage.

This is the default paragraph with default bold style.

This is emphasized Heading 1 - class emphasize

This is emphasized paragraph - class emphasize

This Heading 1 belongs to 2 classes: emphasize and bgimage

Figure 5.28 – Comparison between single webpage (left) and web page linked to .css file (right)

If the class specified inside the opening tag of an HTML element is not found, then the browser will not apply any style, ignoring the class attribute. The left webpage in Figure 5.28 has no link to the CSS file which contains style definitions for classes `emphasize`, `h1.bgimage` and for `#firstparagraph` id. This way, although the first paragraph has the `#firstparagraph` id, the first heading belongs to the `bgimage` class, the third paragraph belongs to `emphasize` class and the last heading belongs to both `emphasize` and `bgimage` class, these HTML elements are shown with the default browser setting, like they had no attributes in their opening tags.

On the other hand, the bold text from the first paragraph inherits the italic font style defined in the CSS file for the #firstparagraph id that the container paragraph has. For this text, first the paragraph there are four styles applied in this order:

1. Default browser style – makes the text bold;
2. Embedded paragraph style defined in the CSS file – makes the color of the text blue, the alignment justified and sets the font size to 30;
3. The style defined in the CSS file for the #paragraph id to which the container paragraph belongs – changes the text color to green, indents the text of the entire paragraph with 30 pixels and sets the font style to italic;
4. Embedded bold style defined in the CSS file – changes again the color to RGB(255,128,10) and increases the font size to 25.

This way, the bold HTML element from the first paragraph will display a text with the following characteristics: bold weight and italic style, size of 25 and color defined by RGB(255,128,10) combination.

5.6 Forms in HTML

It might happen that when you visit a webpage, you are asked to fill in some survey where you have to answer different question by choosing one of the suggested answers, checking one or more answers or writing text on how you feel about a certain product or a certain service. Also, when you registered on a website, usually you are asked for your e-mail address, for a password and maybe for some personal data like gender, home address, phone number and so on.

Collected information is usually sent to a server which processes it and takes appropriate measures: sends you an e-mail with registration data, adapts the webpage layout and content according to your preferences, records your data into a database for further analyses of survey's results. All this information is collected by using forms inserted on the visited web pages. A form contains different elements, called controls, which allow the visitor to interact with the webpage in different ways: entering some text, clicking a button, choosing or checking an option.

A form HTML element is defined using <form> and </form> tags and may include controls, text and markups (headings, paragraphs, lists) which define its layout. A form represents a container for all the controls defined between its opening and closing.

The following controls can be defined inside a form HTML element: - buttons; - checkboxes; - radio buttons; - menus; - text inputs; - file select; - hidden controls; - object controls.

Each control can be identified by the value of its name attribute. The scope of the name attribute of a control is the form it belongs to. This means that if there are two different controls with the same name but on different forms then they can be uniquely identified by their names throughout the entire webpage. Grouped controls, like checkboxes and radio buttons, may share the same name on the same form, the difference between them being made by their value attribute.

Each control has both a initial and a current value, represented by character strings. The initial value may be specified using the value attribute. When the form is loaded, the current value is set to the initial value and may be changed by the visitor of the page or by scripts. The initial value never changes so the control will be set back to this when the form is reset or the page is reloaded.

Most of the controls can be defined using <input .../> tag where the type attribute specifies the type of control used. There are ten controls that can be defined using the <input .../> tag, by specifying the following values of the type attribute:

- text: creates a text input control with only one line where the visitor may insert any text: e-mail address, name, phone number etc.;
- password: a text input control which displays a generic character like asterisk instead of each character;
- checkbox: an input control which has an on/off switch that may be toggled by the visitor. More checkboxes can be switched on, no matter they share the same name or not;
- radio: an input control which has an on/off switch. If more radio buttons share the same name on the same form, the visitor may select only one of them;
- submit: a button that submits the form when pressed by the visitor. All the controls names and their values will be sent for processing to the default processing agent specified in the <form> opening tag;
- image: a submit button decorated with an image;
- reset: a button that resets the form when pressed by the visitor. All the controls on the form will be set back to their initial values;
- button: a button that can be pressed by the visitor and that executes scripts depending on its defined events;
- hidden: controls that are not displayed but have associated values which are submitted with the form;
- file: controls that allow the visitor to select a file in order to submit its content.

```
<body>
  <form      action="mailto:dragos.vespan@ie.ase.ro"      method="post"
  enctype="text/plain">
    First name: <input type="text" name="firstname">
    Middle   name:   <input   type   =   "text"   name="middlename"
      style="width:25px" maxlength = "2"> <br>
    Last name: <input type="text" name="lastname"><br>
```

```
        <label for="email"> E-mail: </label>
        <input type="text" name="email" id="email" size = "40"><br>
        Password: <input type = "password" name = "password"> <br>
        <fieldset>
            <legend> Monthly income </legend>
            <input type="radio" name="income" value="unspecified" checked
               = "checked"> I don't want to specify it <br>
            <input type="radio" name="income" value="low"> 0 - 700 <br>
            <input type="radio" name="income" value="medium"> 701 -
               1700 <br>
            <input type="radio" name="income" value="high"> &gt; 1701 <br>
        </fieldset>
        I want password to be sent by e-mail <input type="checkbox"
            name="notifications" value="email" checked = "checked"><br>
        I want to receive the newsletter by e-mail<input type="checkbox"
            name="notifications" value="newsletter"><br>
        <label >
            I want to receive special offers by e-mail
            <input type="checkbox" name="notifications" value="offers"><br>
        </label>
        <label for="img"> You can submit the form: </label>
        <input type = "image" src="img/button.jpg" id="img" alt="Submit now">
        <br> Please attach your CV if available: <br>
        <input type = "file" value="Choose a file" name="file" size = "40">
        <input type = "hidden" name="confirmation" value="yes"> <br>
        <input type="button" value = "Close the window" name="close"
            onclick='window.close()'>
        <input type="submit" value="Send">
        <input type="reset"> <br>
    </form>
</body>
```

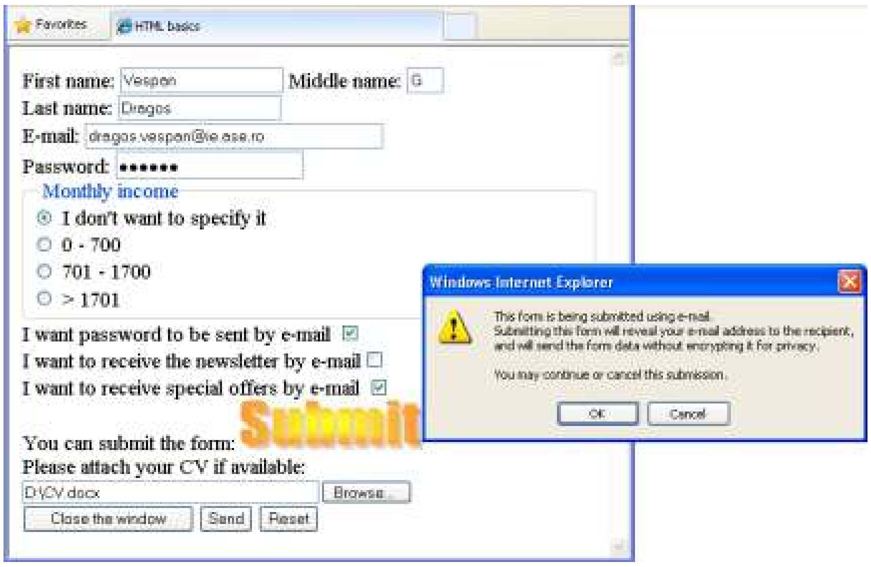
The code above exemplifies the use of all input types on one form and is displayed by Internet Explorer as in Figure 5.29. By default, the width of all controls displaying a textbox (text, password or file) is about 20 characters long. The width may be changed by either setting the size attribute to a value approximating the number of characters that should fit into the control (the length of the email input is set to display about 40 characters) or by specifying the width in pixels using the style attribute (the middlename input width is set to 25 pixels). The maximum number of characters that the visitor may write into a text input can be specified using the maxlength attribute: the middlename text input will allow the visitor to enter a maximum of 2 characters.

Some controls have implicit labels given by the value of the value attribute: the text written on the button input is "Close the window" and the text written on the submit input is "Send". The label HTML element may be used to attach information to controls that do not have implicit labels associated (like text, radio or checkbox). Each label HTML element is associated explicitly or implicitly to exactly one control on the form. The explicit association is made by using for attribute in the <label> opening tag. The value of this attribute must be the same as the value of the id attribute defined in the opening tag of the associated control. The email text input and img image input have explicit labels attached. The implicit association is made by inserting the associated control between <label> and </label> tags. Only one control may be inserted inside a label HTML element. The notifications checkbox input with value equal to offers has an implicit label attached. Labels may be rendered by internet browsers in different ways: for instance, the read by speech synthesizers may identify controls through their associated labels.

The fieldset HTML element may be used for grouping visually the controls that belong to the same category. The grouped formed by all the controls defined between <fieldset> opening tag and </fieldset> closing tag will be surrounded by a line: in the example above, all the radio input controls are defined inside a fieldset HTML element so that the visitor easily identifies the group they form. The content of the legend HTML element inside the fieldset represents the description of the entire group and is displayed on the line that surrounds it.

Both radio and checkbox input controls may be selected or checked by default if the checked attribute with the value set to checked is inserted in their opening tag. The notifications checkbox input control with the value email is checked by default and may be unchecked by the visitor. Checking either or both of the other two checkbox controls will not affect its state. The visitor may check or uncheck all the checkbox controls. The income radio input control with the value low is also selected by default. It will be automatically deselected when the visitor selects another income radio input control. There is now way the visitor can select two income radio input controls at once or deselect all of them.

The image input control will insert into the webpage the image from the file specified in the src attribute. This image will behave by default like a submit button and will submit the coordinates of the position inside the button where the mouse pointer was clicked: distances, in pixels, from the left margin (x) and from the upper margin (y) of the button. The value of the alt attribute is shown as a small popup when the mouse pointer is over the button.



First name: Vespan Middle name: G
Last name: Dragos
E-mail: dragos.vespan@ie.ase.ro
Password: ●●●●●●
Monthly income
[✓ I don't want to specify it]
0 - 700
701 - 1700
> 1701
I want password to be sent by e-mail ✓
I want to receive the newsletter by e-mail ☐
I want to receive special offers by e-mail ✓
You can submit the form: Submit
Please attach your CV if available:
D:\CV.docx Browse...
Close the window Send Reset

Windows Internet Explorer

This form is being submitted using e-mail.
Submitting this form will reveal your e-mail address to the recipient, and will send the form data without encrypting it for privacy.

You may continue or cancel this submission.

OK Cancel

Figure 5.29 - Input controls on HTML forms

The file input control allows the visitor to visually choose a file from the computer by clicking the Browser button. The hidden input control is not shown to the visitor so its value cannot be changed. Still, its name and default value are submitted to the processing agent. The button input control has two attributes (value and name) and one event (onclick) defined in the opening tag. The value attribute sets the text displayed on the button and the onclick attribute sets the action to be done when the visitor clicks on the button (closing the current window).

When the visitor submits a form (by clicking the submit button) the browser processes it in four steps:

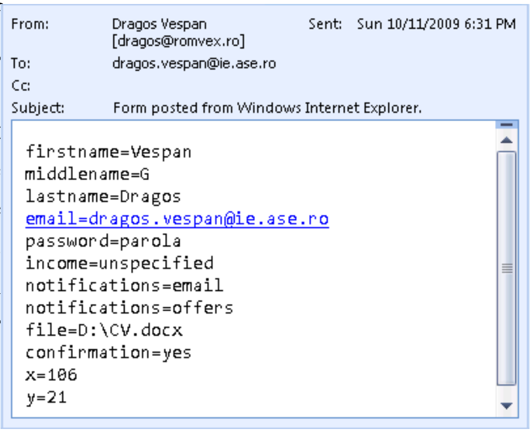
1. identifies the successful controls;
2. builds a form data set;
3. encodes the form data set;
4. submits the encoded form data set.

A successful control is a control valid for submission which has a current value associated with its name as part of the form data set. Thus the name attribute is required for all types of controls except submit and reset input controls or buttons. Also, if a control does not have a current value when the form is submitted, it might not be treated as a successful control. From the example in Figure 5.29, the following controls are successful:

- all text input controls and the password input control with the values written by the visitor;

- the first radio input control with the value unspecified;
- the first and the third checkbox input controls with the values email and offers;
- the file input control with the value D:\CV.docx;
- the confirmation hidden input control;
- the img input control which was clicked at coordinates: x=106 and y = 21.

A form data set is a sequence of *controlname=currentvalue* pairs belonging to all successful controls. This is encoded according to the content type specified in the enctype attribute inside the <form> opening tag. The text/plain value of this attribute forces the browser to submit the names and values of the successful controls in clear text, without any encryption.


From: Dragos Vespan [dragos@romvex.ro] Sent: Sun 10/11/2009 6:31 PM
To: dragos.vespan@ie.ase.ro
Cc:
Subject: Form posted from Windows Internet Explorer.

```
firstname=Vespan
middlename=G
lastname=Dragos
email=dragos.vespan@ie.ase.ro
password=parola
income=unspecified
notifications=email
notifications=offers
file=D:\CV.docx
confirmation=yes
x=106
y=21
```

Figure 5.30 - E-mail sent using form submission

The method attribute of the form HTML element specifies the HTTP method used to send the form to the processing agent. This attribute may take two values: get and post. When the get value is set and the action is a http URI, the browser takes the value of the action attribute, appends a ? to it and then appends the encoded form data set. The post value of the method attribute forces the browser to include the form data set in the body of the form before sending it to the processing agent specified in the action attribute.

In the example from Figure 5.29, the value of the action attribute is mailto:dragos.vespan@ie.ase.ro which means that the processing agent is the default e-mail program. This will send the form data set in plain text to the specified e-mail address, using the default e-mail account. Internet Explorer browser warns the visitor about the e-mail being sent. The e-mail that the recipient receives when the form is submitted is shown in Figure 5.30.

```
<body>
  <form    action="http://info.ase.ro/prog/analyze"    method="get"
    enctype="multipart/form-data">
    <label> Select your education:
      <select  name="education">
        <option selected value="Unspecified">Not specified </option>
        <option value="highschool">A-level degree</option>
```

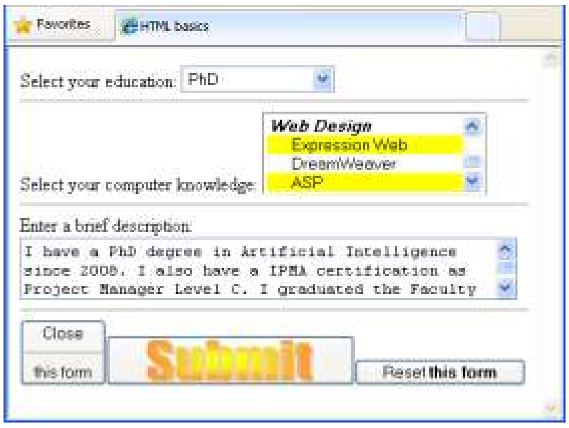
```
            <option value="university">Bachelor degree</option>
            <option value="master">Master degree</option>
            <option value="doctorate" label="PhD">PhD degree</option>
         </select>
      </label> <hr>
      <label> Select your computer knowledge:
         <select name="languages" multiple size="4">
            <option selected label="none" value="none">None</option>
            <optgroup label="Computer programming">
               <option value="VB">Visual Basic</option>
               <option value="VC">Visual C++</option>
               <option value="VJ">Visual Java</option>
            </optgroup>
            <optgroup label="Web Design">
               <option value="ExW"> Expression Web </option>
               <option value="DW"> DreamWeaver </option>
               <option value="ASP">ASP</option>
               <option value="PHP">PHP</option>
            </optgroup>
         </select>
      </label> <hr>
      <label>
         Enter a brief description:
         <textarea name="description" rows="3" cols = "50">
         </textarea>
      </label> <hr>
      <button type="button" onclick="window.close()"> Close <hr> this
      form </button>
      <button type="submit"> <img src="img/button.jpg"> </button>
      <button type="reset"> Reset <b> this form </b> </button>
   </form>
</body>
```

The code above exemplifies the use of select, optgroup, option, textarea and button controls and the Figure 5.31shows the way it is rendered by Internet Explorer.

The select control allows the visitor to select one or more elements from a list of available items. Each one of these elements is represented by an option HTML element. The name of each element is given by the content of the corresponding option HTML element. Still, if the label attribute is specified in its opening tag, then the value of this attribute will be displayed instead of the content.

It is mandatory for a select control to include at least one option element. Also, the name attribute must be present inside the opening tag of the select control, in order to be submitted along with the value of the selected option.

The value of the size attribute sets the number of visible rows displayed by the select control. If the size attribute is not specified then, by default, the control is shown as a dropdown list. If the value of the size attribute is greater than one then the select control is displayed as a scrolling list box.



Select your education: PhD

Select your computer knowledge: *Web Design* Expression Web DreamWeaver ASP

Enter a brief description:

```
I have a PhD degree in Artificial Intelligence
since 2008. I also have a IPMA certification as
Project Manager Level C. I graduated the Faculty
```

Figure 5.31 - Other controls used in HTML forms

The first select control in the code above has only the name attribute specified inside its opening tag. Thus, it will be displayed as a dropdown list with five elements from which the visitor can select only one.

An option element may be preselected if it includes the selected attribute (with no value) inside its opening tag. This way, if the visitor does not interact with the select control that contains it then the preselected element will be submitted by default. It is wrong to have multiple items preselected for a select control that does not include the multiple attribute.

The first option element of the first select control is preselected and, because the control is not a multiple selection one, it will be automatically deselected when the visitor chooses another option element. In Figure 5.31 the PhD option is selected so the pair education:doctorate will be submitted to the processing agent. Note that for this option HTML element, the value of its label attribute is displayed instead of its content.

If the multiple attribute (with no values) is present in the opening tag of a select control, then the visitor may choose several options from the list at once, by holding down the Ctrl key and clicking the right options with the mouse. More, such a select control may contain more preselected options which will be deselected if the visitor clicks on them.

If there are too many options to choose from the list, they can be grouped by categories so that the visitor visualizes and identifies them better. The optgroup HTML element is used for grouping logically the options of a select control. The value of the label attribute specified in the optgroup element's opening tag specifies the category to which the contained options belong. Also, an optgroup element cannot be selected.

The second select control from the code above contains two groups, one with three options and one with four options. Also, it contains another separate option with none value which is preselected. On the webpage in Figure 5.31 three options were selected: VB, ExW and ASP. This way, three pairs were also generated in order to be submitted: languages=VB&languages=ExW&languages=ASP, Figure 5.32. If no option element is selected inside the select control then the control is not succesfull and neither the control name nor its values will be submitted to the processing agent.

If there is more text that the visitor should fill in, like his own description in a few phrases for instance, then the textarea control may be used. This control allows the user to enter multiple lines of text at once, and to visualize all the written text by using the attached scroll bars. The attributes that can be set within its opening tag are name, rows and cols. The name attribute along with the content of the textarea control inserted by the visitor are submitted to the processing agent. The rows attribute specifies the number of rows visible in the control. If more rows of text are written by the visitor, then the browser will attach vertical scroll bars to the control. The cols attribute specifies the width in average character widths. The browser wraps the text of the long lines avoiding this way the use of horizontal scroll bars. The textarea control from the example in Figure 5.31has a width of 50 characters and displays three lines of text at once.

Instead of the button input control, the button control may be used for placing buttons on the form. The functionality is the same and the difference consists in that the button control provides richer rendering possibilities due to the fact that it accepts content.

Both value and name attributes may miss from the opening <button> tag. If the name attribute is present then the value of the button will be submitted to the processing agent and, if there is no value attribute specified, then the content of the button control will be submitted instead. The type attribute is mandatory and may take one of the three values: button, submit and reset, with the same functionalities as the input button controls.

Almost any type of HTML content may be inserted between <button> opening tag and </button> closing tag. The first button element in Figure 5.31contains text separated by a horizontal rule. The content of the second button element is an image on which the visitor has to click in order to submit the form. The third button control contains bold text.

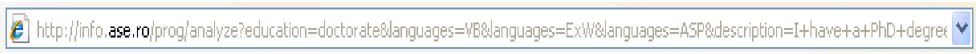

http://info.ase.ro/prog/analyze?education=doctorate&languages=VB&languages=ExW&languages=ASP&description=I+have+a+PhD+degre

Figure 5.32 - URI submitted by the form

The value of action attribute of the form control presented in Figure 5.31 represents the application called analyze which is found in the prog folder on http://info.ase.ro. The method attribute has the get value which will make the browser append ? after to the URI specified by the value of the action attribute followed by the form data set. This URI will be sent to the analyze application, responsible for processing the form data set. The form data set contains a sequence of control_name:value pairs for all successful controls in the form. If a control has more values selected, then its name will make a pair with each selected value inside the form data set. When the multipart/form-data encoding type is used, the control_name:value pairs inside the form data set are separated by & and the spaces contained in the values of the controls are replaced by +, like in Figure 5.32.

5.7 Differences between HTML and XHTML

XHTML stands for Extensible HyperText Markup Language and represents a stricter and cleaner version of HTML. XHTML is HTML defined as an XML application and represents a W3C recommendation, being supported by all the browsers.

XHTML represents a combination between XML, designed for describing data, and HTML, designed for displaying data. This way, an XHTML document is in fact a HTML document where everything is marked up with respect to strict conditions.

The main differences between HTML and XHTML are:

1. in HTML elements must be closed strictly in the reverse order they were opened. Although the syntax
   `<b><i>bold and italic text </b></i>`
   is allowed in HTML, it is wrong in XHTML and the correct version is
   `<b><i>bold and italic text </i></b>.`
   The HTML sequence
   `<b> bold <i> and italic </b> and only italic </b>`
   will be rewritten in XHTML as
   `<b>bold <i> and italic </i></b><i> and only italic </i>.`
   Although HTML allows non empty elements like <p> to miss the ending tag. In XHTML this is not possible.
2. Empty elements must be closed in XHTML by using / in their opening tag:
   `<img src="img/sunset.jpg" />`
   A trailing space should be added before /> .
3. XHTML does not allow non-empty elements to be defined as empty by inserting / in their opening tag: `<p />` is wrong and `<p></p>` is correct.

4. In XHTML all element names and attributes must be written with lowercase. Although <HTML> and <A HREF="#second" > are allowed in HTML, in XHTML they have to be written as <html> and <a href="#second">.
5. In XHTML, all attributes must be written between single or double quotes. Although

   <button type=button onclick=window.close()>

   is allowed in HTML 4.01. In XHTML the statement must be rewritten as

   <button type="button" onclick="window.close()">
6. In XHMTL, all attributes must be represented as attribute_name="attribute_value" pairs. Although

   <option selected value="Unspecified">

   is correct in HTML, in XHTML it has to be rewritten as

   <option selected="selected" value="Unspecified">
7. The HTML name attribute for the elements a, applet, form, frame, iframe, img and map was replaced by the id attribute in XHTML.
8. The following nesting prohibitions are defined in XHTML:
   - a must not contain other a elements;
   - pre must not contain img, object, big, small, sub or sup elements;
   - button must not contain input, select, textarea, label, button, form, fieldset, iframe or isindex elements;
   - label must not contain other label elements;
   - form must not contain other form elements.
9. When specifying the language of an element in XHTML, both lang and xml:lang attributes must be used, like this:

   <div lang="en" xml:lang="en"> Hello! </div>
10. The display of the ampersand character & must always be done by using the &amp; character entity. For instance, when the a element is used with href attribute referring to a CGI script with parameters, instead of

    http://info.ase.ro/prog/analyze?education=doctorat&languages=VB

    which is valid in HTML, the string should be written as

    http://info.ase.ro/prog/analyze?education=doctorat&amp;languages=VB
11. All XHMTL documents must have a DOCTYPE declaration and html, head, title and body elements must be present.

Contents

6 VBScript

6.1 Introduction

VBScript is a subset of VBA and it allows usage of most familiar functions of these one. Microsoft says that VBScript is integrated with World Wide Web browsers (more exactly with Internet Explorer versions of Microsoft) and designed to work with ActiveX controls and other objects embedded in active HTML documents. VBScript can be used as Web scripting client in Microsoft Internet Explorer and as Web server scripting in Microsoft Internet Information Services. Web server scripting is accessible at Windows command prompt by one of the commands CScript.exe (DOS interpreter) or by WScript.exe (windows interpreter). In the following paragraphs the Web scripting client introduced.

In an HTML page the code modules are supported through the <SCRIPT></SCRIPT> tag. Each script section forms an independent code module that may have its own variables, functions and subroutines (they are similarly to the standard *.bas* modules found in Visual Basic).

The forms are created using the <FORM></FORM> tag (not the form control in Visual Basic) and they are not visible as separate windows in the application. The forms are ways to group controls together for the purpose of addressing their properties and methods in code or to submit data to back-end process.

6.2 Using and placing VBScripts in a HTML page

The script blocks can be placed and used anywhere within a HTML page but we can obtain some benefits when placing in the heading section or body section (as block script or inline script) of an HTML page. In the following paragraphs are introduced some examples of using (and placing) VBScripts in a HTML page.

6.2.1 VBScript in the body of the HTML file

VBScript in the body of the HTML page will be executed when the page loads. Generally, the scripts in the body, will generate the content of the page.

Example:

```
<html>
 <head>
   <title>
        Page containing VBScript
   </title>
 </head>
 <body>
  <script TYPE="text/vbscript" LANGUAGE="VBScript">
     document.write("This text is displayed by a VBScript!")
  </script>
 </body>
</html>
```

Comments:

The tag <script> have the attribute TYPE, that specifies the script type (VBScript in our case). The attribute LANGUAGE indicates the scripting language.

This script composed by a single command that displays inside the page the text: "This text is displayed by a VBScript!".

If the script is included in a comment tag (<!--) then the browsers that do not „know" (interpret) VBScript will not display (it skips over) in the page the script text (script source), as shown in the folowing sample:

```
<!--
document.write("<i>"+"This text is displayed by a
VBScript!"+"<\/i>")
//-->
```

The browsers that know VBScript will process the script, even this included in a comment line/block.

The string „//", comment in VBScript, tell to the browser do not process the line „-->". We can not use the sintax „//<!--", for the comment beginning tag, because a browser that do not interpret VBScript will display that string „//".

6.2.2 VBScript in heading

If we want be sure that the script executes before displaying any element in the page we can include this in the heading part of the HTML page (file). The VBScript in the head section will be executed when called, or when an event is triggered.

Example:

```
<html>
 <head>
   <title>
       Page with VBScript
   </title>
  <script type="text/vbscript">
     document.write("This text is displayed by a VBScript!")
  </script>
 </head>
 <body>
```

```
   <P>  This text must appear in the page after the execution of the
VBScript.
 </body>
</html>
```

The number of scripts placed in the head section and in the body of a HTML page is unlimited.

6.2.3 Inline VBScript

The inline scripting code can be associated to forms objects to respond to the events of that objects.

Example:

```
<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">
    ' replace the message that follows with the code you want
    ' be executed when clicking the button
    MsgBox "Button Pressed!"
  </SCRIPT>
</FORM>
</BODY>
</HTML>
```

In this example the button object called „Button1" has an associated script for the click event (EVENT="onClick"). When the user presses the button the script executed.

6.3 Variables and Constants

6.3.1 Variables

Variables are named storage locations that can contain data that can be modified during script running. The variables are the memory cells used for storing forms input data and/or its computational results. The only data type accepted is as the Variant in VB6, which can contain any kind of data. Table 6.1 shows the data subtypes accepted. The declaration of variables is realized by using the Dim or ReDim statement.

The naming of variables in VBScript uses the rules:

1. An identifier must begin with a letter;
2. Can't be longer than 255 characters;
3. Can't contain embedded period or embedded type declaration character;

4. Must be unique in same scope (the range from which the variable can be referenced).

The number of variables per procedure is limited to 127 (an array counts as one variable) and each script is limited do not have more than 127 module-level variables. The length of the time a variable exists is its lifetime. A script level variable's lifetime begins when its declaration statement is encountered as procedure begins, and ends when the procedure concludes.

The array elements can have how many dimensions required, each dimension defined by separating from previous with a comma, as in this example Dim MatrixAlpha(10,20,30) that defines a three dimensional array. The addresses of elements (the index) start from 0 for every dimension. In the previous example the number of elements of the array called MatrixAlpha is 11x21x31.

Examples:

```
Dim Pi,CompanyName,EndDate
rem Assigning values
Pi=3.14
CompanyName=”Media Advertising, Inc.”
EndDate=#12-31-2007#
```

Table 6.1. Possible Data Subtypes for a Variant

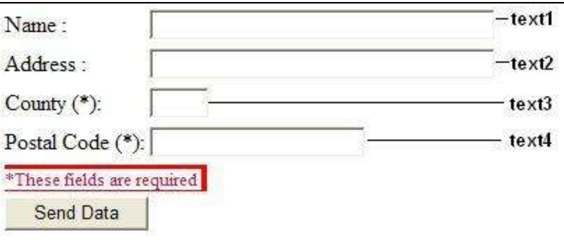
Subtype	**Meaning**
Empty	Variant is un-initialized. Value is either 0 for numeric variables or a zero-length string ("") for string variables.
Null	Variant intentionally contains no valid data.
Boolean	Contains either True or False.
Byte	Contains an integer in the range 0 to 255.
Integer	Contains an integer in the range -32,768 to 32,767.
Single	Contains an integer in the range -2,147,483,648 to 2,147,483,647.
Long	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.
String	Contains a variable-length string that can be up to approximately

	2 billion characters in length.
Date (Time)	Contains a number that represents a date between January 1, 100 to December 31, 9999.
Object	Contains an object.
Error	Contains an error number.

Example:

```
<SCRIPT TYPE="text/vbscript">
<!--
rem Defines two variant variables
Dim DegreesFahrenheit, DegreesCelsius
Rem Defines an array A of 100 elements and an array B of an unspecified
dimension (used for dynamic resizing)
Dim A(100), B()
Rem Dynamically resizes the array B
ReDim B(50)
-->
</SCRIPT>
```

Example:
This example shows how you can reference and use variables defined in forms (as text box objects) and assign a value. The procedure cleans the text boxes in the form when the web page containing the form is loaded.



The form and his internal elements

```
<HTML>
<BODY bgColor=white>
<TABLE>
<TR>
<TD valign=top>            </TD>
<TD valign=top>
<TABLE> <tr>  <td colspan=2></td> </tr>
<TR>  <TD>Name :</TD><TD><INPUT id=text1 name=text1
          style="WIDTH: 248px; HEIGHT: 22px" size=32></TD>
</TR>
<TR>
  <TD>Address :</TD><TD><INPUT id=text2 name=text2
          style="WIDTH: 248px; HEIGHT: 22px" size=32></TD>
</TR>
<TR>
```

```
 <TD>County&nbsp;(*):</TD><TD><INPUT id=text3 name=text3
                      style="WIDTH: 43px; HEIGHT: 22px"></TD>
</TR>
<TR>
 <TD>Postal&nbsp;Code (*):</TD><TD><INPUT id=text4 name=text4 width =
                      "10"></TD>
</TR>
<TR>
 <TD colSpan=2><FONT
          style="BACKGROUND-COLOR: red"><FONT
          style="BACKGROUND-COLOR: #ffffff" color=crimson size=2><FONT
          style="BACKGROUND-COLOR: white">*These fields are
          required</FONT>&nbsp;</FONT> </FONT>       </TD>
</TR>
<TR>
 <TD colSpan=2><INPUT id=button1 name=button1 type=button value=" Send
Data "
     LANGUAGE=vbscript onclick="return button1_onclick()"></TD>
</TR>
</TABLE>
</TD>
</TR>
</TABLE>
</BODY>
<SCRIPT LANGUAGE=VBScript>
rem When the page is loaded, clear the text boxes
sub Window_onLoad
          rem Reset edit boxes
          text1.value = ""
          text2.value = ""
          text3.value = ""
          text4.value = ""
end sub

</SCRIPT>
<SCRIPT LANGUAGE=JScript>
</SCRIPT>

</HTML>
```

Arrays. The array is a data structure in which we store data items of the same type. The massive (or array) is used for representing vectors and matrix in the internal memory of computer. An array can be thought of as a simple variable with an *index*, or *subscript*, added. We can access each individual item (called *array element*) stored in the array. If **A** is an array of **n** elements, we can write A(*expr*) , where *expr* is an integral expression, to access an element of the array. We call *expr* subscript, or index, of A. The expression A(i) can be made to refer to any element in the array by assignment of an appropriate value to the subscript i, that mean a value that range from 1 to n (or, depending on the programming language used and declaration of the array, from 0 to n-1). The elements of the massive are called indexed variables, because the addressing of an element is depending on its position into the array. This position is represented by means of an index value for every element.

Some programming languages admit multiple variables massive. In the internal memory, massive are usually represented in a linear form, in a contiguous zone which is divided into other zones of the same length for every element.

The address of an element is obtained with the beginning address [**A**] and the displacement [**d**] of the considered element. The displacement can be obtained knowing the length of an element and the order of the considered element into the massive. A massive has a finite number of elements, all of them having the same type, length and attributes. In almost programming languages a massive is an internal structure, with finite cardinality and homogeneous data, having a proper linear and statistic structure law.

In economy, there are many problems that need to be solved by using the processing of data series, dynamic series, series of intervals, results of the recording of some statistic studies in phenomenon's evolution. Many other types of problems may be solved by data which are represented as vectors and matrices.

Generally, all the elements in an array have the same data type but, in VBScript, since the data type is Variant the individual elements may contain different kinds of data (object, string, numbers, and so on). In VBScript there are two types of arrays: *fixed-size array* which always remains the same size, and *dynamic array* whose size can change at run time.

Fixed-size arrays. An array is declared in VBScript with the same Dim sentence:

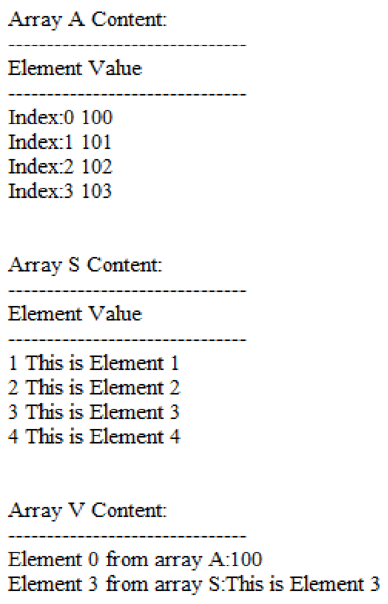
Dim *arrayname*(*n*)

where:

- *arrayname* is the desired name for the array;
- *n* is the maximum number of elements for the array. The index takes values from 0 to n-1 (that means n elements);

A multidimensional array is declared by specifying many dimensions, for example *Dim M(9,6)* defines a two dimensional 10-by-7 array of variant.

To retrieving elements and assigning values we must specify each index value. For example, M(1,1) designates the element from line 1, column 1. It is possible to define multidimensional arrays by defining the values for each desired dimension (with



```
Array A Content:
------------------------------
Element Value
------------------------------
Index:0 100
Index:1 101
Index:2 102
Index:3 103

Array S Content:
------------------------------
Element Value
------------------------------
1 This is Element 1
2 This is Element 2
3 This is Element 3
4 This is Element 4

Array V Content:
------------------------------
Element 0 from array A:100
Element 3 from array S:This is Element 3
```

The output for array operations example

one subscript per dimension).

The following code sequence illustrates the way to use arrays:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=windows-1252" http-equiv="Content-Type" />
<title>VBScript Arrays Example </title>

<script language="vbscript">
 function ArrayOperations()
   Dim i, j, xval ' The index variables
   Dim arrayA(3) 'An array of 4 integers
   Dim arrayS(3) 'An array of 4 strings
   Dim arrayV(2,2)'Declares an array of two elements
   xval="&nbsp;nbsp;nbsp;nbsp;nbsp;nbsp;nbsp;"

  For i = 0 To 3
       arrayA(i) = 100 + i 'Assigning a value to element i
       'document.write(arrayA(i) &  "<br>")
   Next

   Rem Printing the heading of the table
   Document.write("Array A Content:<br>")
   Document.write("------------------------------<br>")
   Document.write("Element Value<br>")
   Document.write("------------------------------<br>")

   Rem Printing the content of arrayA elements
   For j = 0 To 3
     Document.write("Index:" & j & " " & arrayA(j) & "<br>")
   Next

   For i = 0 To 3
       xval=i+1
       arrayS(i) = "This is Element " & xval
       Rem Assigning a value to element i
   Next

   Document.write("<br><br>Array S Content:<br>")
   Document.write("------------------------------<br>")
   Document.write("Element Value<br>")
   Document.write("------------------------------<br>")

   For i = 0 To 3
     xval=cstr(i + 1) & " " & arrayS(i) & "<br>"
     Document.write(xval)
   Next
  Document.write("<br><br>Array V Content:<br>")
  Document.write("------------------------------<br>")
  arrayV(1,2) = arrayA(0)
  arrayV(2,1) = arrayS(2)
  Document.write("Element 0 from array A:" & arrayV(1,2) & "<br>")
  Document.write("Element 3 from array S:" & arrayV(2,1) & "<br>")
 end function

</script>
</head>

<body>
<input name="Button1" type="button" value="Show Arrays"
onclick="ArrayOperations()"/>
</body>

</html>
```

The command button labeled Show Arrays runs, when clicked, the code sequence defined inside of the <script> tag and produces the output shown in figure x.y.

Dynamic Arrays. A dynamic array can be resized at any time and is usual when we don't know exactly the number of elements (for example, the number of documents opened in a session).

To create a dynamic array follow the rules:

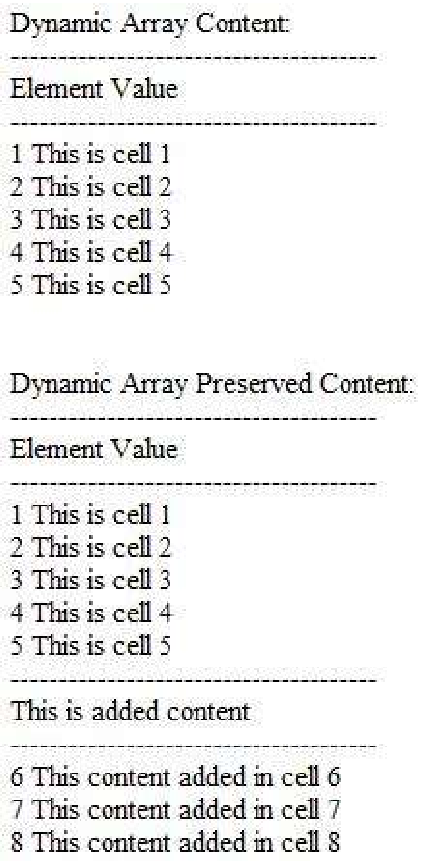
1st. Declare the array with ReDim statement, for example ReDim DynamicArray(5)

2nd. Allocate the extended number of elements with a ReDim statement, for example ReDim DynamicArray(8), that creates a new array of 9 elements. The ReDim statement can appear as many time as necessary. Each time we execute the ReDim statement all values already stored in the array are lost. We can preserve the existing content by using the Preserve keyword ReDim DynamicArray(8), that adds the elements 6, 7, and 8 to the existing array and preserve the existing ones.

The following example shows the way to use dynamic arrays and preserve the content:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=windows-1252" http-equiv="Content-Type" />
<title>VBScript Arrays Example </title>
<script language="vbscript">
```

```
 function DynamicArrayOperations()
   ReDim DynamicArray(5)
   Dim i ' the index name

   document.write("Dynamic Array
Content:<br>")
   document.write("----------------------------------
-<br>")
   document.write("Element
Value<br>")
   document.write("----------------------------------
-<br>")
   For i = 1 To 5
      DynamicArray(i) = "This is cell " & i
      document.write(i & " " & DynamicArray(i)
& "<br>")
   Next
   ReDim Preserve DynamicArray(8)
   document.write("<br><br>Dynamic Array
Preserved Content:<br>")
   document.write("----------------------------------
-<br>")
   document.write("Element
Value<br>")
   document.write("----------------------------------
-<br>")
   For i = 1 To 5
      document.write(i & " " & DynamicArray(i)
& "<br>")
   Next
   document.write("----------------------------------
-<br>")
   document.write("This is added content<br>")
   document.write("-----------------------------------<br>")
   For i = 6 To 8
      DynamicArray(i) = "This content added in cell " & i
      document.write(i & " " & DynamicArray(i) & "<br>")
   Next

  end function
</script>
</head>
<body>
<input name="Button1" type="button" value="Show Arrays"
onclick="DynamicArrayOperations()"/>
</body>
</html>
```



```
Dynamic Array Content:
-----------------------------------
Element Value
-----------------------------------
1 This is cell 1
2 This is cell 2
3 This is cell 3
4 This is cell 4
5 This is cell 5


Dynamic Array Preserved Content:
-----------------------------------
Element Value
-----------------------------------
1 This is cell 1
2 This is cell 2
3 This is cell 3
4 This is cell 4
5 This is cell 5
-----------------------------------
This is added content
-----------------------------------
6 This content added in cell 6
7 This content added in cell 7
8 This content added in cell 8
```

The output for dynamic array operations example

The command button labeled Show Arrays runs, when clicked, the code sequence defined inside of the <script> tag and produces the output shown in figure x.y.

6.3.2 Constants

Constants can appear as such anywhere as literals, intrinsic constants available in the VBScript programming environment or in other Windows applications, or as declarations in the declarative part of the program. Literals can be used only once in the code; if you want use many times you must declare them each time. The constants are like a cross between literals and variables: they have a single never-changeable value just as literals but must be declared and assigned with the value just as variables.

The user can define his constants by following the procedure:

1) Define the memory variable for constant with the Const keyword (instead of Dim);
2) Assign the value, as literal value, to the variable;
3) Use the name of the variable anywhere (in his scope) is required the literal value assigned to it.

Examples:

Constant	Type
"Welcome to the information century!"	string
$25,000.00	currency
3.14	positive real number
-123	negative integer number
0.123e+3	number written in the scientific notation
“11/12/2009”	date

Constants can be defined as a declaration statement by using the syntax:

Const *constantName=expression*[,…]

where:

- *constantName* is an user identifier defined by following the naming rules;
- *expression* an expression evaluated to an agreed data type whose evaluation is considered the default value for the constant.

Examples:

```
Const Pi = 3.14159
Const Vat = 0.19, Star = “★”
```

A declared constant is a named storage location that contains data that can not be modified during the program execution. The most used constants are number constants and string constants. A string constant is sequences from 0 to 1024 characters enclosed in quotes.

6.4 Assignments and expressions

Assignments

The general syntax for assignment is:

variable = *expression*

The interpretation of this is: the *variable* before the assignment operator is assigned a value of the *expression* after it, and in the process, the previous value of *variable* is destroyed.

An assignment statement stores a literal value or a computational result in a variable and is used to perform most arithmetic operation in a program.

Expressions

An *expression* can be an expression on character string, a logical expression or arithmetic expression. It can be a variable, a constant, a literal, or a combination of these connected by appropriate operators (table 6.2).

1. An *expression on character string* can be built (in VB but not only) using:

- the concatenation operator: &
- intrinsic functions for extracting substrings from a string variable or string constant such as:

  Right(*string*,*number_of_characters*) - extracting substring from the end

  Left(*string*,*number_of_characters*) - extracting substring from the beginning
- functions that manipulate strings:

  Cstr(*expression*) – convert the expression in a character string;

  Lcase(*string_expression*) – convert the string in lower case;

  Ltrim(*string*), Rtrim(*string*), Trim(*string*) – eliminates the spaces (trailing) from left (leading blanks), right and, respectively left-right;

  Str(*number*) – converts number in string;

  Ucase(*string*) – converts string to uppercase.

2. A *logical expression* can be:

• **simple**, with the general syntax:

<*variable*>[<*relation_operator*><*variable*>]

or

<variable>[*<relation_operator><constant>*]
<relation_operator>::=<|<=|>|>=|=|<>

• **complex**, with the general syntax:

e_1 **Eqv** e_2 - equivalence;
e_1 **Imp** e_2 - logical implication;
e_1 **Xor** e_2 - exclusive or;
<logical_expression$_1$><logical_operator><logical_expression$_2$>

where the *logical_operator* can be:
And, Or as binary operators (connectives); Not as unary operator.

The precedence of evaluation of logical operators is Not, And, Or, Xor, Eqv, Imp (table 6.2).

The logical functions works as explained in chapter two in the book "**Informatics: Computer Hardware and Programming in Visual Basic"** [AvDg03]. Each one has an associated truth table that take carry of the two states True or False and, in some programming environments, a state called Empty (or Null) to distinguish between False an non value.

3. An *arithmetic expression* uses the syntax:
<operand$_1$><arithmetic_operator><operand$_2$>
where:
- *<operand$_1$>*,*<operand$_2$>* can be numeric variables, constants or arithmetic expressions (or calls to functions that returns numeric values)
- *arithmetic_operator* (binary operators) is one of those shown in table 6.2, column Arithmetic.

Table 6.2. VBScript Operators

Arithmetic		**Comparison**		**Logical**	
Description	**Symbol**	**Description**	**Symbol**	**Description**	**Symbol**
Exponentiation	**^**	Equality	**=**	Negation	**Not**
Unary negation	**-**	Inequality	**<>**	Conjunctio n	**And**
Multiplication	*****	Less than	**<**	Disjunction	**Or**
Division	**/**	Greater than	**>**	Exclusion	**Xor**
Integer division	\	Less than or equal to	**<=**	Equivalenc e	**Eqv**
Modulo arithmetic	**Mod**	Greater than or equal to	**>=**	Implication	**Imp**
Addition	**+**	Object	**Is**		

		equivalence			
Subtraction	**-**				
String concatenation	**&**				

6.5 Procedures and functions

The procedures are small logical components in which you can break (split) a program for a specific task. They are very useful for condensing repeated or shared tasks (such as calculations frequently used). The procedures are called to do their job from other procedures. Generally a procedure can take arguments, perform a series of statements, and change the value of its arguments.

The major benefits of programming with procedures are:
- procedures allow you to break your programs into discrete logical units, each of each you can debug more easily than an entire program without procedures;
- procedures used in one program can act as building blocks for other programs, usually with little or no modification.

The general form of a VBScript procedure/function can be described as follows:

Procedure_type Procedure_name ([Argument_list]) } **the procedure heading**

[declaration_statements] } **procedure body**
[executive_statements]
End Procedure_type

The *Procedure_type* defines if function or procedure:

Procedure_type::=**Sub** | **Function**

The user identifier *Procedure_name* is declared to be the name of a procedure or function.
The *argument_list* declares the values that are passed in from a calling procedure.

A procedure can have two parts:

– a **declaration** part that contains reservations of memory cells that are needed to hold data and program results and what kind of information will be stored in each memory cell.
– an **executive** part that contains statement (derived from the algorithm you want to communicate to the computer) that are translated into machine language and later on executed.

Any identifier declared in the declaration part is by default usable only during the execution of the procedure and can be referenced only within the procedure.

The procedure *executive_statements* describes the data manipulation performed when the procedure is activated through a procedure call statement. The procedure call statement initiates the execution of a procedure.

After *procedure_name* has finished executing, the program statement that follows the procedure call will be executed. The information passed between a procedure and the program that calls it are called procedure parameters.

The values passed into a procedure by the calling program are called procedure inputs and the results returned to the calling program are called procedure outputs.

There are two types of procedures used in VBScript:
1) **Sub** procedures do not return a value;
2) **Function** procedures return a value; you return a value by assigning it to the function name itself: *Function_name=expression* for the return.

A call to a Sub procedure is a stand-alone statement. A Sub procedure can be invoked by a Call statement:

Call *Procedure_name* (*argument_list*) – if specified the argument list must be enclosed in parenthesis;
or
Procedure_name argument_list

A function procedure can return a value to the caller. A call to a function procedure can be realized using the syntax:
Variable_name=Function_name(*arguments*) or **Call** *Function_name(arguments)* or *Function_name arguments*

There are two differences between Sub and Function procedures:
- generally, you call a function by including the function procedure name and arguments on the right side of a larger statement or expression (*Variablename=Functionname*());
- the result value is returned to the caller by intermediate of an assignment statement to the name of the function.

When we call functions or procedures without arguments we must include an empty set of parenthesis () after the procedure/function name if the function is in the right part of an assignment or in an expression; if the call is a stand-alone sentence the parenthesis not required (and not allowed!).

6.6 Decisional (conditional/alternative) statements

The decision structure is used for choosing an alternative (an operation or block of operations) from two possible alternatives. Algorithm steps that select from a choice of actions are called decision steps.

If … Then … Else. The first syntax for alternative structure can be expressed as:

```
If condition Then
    operation_1
  Else
    operation_2
End If
```

The decision block can be expressed in a natural language as:

- evaluate the expression that defines the logical condition <*condition*>;
- If the result of evaluation is True
  Then execute *operation*$_1$ (a block of sentences)
  Else execute *operation*$_2$ (a block of sentences);
- continue the execution with the next step in the flow.

If the condition is True Then the group between Then and Else will be executed Else the group of sentences between Else and End If will be executed.

The logical condition <*condition*> is a logical expression that will be evaluated either to True or either to False. The logical conditions can be simple or complex logical conditions.

A simple logical condition has the general syntax:
<*variable*> [<*relation_operator* ><*variable*>]
or
<*variable*> [<*relation_operator* ><*constant*>]
The *relation_operator* can be one of:

Relation Operator	Interpretation
<	**Less than. Example:** delta < 0
<=	**Less than or equal. Example:** delta <= 0
>	**Greater than. Example:** delta > 0
>=	**Greater than or equal. Example:** delta >= 0
=	**Equal to. Example:** a = 0
<>	**Not equal. Example:** a<>0

If <*variable*> is number or Boolean then is possible to directly compare with 0, respectively True and is not necessary to write the equal relation operator.

The simple logical conditions will be connected by the **AND**, **OR, and NOT** logical operators to form complex conditions. The logical operators are evaluated in the order NOT, AND, and OR. The change of the natural order of evaluation can be done by using parenthesis in the same way for arithmetic expressions. The precedence of operator evaluation in Boolean expressions (logical expressions) is:

Not
^,*, /, div, mod, and
+, -, or
<, <=, =, <>, >=, >

If … Then. It is possible do not have a specific operation on the two branches, that means situations as expressed in one of the syntaxes:

a) Conditionally executing only one statement:

```
    If condition Then statement
            If the condition is True then the statement is executed
```

b) Conditionally executing a set of sentences (a block of):

```
    If condition Then
            Sequence of statements1
    End If
```

If the condition is true then the set of sentences placed between If and End If are executed.

VBScript allows nesting *if ... then ... else* sentences (one if statement inside another) to form complex decision structures (decisions with multiple alternatives).

If … Then … ElseIf. The nested If allows deciding between several alternatives and can be coded as a multiple-alternative decision. The syntax for that nested If is:

```
    If condition1
            Then
                    sequence1
            ElseIf condition2 Then

                    sequence2
            ⋮
            Else
            ⋮
    End If
```

Can be added as many ElseIf clauses as needed to provide alternative choices.

For the first time *condition*$_1$ is tested. If the result is False *condition*$_2$ and so on until a True evaluated condition reached for each the associated sentence block executed. After executing the reached block the control of processing is passed to the next sentence after End if. If no condition evaluates to True then the sentence block associated to the Else branch executes (if Else defined; if not nothing executes).

Case of. Select case (or case of) executes one of several groups of statements depending on the value of an expression (called selector). The case structure (and statement) is especially used when selection is based on the value of a single variable or a simple expression (called the case selector).

```
Select Case test_ expression
   [Case expression_list1
      [sentences1]]
   [Case expression_list 2
```

[*sentences* $_2$]]
⋮
[**Case Else**
[*sentences* $_n$]]
End Select
- each *expression_list*$_i$ is represented (or formed) by one or many comma separated values (value list);
- in the block **Select Case** the case **Else** can appear only once and only as a last case;
- if many cases fit to *test_ expression* then the first founded will be executed;
- each sentence block (*sentences*$_i$) can include zero, one or many sentences;
- the evaluation of the test expression is realized only once at the beginning of the Select Case structure.

6.7 Repeating Structure

The repeating structure repeats a block of statements while a condition is True or Until a condition becomes True. The repetition of steps in a program is called a **loop**. The executions of such blocks follow the scenario (while): the condition is evaluated and if the condition evaluates to:

• **True** then executes the block of statements;

• **False** then end the execution of the cycle (Loop) and continue the execution of the program.

If for the first time the condition is False the sentence block is simply skipped.

Conditional Loop with Condition Evaluated First

Syntax:

$$\begin{array}{ll}
\textbf{Do } [\{\textbf{While}|\textbf{Until}\}\textit{condition}] & \rightarrow \text{beginning of cycle} \\
\quad \left.\begin{array}{l} [\textit{statements}] \\ [\textbf{Exit Do}] \\ [\textit{statements}] \end{array}\right\} & \rightarrow \text{body of the cycle} \\
\textbf{Loop} & \rightarrow \text{the end of sentence block}
\end{array}$$

The commands Loop and Exit Do are used to do:
- **Loop** – an unconditional jump (or branch) to the beginning of the associated cycle (the evaluation of the condition);
- **Exit Do** – an unconditional ending of the cycle (a jump to the next sentence defined under the loop that ends the body of the cycle).

Do…Until work as:

1) Execute statements;
2) Evaluate the condition Loop or Exit

The block of commands between Do… and Loop will be executed while/until the conditional expression “condition” evaluates to True.

Example:

In this example we suppose that the user types numbers containing digits between 0 (zero) and 9 (nine) as integer values. Because the function InputBox() reads text values the mistakes (any other characters than digits and/or spaces between digits) will produces a computation error message. To avoid that is necessary to test whether the typed value is number or not.

```
<html>
<head>
<title>A function definition</title>
<script language=vbscript>
<!--
' Read_Number reads a value from keyboard and verifies
' if number or not (a process called validation). If the value is not a number
' signals that to the user who can choose between cancel or resume the
operation from typing
Function Read_Number(xNr, denNr)
  Dim Answer
  Do While True = True ' an infinite cycle
    xNr = InputBox("Type the value for " & denNr & ":", "Example")
    If (IsNumeric(Trim(xNr)))= False Then
       Answer = MsgBox("The Value for " & denNr & " must be Numerical !")
       If Answer = 2 Then ' Cancel Button pressed
           Read_Number = "*Cancel" ' The returned value to the caller is
*Cancel
           Exit Do ' Exit from the infinite cycle and return to the caller
       End If
     Else
       Read_Number = Trim(xNr) ' The returned value will be the number
                              ' without extra spaces (to the left or right)
       Exit Do ' Exit from the infinite cycle and return to the caller
    End If
 Loop ' Restart the cycle
End Function
-->
</script>
</head>
<body>
<script language=vbscript>
<!--
dim numarcitit
document.write("Varsta:" & read_number(numarcitit,"Varsta "))
-->
</script>
</body>
</html>
```

Conditional Loop with Condition Evaluated After

In this case the operation is executed first and then the condition is evaluated:

```
Do
        operations
Loop {While | Until} condition
```

It can be described as:
- the *operations* are executed;
- the *condition* is evaluated;
- **if** the result of the evaluation of the *condition* is False *then loop* to execute again the *operations*;
- **if** the evaluation of the *condition* is True *then* continue the execution of the program (and close the loop).

Counted Loop. The statement executes a set of statements (operation1) within a loop a specified number of times. A variable is used as counter to specify how many times the statements inside the loop are executed.

Syntax:

```
For counter = $i_v$ To $f_v$ [Step s]
        operations
        [Exit For]
Next [counter]
```

Where:
- i_v – is the initial value (start value – usually 0 or 1);
- f_v – is the end value (the expected value – usually how many times);
- *step* – is the increasing (or decreasing) step for counter; if a value for Step not specified the default is used (+1);
- the value for *s* can be positive or negative:
  - if *s* is positive then must have the inequality $i_v<f_v$ (otherwise the cycle never executes);
  - if *s* is negative then must have the inequality $i_v>=f_v$ (otherwise the cycle never executes);
- the cycle can be stopped unconditionally by intermediate of the sentence **Exit For**

The execution of For (VB) sentence follows the scenario:
1. The value i_v is assigned to the variable *counter*;
2. The value of variable *counter* is compared with the end value f_v (If the value for step is negative is checked if $counter<f_v$);
3. The operations are executed;
4. The value of variable *counter* is incremented with the value step (1 if step not specified);
5. Repeat the steps from 2 to 5.

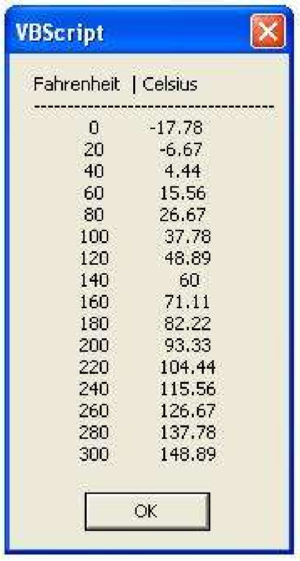
The interpretation of the elements of For…Next sentence is:
what cycle number is   how many times

For *counter=start_value* To *end_value* [Step *increment_decrement*]
    [*statements*]
    [Exit For] ← stop the cycle
    [*statements*]
    Next [*counter*]

Example:

We want to list a Fahrenheit to Celsius correspondence table based on the computation formula:
CELSIUS^{o} = (5/9)*(FAHRENHEIT^{o} - 32)

The correspondence table will be displayed (figure 6.1) starting with the minimal value (**min**) 0 (zero) and ending with the maximal value (**max**) of 300 degrees and the computation and display will be done from 20 to 20 degrees (**pas**). We use, to solve this problem, assignments instructions, the function MsgBox to display the result and an instruction For that allow us to repeat the execution of a group of sentences until a specified condition satisfied. Following paragraphs contains three solutions of that problem.



```
Fahrenheit  | Celsius
------------------------------------
     0         -17.78
    20          -6.67
    40           4.44
    60          15.56
    80          26.67
   100          37.78
   120          48.89
   140            60
   160          71.11
   180          82.22
   200          93.33
   220         104.44
   240         115.56
   260         126.67
   280         137.78
   300         148.89
```

Figure 6.1 The output as a message box

The script looks as:

```
<html>
<head>
<title>Fahrenheit-Celsius</title>
<script language=vbscript>
<!--
 Sub Coresp_Temp()
    Dim min, max, pas, fahrenheit, celsius, tabel
    ' Computation of the correspondence Co- Fo
    min = 0 ' Starting Value
    max = 300 ' Ending Value
    pas = 20 ' From 20 to 20 degrees
    fahrenheit=0
    celsius=0
    tabel=""
    tabel = "Fahrenheit  | Celsius  " & Chr(13) & Chr(10) &_
            string(36, "-") & Chr(13) & Chr(10)
    For fahrenheit = min To max Step pas
```

```
        celsius = (5/9)*(fahrenheit-32)
        tabel = tabel & Right(Space(12) & round(fahrenheit,2),12) & "    " _
                    &   Right(Space(12)   &   round(celsius,2),12)   &   Chr(13)   &
Chr(10)
      Next
      MsgBox(tabel)
  End Sub
-->
</script>
</head>
<body>
<script language=vbscript>
<!--
 Call Coresp_Temp
-->
</script>
</body>
</html>
```

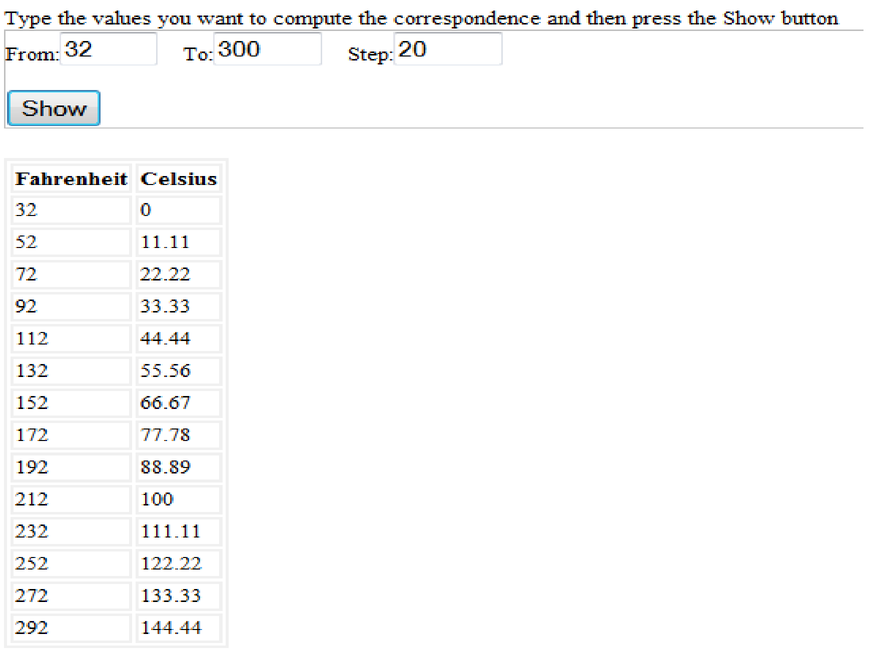
In the next version the script will display the output to a page that will be displayed by the browser.

```
<html>
<head>
<title>Fahrenheit-Celsius</title>
<script language=vbscript>
<!--
  Sub Coresp_Temp()
      Dim min, max, pas, fahrenheit, celsius
      ' Computation of the correspondence Co- Fo
      min = 0 ' Starting Value
      max = 300 ' Ending Value
      pas = 20 ' From 20 to 20 degrees
      fahrenheit=0
      celsius=0
tabs="&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;"
      document.write("Fahrenheit  | Celsius  " & "<br />")
      document.write(string(36, "-") &  "<br />")
     For fahrenheit = min To max Step pas
        celsius = (5/9)*(fahrenheit-32)
        document.write(Right(Space(12) & round(fahrenheit,2),12) & tabs _
                    & Right(Space(12) & round(celsius,2),12) &  "<br />")
     Next
  End Sub
-->
</script>
</head>
<body>
<script language=vbscript>
<!--
```

```
 Call Coresp_Temp
-->
</script>
</body>
</html>
```

The result displayed in a separate page in the shape of a table defined with ASCII characters.

In the next version the script will display the output to a table inside of the page that will replace a defined division using the getElementById() method and innerHTML property.


Type the values you want to compute the correspondence and then press the Show button

From: 32 To: 300 Step: 20

Fahrenheit	Celsius
32	0
52	11.11
72	22.22
92	33.33
112	44.44
132	55.56
152	66.67
172	77.78
192	88.89
212	100
232	111.11
252	122.22
272	133.33
292	144.44

The table generated row by row and cell by cell is stored in the variable named “report” and substituted when ready to <div id="replaceMe"></div> as shown in the following script:

```
<html>
<head>
<title>VBScript solution</title>
<script type="text/vbscript" language="vbscript">
<!--
 function Coresp_Temp()
    Dim min, max, pas, fahrenheit, celsius, report
    ' Computation of the correspondence Co- Fo
    min = minTemp.value
    max = maxTemp.value
    pas = stepTemp.value
    fahrenheit=0
    celsius=0
    report="<table style='width: 50%; border-width: 2px; border-style: solid;'><tr>" &_
    "<td style='border-style: solid; border-width: 2px;'><strong>Fahrenheit</strong></td>" &_
    "<td style='border-style: solid; border-width: 2px;'><strong>Celsius</strong></td></tr>"
    For fahrenheit = min To max Step pas
      celsius = (5/9)*(fahrenheit-32)
      report=report & "<tr><td style='border-style: solid; border-width: 2px;'>"
      report=report & round(fahrenheit,2)
      report=report & "</td><td style='border-style: solid; border-width: 2px;'>"
      report=report & round(celsius,2)
      report=report & "</td></tr>"
    Next
    report=report & "</table>"
    document.getElementById("replaceMe").innerHTML=report
  End function
-->
</script>
</head>
<body>
<h1>VBScript based solution.&nbsp;</h1><br/>
Type the values you want to compute the correspondence and then press the Show button<br/>
<div style="border:1px solid #c0c0c0">
From:<INPUT type="text" id="minTemp" name="minTemp" size="4" align="right" value="0" >
&nbsp;&nbsp;&nbsp;To:<INPUT type="text" id="maxTemp" name="maxTemp" size="5" align="right" value="300">
&nbsp;&nbsp;&nbsp;Step:<INPUT type="text" id="stepTemp" name="stepTemp" size="5" align="right" value="20">
<br/>
<br/><INPUT type="button" value="Show" ID="vbcompdat" onclick="Coresp_Temp()">
</div>
<p></p>
<div id="replaceMe"></div>
</body>
</html>
```

The result displayed in a separate page in the shape of a table defined with HTML tags.

For Each ... Next**.** This sentence allows to apply a set of sentences to an object collection or to a multitude (arrays, vectors, multidimensional massive) without specifying the number of cycles (that specification is difficult if the dynamic memory reservation used).

The syntax of that sentence is:

For Each *element* **In** *group*
    Sentences
Next *element*

Example:

```
<head>
<title>Using For Each sentence</title>
</head>
<html>
<body>
<script language=vbscript>
<!—
dim divisions(2), i, indent
divisions(0)="English"
divisions(1)="French"
divisions(2)="German"
i=1
indent="&nbsp;&nbsp;&nbsp;&nbsp;&nbsp"
document.write("The faculty sections are:<br />")
For Each x in divisions
  document.write( indent & indent & i & ") <b>" & x & "</b><br />")
  i=i+1
Next
-->
</script>
</body>
</html>
```

the code placed in the body of the HTML page will produces the output:

```
The faculty sections are:
        1) English
        2) French
        3) German
```

In the following example Web server VBScript the sentence For Each ... is used to display the logical disks drives extracted from the operating system database (use the command prompt to run WScript.exe to interpret and execute the script)

Example:

```
Option Explicit
On Error Resume Next
```

```
Dim colDrives 'the collection that comes from WMI
Dim drive   'an individual drive in the collection
set colDrives = GetObject("winmgmts:").ExecQuery("select size,freespace " &_
 "from Win32_LogicalDisk where DriveType <> Null")
WScript.Echo "Drive      Size           Freespace"

For Each drive in colDrives 'walks through the collection
   WScript.Echo left(drive.DeviceID & Space(10),10) & " " & left(drive.size &_
Space(15),15) & " " & left(drive.freespace & Space(15),15)
Next
```

6.8 Inserting Objects in HTML pages

The <object> tags allow specifying an object identification and dimensions data. The <param> tags allows set the initial values for object properties.

Example:

```
<OBJECT
  classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
  id=lblActiveLbl
  width=250
  height=250
  align=left
  hspace=20
  vspace=0
>
<PARAM NAME="Angle" VALUE="90">
<PARAM NAME="Alignment" VALUE="4">
<PARAM NAME="BackStyle" VALUE="0">
<PARAM NAME="Caption" VALUE="A Simple Label">
<PARAM NAME="FontName" VALUE="Verdana, Arial, Helvetica">
<PARAM NAME="FontSize" VALUE="20">
<PARAM NAME="FontBold" VALUE="1">
<PARAM NAME="FontColor" VALUE="0">
</OBJECT>
```

This example adds an ActiveX control, called Label, to a page. The object must be installed before usage in the client computer if you want to work this script.

For such inserted objects (or controls, as called in VB) we can get properties, set properties, and invoke methods just as with any of the form controls in VB.

Information about the properties, methods, events, and class identifiers (CLSID) for several ActiveX controls that are available for use with Internet Explorer can be found on the Microsoft® Web site (http://activex.microsoft.com).

6.9 Input Output Operations with InputBox and MsgBox

VBScript offers two functions for input/output operations: input - InputBox(…) and output - MsgBox(…) used to activate a standard dialog.

InputBox. Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns a string containing the contents of the text box.

Syntax:

InputBox(*prompt*[, *title*] [, *default*] [, *xpos*] [, *ypos*] [, *helpfile*, *context*])

where:

Argument	Description
prompt	Required. Is a string expression that is displayed as the message in the dialog box. Can be up to 1024 characters in length. If ***prompt*** consists of more than one line they must be separated by a carriage return character (**Chr(**13**)**), a linefeed character (**Chr(**10**)**), or carriage return–linefeed character combination (**Chr(**13**)** & **Chr(**10**)**).
title	Optional. String expression displayed in the title bar of the dialog box. If you omit ***title***, the application name is placed in the title bar.
default	Optional. String expression displayed in the text box as the default response if no other input is provided. If you omit ***default***, the text box is displayed empty.
xpos	Optional. A numeric expression that specifies, in twips, the horizontal distance of the left edge of the dialog box from the left edge of the screen. If ***xpos*** is omitted, the dialog box is horizontally centered.
ypos	Optional. Numeric expression that specifies, in twips, the vertical distance of the upper edge of the dialog box from the top of the screen. If ***ypos*** is omitted, the dialog box is vertically positioned approximately one-third of the way down the screen.
helpfile	Optional. String expression that identifies the Help file to use to provide context-sensitive Help for the dialog box. If ***helpfile*** is provided, ***context*** must also be provided.
context	Optional. Numeric expression that is the Help context number assigned to the appropriate Help topic by the Help author. If ***context*** is provided, ***helpfile*** must also be provided.

Example:

The call InputBox(“Prompt”, ”Valoare_implicita”, ”Titlu”) will produces the dialog box from figure 6.2.

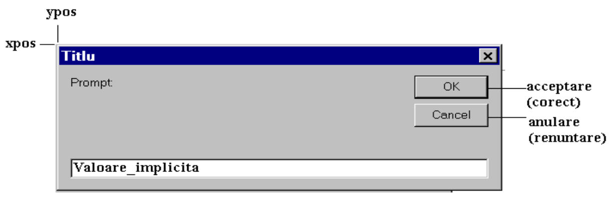


Figure 6.2 Example of using InputBox

MsgBox. Displays a message in a dialog box, waits for the user to click a button, and returns an **Integer** indicating which button the user clicked.

Syntax:

MsgBox(prompt[, buttons] [, title] [, helpfile, context]**)**

where:

Argument	Description
prompt	Required. Is a string expression that is displayed as the message in the dialog box. The maximum length of ***prompt*** is approximately 1024 characters, depending on the width of the characters used. If ***prompt*** consists of more than one line, you can separate the lines using a carriage return character (**Chr(**13**)**), a linefeed character (**Chr(**10**)**), or carriage return – linefeed character combination (**Chr(**13**)** & **Chr(**10**)**) between each line.
buttons	Optional. A numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If omitted, the default value for ***buttons*** is 0.
title	Optional. String expression displayed in the title bar of the dialog box. If you omit ***title***, the application name is placed in the title bar.
helpfile	Optional. String expression that identifies the Help file to use to provide context-sensitive Help for the dialog box. If ***helpfile*** is provided, ***context*** must also be provided.
context	Optional. Numeric expression that is the Help context number assigned to the appropriate Help topic by the Help author. If ***context*** is provided, ***helpfile*** must also be provided.

The value for ***buttons*** argument can be determined as a sum of the following Visual Basic constants:

Constant	Value	Description
vbOKOnly	0	Display **OK** button only.
vbOKCancel	1	Display **OK** and **Cancel** buttons.
vbAbortRetryIgnore	2	Display **Abort**, **Retry**, and **Ignore** buttons.
vbYesNoCancel	3	Display **Yes**, **No**, and **Cancel** buttons.
vbYesNo	4	Display **Yes** and **No** buttons.

vbRetryCancel	5	Display **Retry** and **Cancel** buttons.
vbCritical	16	Display **Critical Message** icon.
vbQuestion	32	Display **Warning Query** icon.
vbExclamation	48	Display **Warning Message** icon.
vbInformation	64	Display **Information Message** icon.
vbDefaultButton1	0	First button is default.
vbDefaultButton2	256	Second button is default.
vbDefaultButton3	512	Third button is default.
vbDefaultButton4	768	Fourth button is default.
vbApplicationModal	0	Application modal; the user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal; all applications are suspended until the user responds to the message box.
vbMsgBoxHelpButton	16384	Adds Help button to the message box
VbMsgBoxSetForeground	65536	Specifies the message box window as the foreground window
vbMsgBoxRight	524288	Text is right aligned
vbMsgBoxRtlReading	1048576	Specifies text should appear as right-to-left reading on Hebrew and Arabic systems

For example the call MsgBox("Prompt",vbInformation+vbOkCancel, "Titlu") will produces the dialog shown in figure 6.3. The returned values correspond to the pressed button:

Constant	Value	Description
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

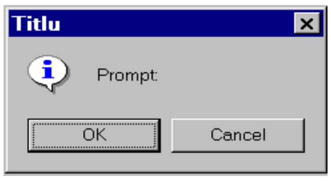


Figure 6.3 Example of an MsgBox dialog

In the following example is illustrated how to write the message on many lines and what means a string expression:

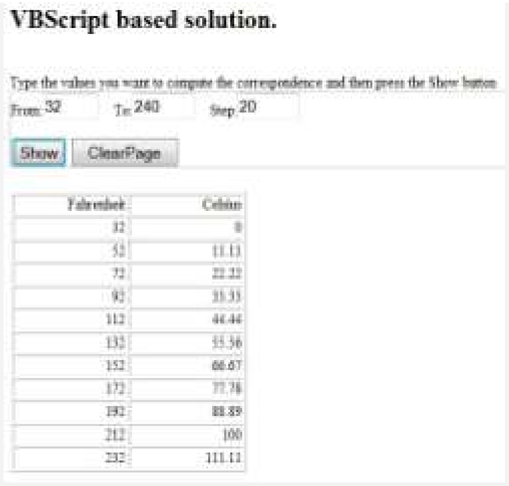
MsgBox "The Student " & Name & **Chr**(13) & **Chr**(10) & " has the average mark: " & media

What is between "…" means literal strings that will be placed as such in the message; *Name* and *media* are named variables whose content will be concatenated by the string concatenation operator (&) with the literals and the function calls **Chr**(13) & **Chr**(10) means display what follows on a separate line.

6.10 Combining VBScript and Forms

In the example below the function validateValues(), is called by the procedure Coresp_Temp() that in turn is activated by the event click on the button Show, and looks to the form fields values and check if they are numbers. If not numbers an error message returned to the caller and displayed in a message box and the processing stops. If all numbers the Coresp_Temp() procedure produces the HTML tags for a table containing the correspondence value between Celsius an Fahrenheit degrees. Finally it displays the table (stored in the memory variable repx) by replacing the division called “replaceMe” in the web page. The button called ClearPage clears the forms fields together with the content generated in the page.

```
<html>
<head>
<title>VBScript solution</title>
<script type="text/vbscript"
language="vbscript">
<!--
  function
validateValues(min,max,pas)
   dim ret
   ret="Yes"
   if isnumeric(trim(minTemp.value))
then
      min=trim(minTemp.value)
     else
      if ret="Yes" then ret=” ”
      ret=ret & "<<From>>"
   end if
   if isnumeric(trim(maxTemp.value))
then
      max=trim(maxTemp.value)
     else
      if ret="Yes" then ret=” ”
      ret=ret & "<<To>>"
   end if
   if isnumeric(trim(stepTemp.value)) then
      pas=trim(stepTemp.value)
     else
      if ret="Yes" then ret=” ”
      ret=ret & "<<Step>>"
   end if
   validateValues=ret
  end function
  Sub Coresp_Temp()
      Dim min,max,pas,fahrenheit, celsius,conr, repx
```



VBScript based solution.

Type the values you want to compute the correspondence and then press the Show button

From: 32 To: 240 Step: 20

Fahrenheit	Celsius
32	0
52	11.11
72	22.22
92	33.33
112	44.44
132	55.56
152	66.67
172	77.78
192	88.89
212	100
232	111.11

The page displayed by the code

```
     ' Computation of the correspondence Co- Fo
     if (validateValues(min,max,pas))<>"Yes" Then
       msgbox "Err. 01. The value you typed in " & validateValues(min,max,pas) & "
box is not a number ? Correct them and press again."
       exit sub
     end if
     fahrenheit=0 :  celsius=0
     repx="<table border=1 style='text-align:right'><tr><td width=120px>Fahrenheit
</td><td width=120px> Celsius  </td></tr>"
     For fahrenheit = min To max Step pas
       celsius = (5/9)*(fahrenheit-32)
       repx=repx & "<tr><td width=120px>" & Right(Space(24) &
round(fahrenheit,2),12) & _
                "</td><td width=120px>" & Right(Space(24) & round(celsius,2),16) &
"</td></tr>"
     Next
     repx=repx & "</table>"
     document.getElementById("replaceMe").innerHTML=repx
  End Sub
  function ClearFields()
    minTemp.value=""
    maxTemp.value=""
    stepTemp.value=""
    document.getElementById("replaceMe").innerHTML=""
  end function
-->
</script>
</head>
<body>
<h1>VBScript based solution.&nbsp;</h1><br/>
Type the values you want to compute the correspondence and then press the
Show button<br/>
<div style="border:1px solid #c0c0c0">
From:<INPUT type="text" id="minTemp" name="minTemp" size="4" align="right"
value="0" >
&nbsp;&nbsp;&nbsp;To:<INPUT type="text" id="maxTemp" name="maxTemp"
size="5" align="right" value="300">
&nbsp;&nbsp;&nbsp;Step:<INPUT type="text" id="stepTemp" name="stepTemp"
size="5" align="right" value="20">
<br/>
<br/><INPUT type="button" value="Show" ID="vbcompdat"
onclick="Coresp_Temp()">
<INPUT type="button" value="ClearPage" ID="vbclr" onclick="ClearFields()">
</div>
<p></p>
 <div id="replaceMe"></div>
</body>
</html>
```

7 JavaScript

7.1 JavaScript – An introduction

JavaScript is a scripting language that gives HTML designers a programming tool and that can be used for easy management of user interface: it can put dynamic text into a HTML page, it can make the page react to events or it can create and easy manipulate cookies. A JavaScript inserted in the HTML document allows a local recognition and processing (that means at client level) of the events generated by the user such as those generated when the user scans the document or for management of fill-in forms, for example, we must recuperate the information referencing the client (name, address, payment etc). By inserting a JavaScript in the HTML page we can validate the data filled by the client (for example we can validate the Credit Card Account, solvability, transactions history, etc) before it is submitted to the server.

JavaScript allows restructuring an entire HTML document for which we can add, remove, change, or reorder items on a page. In order to change anything on a page, JavaScript needs access to all elements in the HTML document. This access, along with methods and properties to add, move, change, or remove HTML elements, is given through the Document Object Model (DOM).

In 1998, W3C published the Level 1 DOM specification. This specification allowed access to and manipulation of every single element in an HTML page. All browsers have implemented this recommendation, and therefore, incompatibility problems in the DOM have almost disappeared. The DOM can be used by JavaScript to read and change HTML, XHTML, and XML documents. The DOM is separated into different parts (Core, XML, and HTML) and different levels (DOM Level 1/2/3):

- Core DOM - defines a standard set of objects for any structured document;
- XML DOM - defines a standard set of objects for XML documents;
- HTML DOM - defines a standard set of objects for HTML documents.

Every object can have his own Collections, Attributes (Properties) and Methods. Table 7.1 shows the JavaScript objects and table 7.2 shows the HTML DOM objects.

Table 7.1 The JavaScript objects

Object	**Description**
Window	The top level object in the JavaScript hierarchy. The Window object represents a browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag

Navigator	Contains information about the client's browser
Screen	Contains information about the client's display screen
History	Contains the visited URLs in the browser window
Location	Contains information about the current URL

Table 7.2 HTML DOM objects

Object	Description
Document	Represents the entire HTML document and can be used to access all elements in a page
Anchor	An <a> element
Area	An <area> element inside an image-map
Base	An <base> element
Body	The <body> element
Button	A <button> element
Event	Represents the state of an event
Form	A <form> element
Frame	A <frame> element
Frameset	A <frameset> element
Iframe	An <iframe> element
Image	An <img> element
Input button	A button in an HTML form
Input checkbox	A checkbox in an HTML form
Input file	A fileupload in an HTML form
Input hidden	A hidden field in an HTML form
Input password	A password field in an HTML form
Input radio	A radio button in an HTML form
Input reset	A reset button in an HTML form
Input submit	A submit button in an HTML form
Input text	A text-input field in an HTML form
Link	A <link> element
Meta	A <meta> element
Option	An <option> element
Select	A selection list in an HTML form
Style	An individual style statement
Table	A <table> element
TableData	A <td> element
TableRow	A <tr> element
Textarea	A <textarea> element

JavaScript is hardware and software platform independent. Within a JavaScript inserted in the HTML page we can validate the data supplied by the client (for example, to validate the card account, financial availability, history regarding previous transactions etc.).

For an inserted JavaScript the <script type="text/javascript"> and </script> tags tells where the JavaScript starts and ends:

```
<html>
<body>
<script type="text/javascript">
<!--
... // put here the script body
//-->
</script>
</body>
</html>
```

The properties innerText and innerHTML allow us to access the contents - the code - contained in an object. By manipulating the innerText and innerHTML properties, we can change, dynamically, the text on a page (without reloading the page). For example, given a paragraph whose id = "sampleparagraph", its innerText and innerHTML may be accessed via:

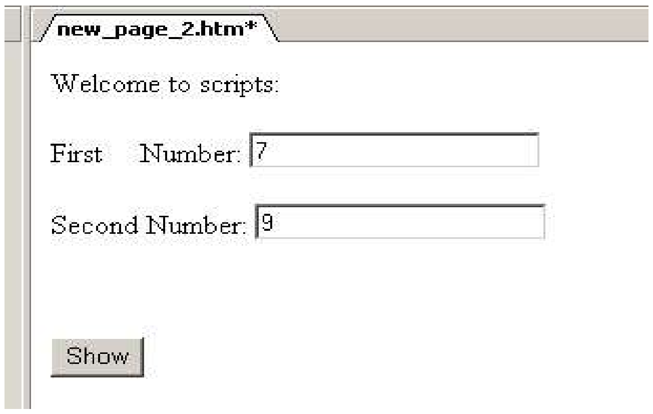
document.getElementById('sampleparagraph').innerHTML – this is interpreted as HTML

document.getElementById('sampleparagraph').innerText – this is interpreted as text

If the content of sampleparagraph is “<b> inner text</>” then:

- innerText would display as <b>inner text</b>
- innerHTML would display as **inner text.**

Figure 3.1 shows a HTML form containing a VBScript and a JavaScript.


new_page_2.htm*

Welcome to scripts:

First Number: 7

Second Number: 9

Figure 3.1 A Java Script Example

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>New Page 2</title>
<script type="text/javascript" language="javascript">
  <!--
     function calc(a, b){ return (a*b);}
 // -->
</script>
<script id=clientEventHandlersVBS language=vbscript>
<!--
Sub Validate_onclick
     document.write("You Type:"+cstr(text1.value)+":"+cstr(text2.value))
End Sub
-->
</script>
</head>
<body>
     <script type="text/javascript" language="javascript">
        var welcmess="Welcome to scripts:";
        document.write(welcmess)
     </script>
     <p></p> First&nbsp;&nbsp;&nbsp;&nbsp; Number: <INPUT type="text"
ID=Text1 value="0" name="text1" size="20">
     <p></p> Second Number: <INPUT type="text" ID=Text2 value="0"
name="text2" size="20">
     <p></p>&nbsp;<p><INPUT type="button" value="Show" id="Validate"></p>
</body>
</html>
```

The Style object represents an individual style statement that you can think as an inline style declaration. The Style object can be accessed from the document or from the elements to which that style is applied. For example, given a form whose id = "form1", its styles may be accessed via:

```
document.getElementById('form1').style.property
```

where *property* is one of the many style properties available to a given element. Table 7.7 shows some common style properties that we can manipulate.

Table 7.7 Style properties

Property	**Description**
style.background	Sets or retrieves the background picture tiled behind the text and graphics in the object.
style.backgroundAttachment	Sets or retrieves how the background image is

	attached to the object within the document.
style.backgroundColor	Sets or retrieves the color behind the content of the object.
style.backgroundImage	Sets or retrieves the background image of the object.
style.border	Sets or retrieves the width of the border to draw around the object.
style.borderBottom	Sets or retrieves the properties of the bottom border of the object
style.borderBottomColor	Sets or retrieves the color of the bottom border of the object.
style.borderBottomStyle	Sets or retrieves the style of the bottom border of the object.
style.borderBottomWidth	Sets or retrieves the width of the bottom border of the object.
style.borderCollapse	Sets or retrieves a value that indicates whether the row and cell borders of a table are joined in a single border or detached as in standard HTML.
style.borderColor	Sets or retrieves the border color of the object.
style.borderLeft	Sets or retrieves the properties of the left border of the object
style.borderLeftColor	Sets or retrieves the color of the left border of the object.
style.borderLeftStyle	Sets or retrieves the style of the left border of the object
style.borderLeftWidth	Sets or retrieves the width of the left border of the object.
style.borderRight	Sets or retrieves the properties of the right border of the object.
style.borderRightColor	Sets or retrieves the color of the right border of the object.
style.borderRightStyle	Sets or retrieves the style of the right border of the object.
style.borderRightWidth	Sets or retrieves the width of the right border of the object.
style.borderStyle	Sets or retrieves the style of the left, right, top, and bottom borders of the object
style.borderTop	Sets or retrieves the properties of the top border of the object.
style.borderTopColor	Sets or retrieves the color of the top border of the object.
style.borderTopStyle	Sets or retrieves the style of the top border of the object.
style. borderTopWidth	Sets or retrieves the width of the top border of the object.
style.borderWidth	Sets or retrieves the width of the left, right, top, and bottom borders of the object.

style.bottomMargin	Sets or retrieves the bottom margin of the entire body of the page.
style.color	Sets or retrieves the color of the text of the object
style.font	Sets or retrieves a combination of separate font properties of the object. Alternatively, sets or retrieves one or more of six user-preference fonts.
style.fontFamily	Sets or retrieves the name of the font used for text in the object.
style.fontSize	Sets or retrieves a value that indicates the font size used for text in the object.
style.fontStyle	Sets or retrieves the font style of the object as italic, normal, or oblique.
style.fontVariant	Sets or retrieves whether the text of the object is in small capital letters.
style.fontWeight	Sets or retrieves the weight of the font of the object
style.margin	Sets or retrieves the width of the top, right, bottom, and left margins of the object.
style.marginBottom	Sets or retrieves the height of the bottom margin of the object.
style.marginHeight	Sets or retrieves the top and bottom margin heights before displaying the text in a frame.
style.marginLeft	Sets or retrieves the width of the left margin of the object.
style.marginRight	Sets or retrieves the width of the right margin of the object.
style.marginTop	Sets or retrieves the height of the top margin of the object.
style.marginWidth	Sets or retrieves the left and right margin widths before displaying the text in a frame.
style.padding	Sets or retrieves the amount of space to insert between the object and its margin or, if there is a border, between the object and its border
style.paddingBottom	Sets or retrieves the amount of space to insert between the bottom border of the object and the content.
style.paddingLeft	Sets or retrieves the amount of space to insert between the left border of the object and the content.
style.paddingRight	Sets or retrieves the amount of space to insert between the right border of the object and the content.
style.paddingTop	Sets or retrieves the amount of space to insert between the top border of the object and the content.
style.position	Sets or retrieves the type of positioning used for

	the object.
style.textAlign	Sets or retrieves whether the text in the object is left-aligned, right-aligned, centered, or justified.
style.textDecoration	Sets or retrieves a value that indicates whether the text in the object has blink, line-through, overline, or underline decorations.
style.textIndent	Sets or retrieves the indentation of the first line of text in the object.
style.topMargin	Sets or retrieves the margin for the top of the page.
style.vAlign	Sets or retrieves how text and other content are vertically aligned within the object that contains them.
style.visibility	Sets or retrieves whether the content of the object is displayed.
style.zIndex	Sets or retrieves the stacking order of positioned objects.

The JavaScript sentences involving text strings can be brake up within the text string by using the \ (backslash) character.

The multiline comments can be defined between /* and */; the one line or the inline comments can be defined by using // (two slashes). The extraspace is ignored and the sentences are case sensitive. The ; (semicolon) ending sentence character is optional for sentences defined alone on a line and compulsory for separating the commnds defined in the same line (generally the inline scripts).

7.2 Using and placing JavaScripts in a HTML page

In the following paragraphs are introduced some examples of using (and placing) JavaScripts in a HTML page.

7.2.1 JavaScript in the body of the HTML file

Java script in the body of the HTML page will be executed when the page loads. Generally, the scripts in the body, will generate the content of the page.

Example:

```
<html>
 <head>
   <title>
        Page containing JavaScript
   </title>
  </head>
 <body>
   <script type="text/javascript">
        document.write("This text is displayed by a JavaScript!")
   </script>
 </body>
</html>
```

Comments:

The tag <script> have the attribute „type”, that specifies the script type (JavaScript in our case). This script composed by a single command that displays inside the page the text: "This text is displayed by a JavaScript!". If you want include many commands on the same line this must be separated by the **”;”** (semi colon) character.

The concatenation of text string is realized by using the + character, for example the expression "This text is " + "concatenated." will produce the string "This text is concatenated."

The „/” have a special meaning for the HTML language and consequentely when we want display the slash character itself we must precede (prefix) this by a „\” (backslash), as illustrated in this example:

```
document.write("<i>"+"The Operator + is Concatention!"+"<\/i>")
```

If the script is included in a comment tag (<!--) then the browsers that do not „know” (interpret) JavaScript will not display in the page the script text (script source), as shown in the folowing sample:

```
<!--
document.write("<i>"+"This text is displayed by a
JavaScript!"+"<\/i>")
//-->
```

The browsers that know JavaScript will process the script, even this included in a comment line.

The string „//”, comment in JavaScript, tell to the browser do not process the line „-->”. We can not use the sintax „//<!--”, because a browser that do not interpret JavaScript will display that string „//”.

7.2.2 JavaScript in heading

If we want be shure that the script executes before displaying any element in the page we can include this in the heading part of the HTML page (file). The JavaScript in the head section will be executed when called, or when an event is triggered.

Example:

```
<html>
 <head>
   <title>
        Page with JavaScript
   </title>
  <script type="text/javascript">
      document.write("This text is displayed by a JavaScript!")
  </script>
 </head>
 <body>
   <P>  This text must appear in the page after the execution of the
Javascript.
 </body>
</html>
```

The number of scripts placed in the head section and in the body of a HTML page is unlimited.

7.2.3 External JavaScripts

A JavaScript can be stored into an external script file from where we can use in many Web pages. In that way the script is written only once and in every HTML file we want use is enough to invoke the file containing the script. The stored script cannot contain the tag <script> or his pair </script>.

The steps followed when using externaly stored scripts are:

1. The creation of the external file containing the script lines, for example the line:

```
document.write("Text from an external stored script.")
```

2. The file is saved with the wanted name and the extension *js* (java script), for example we name the file *scriptex.js*

3. In the HTML pages we want include the stored script file is added the following script:

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

The „src” attribute of the tag <script> allows specifying the file containing the script we want execute.

7.3 Defining and using variables

JavaScript can contain variable definitions and references to that variables. The variables can be used to store values and the references to that values can be done by referencing the name of the variable. The lifetime of variables can be:

- for variables declared within a function - can only be accessed within the function; they created when encountered their declaration as the function progreses and destroyed when exiting; they called local variables and you can use the same name in diffrent functions;

- for variables declared outside a function – can be accessed anywhere in the page; the name must be unique at that level; the lifetime of these variables starts when they are declared, and ends when the page is closed.

The variable declaration can include an assignment and can be done using one of sentences:

var variableName=somevalue
or
variableName=somevalue
In the following example, on define a variable called „mess" that is initialized with the value „This text contained by the variable called mess", and later on referenced in a write sentence:

```
<html>
  <head>
    <title>
       Page with JavaScript
    </title>
  </head>
  <body>
    <script type="text/javascript">
      <!--
      var mess= " This text contained by the variable called mess"
      document.write("<i>"+mess+"<\/i>")
      //-->
   </script>
 </body>
</html>
```

The variables, functions and, objects names are case-sensitive and must begin with a letter or an _ (underscore) character.

7.4 Methods

JavaScript is an object based programming language and uses objects (as shown in tables 7.1 and 7.2). It has many built-in objects such as Area, Image, Date, Window, and Document, and allow also user defining his own objects. In the following table some methods for document and window explained:

Method	Explanation-Example
document.write("msg")	Displays the message „msg" in the page containing the script Example: 1) displaying text in a page: document.write("This text will be displayed in the page") 2) displaying attributes of a page, such as title and URL: <script type="text/javascript"> document.write(document.title+":"+document.URL) </script> </body> *) The formating of the message to be displayed by

	write and alert methods or other intrinsic functions that manipulate strings is realized by intermediate of escape sequences.

window.alert("msg")	Displays a dialog box (alert box) containing the message „msg” and the OK button.
	Example:
	function display_alert()
	{
	 alert("The message formatting is ensured" + '\n' + "by using a lot of so called \‘escape sequences\’")
	}

window.prompt("msg"," default")	Displays a dialog box prompting the user for input and confirm/cancel the dialog.
	Example:
	function display_prompt()
	{
	  var name=prompt("Type your name here","")
	  if (name!=null && name!="")
	  {
	    document.write("You typed " + name + "! It is that correct ?")
	  }
	}

window.confirm("msg")	Displays a dialog box with a message, a Cancel, and an OK button (similar to MsgBox,).
	Example:
	function display_confirm()
	{
	 var buttonpressed=confirm("Press a button")
	 if (buttonpressed ==true)
	 {
	  document.write("You pressed the OK button!")
	 }
	 else
	 {
	  document.write("You pressed the Cancel button!")
	 }
	}

window.open("URL", "name_of_new_window ", "specifications")	Opens a new browser window for the page indicated by the URL argument. The window can be referenced by the name “name_of_new_window” and can be customized by the values supplied by the “specifications” argument.

	Example:
	<html>
	<head>
	<script type="text/javascript">
	function open_win_ase()
	{
	window.open("http://www.ase.ro","_blank","toolbar =yes, location=yes, directories=no, status=no, menubar=yes, scrollbars=yes, resizable=no, copyhistory=yes, width=400, height=300")
	}
	function open_win_avrams()
	{
	window.open("http://www.aavrams.ro","_blank","to olbar=yes, location=yes, directories=no, status=no, menubar=yes, scrollbars=yes, resizable=no, copyhistory=yes, width=400, height=300")
	}
	</script>
	</head>
	<body>
	<form>
	<input type="button" value="Faculty" onclick="open_win_ase()">
	<input type="button" value="Course Notes" onclick="open_win_avrams()">
	</form>
	</body>
	</html>

*) Common Escape Sequences for text display formatting are represented by:

Ampersand	\&	Carriage return	\r	Backslash	\\
Double quote	\"	Newline	\n	Backspace	\b
Single quote	\'	Form feed	\f	Tab	\t

7.5 Document Object Model (DOM)

The Document Object Model defines HTML documents as a collection of objects and provides access to every element, identified uniquely by intermediate of an id attribute, in a document. Any element may be accessed (by using the method getElementById) and modified by a snippet of JavaScript. The Window object is the top level object in the JavaScript hierarchy (it represents a browser window). A Window object is created automatically with every instance of a <body> or <frameset> tag. You can see and exercises the various elements of DOM HTML by following the link:

http://www.w3schools.com/htmldom/dom_examples.asp

Examples:

a) This sample displays the message „This is first paragraph! Click, and see”. If you click somewhere in the displayed text it displays:
„The background is: *the current color name*
Will be changed in Yellow!”:

```
<html>
<head>
<title>Using DOM</title>
<script language="javascript">
<!--
  function xalert()
  {
  var x=document.getElementById("par1");
  x.style.background="red";
 alert("The background is:" + x.style.background+"\n Will be changed in
Yellow!")
  if(x.style.background=="red")
    {
     x.style.background="yellow";
    }
    else
    {
      x.style.background="red";
    }
  }
-->
</script>
</head>
<body>
   <p id="par1" onclick="xalert()" >This is first paragraph! Click, and
see</p>
</body>
</html>
```

b) This sample uses innerHTML to change dynamically the header identified by „chgheader”:

```
<html>
<head>
<script type="text/javascript">
function getValue()
{
var x=document.getElementById("myHeader")
alert(x.innerHTML)
}
function chgval()
{
 document.getElementById("chgheader").innerHTML="My Header
(Changed)"
}
```

```
</script>
</head>
<body>
<h1 id="myHeader" onclick="getValue()">This is first header</h1>
<p>Click on the header to alert its value</p>
<h2 id="chgheader" onclick="chgval()">This is the second header</h2>
<p>Click on the header to change its value</p>
</body>
</html>
```

7.6 Using and Defining Function

A function contains code (a set of statements) which is executed when triggered by an event or a call to that function. In JavaScript is possible to use the Java language intrinsic functions or user defined functions (must be defined before any usage).

In the case of user defined functions is preferable that the definition is made in the head section of the HTML page to be shure (or to ensure) in that way they loaded before calling. This required because the browser start processing the HTML page before completly downloading this from server. You may call a function from anywhere within a page (or even from other pages if the function is embeded in an external script).

The general syntax of a function is:

```
function function_name([argument1,argument2,etc])
{
  some_statements
  [return expression]
}
```

where:

function_name is the name the function you want have;

argument1,argument2,… the name for the function parameters if it has. Is allowed do not pass any parameter to the function;

some_statements generally variable declarations and executables statements that describes the steps of the algorithm you model. They define together with the return statement (if present) the body of the function;

expression is the expression whose evaluation will represent the returned value. A function can return (the sentence return must be present in the body) a value or not (the return statement is not appearing between those of the function’s body);

A function can be invoked in one of the ways :

- without arguments:

  function_name()

- or with arguments:

  function_name(*argument1,argument2,etc*)

A function is executed when is called. The function can be called within a JavaScript block, via an event handler (inside an HTML tag) or via a href link.

In the following example is called the function „alert” (a standard function of the JavaScript language) with the argument „mess” (the variabe used in the previous example) and that determine the display of an alert box in which the content of variable „mess” displayed.

Example:

The call of an JavaScript intrinsic function.

```
<html>
 <head>
  <title>
     Page with JavaScript
  </title>
 </head>
 <body>
  <script type="text/javascript">
    <!--
    var mess= " This text contained by the variable called mess"
    document.write("<i>"+mess+"<\/i>") // a method of the document object
    alert(mess)  // the intrinsic function invoked
    //-->
   </script>
 </body>
</html>
```

Example:

The call of an user defined function.

```
<html>
 <head>
  <title>
     Page with JavaScript
  </title>
  <script type="text/javascript">
   <!--
    function suma(a,b)
    {
       rezult=a+b
       return rezult
    }
 //-->
 </script>
 </head>
 <body>
  <script type="text/javascript">
    <!--
    var mess= " This text contained by the variable called mess"
    document.write("<i>"+mess+"<\/i>")
    alert(mess)
    alert(suma(“This text is a ”,”<String Concatenation>”))
    document.write("This digit is the result of adding 2 to 7 by using the
user defined function suma:"+suma(2,7))
```

```
    //-->
  </script>
 </body>
</html>
```

Example :

In this example the JavaScript defined in the head part of the page and in the body part in a href tag.

```
<html>
<head>
<title>A Page with scripts</title>
<script type="text/javascript">
   function chgbgcolor()
   {
    document.bgColor="green"
   }
   function chcolor()
   {
    document.bgColor="orange"
   }
</script>
<script id=clientEventHandlersJS language=javascript>
<!—
function window_onload() {
   document.bgcolor="orange"
}
//-->
</script>
<script language=javascript for=window event=onload>
<!—
return window_onload()
//-->
</script>
</head>
<body>
<p>The page not empty  </>
<a href="javascript:chcolor(); alert('The background was changed!')">This
is a script in href</a></p>
<h1 id="header1" onclick="chgbgcolor()"> IT4B: </h1>
<h2 id="header2">Errata</h2>
</body>
</html>
```

7.7 Asignments and expressions

The general syntax for assignment is:

variable = expression

The interpretation of this is: the *variable* before the assignment operator is assigned a value of the *expression* after it, and in the process, the previous value of *variable* is destroyed. An *expression* can be an arithmetic expression, a logical expression or expression on character string. It can be a variable, a constant, a literal, or a combination of these connected by appropriate operators.

An assignment statement stores a literal value or a computational result in a variable and is used to perform most arithmetic operation in a program.

7.7.1 Arithmetic Expression

An *arithmetic expression* uses the syntax:

<operand$_1$><arithmetic_operator><operand$_2$>

where:

- *<operand$_1$>*,*<operand$_2$>* can be numeric variables, constants or arithmetic expressions (or calls to functions that returns numeric values)
- *arithmetic_operator* (binary operators) is one of those shown in table 7.3.

Table 7.3 Arithmetic Operators

Operator	**Description**	**Example** Suppose x=2
+	Addition	x+2 = 4
-	Subtraction	5-x = 3
*****	Multiplication	x*5 = 10
/	Division	9/3 = 3; 5/2 = 2.5
%	Modulus (division remainder)	5%2 = 1; x%2 = 0; 10%5 = 0
++	Increment By One *)	x++ = 3
--	Decrement By One *)	x-- = 1

*) The increment and decrement operators can either be used as a pre- or a post- operator:

Post-Increment: the line of code is executed as then the increment/decrement is performed.
y = x++ is equivalent to the sequence:
y = x
x = x + 1
	Pre-Increment: the increment or decrement is performed before whatever other operations are present within the given line of code.
	y = ++x is equivalent to the sequence:
	x = x + 1
	y = x

For arithmetic expressions the assignment operator can be combined with the arithmetic ones to define compact expressions as shown in the table 7.4.

Table 7.4 Assignment Operators

Operator	**Example**	**Is The Same As**
=	x = y	x = y

+=	x += y
	x+=y-12
		x = x + y
		x=x+(y-12)

-=	x-=y
	x -= y +12
		x=x-y
		x = x - (y + 12)

***=**	x*=y
	x *= 3 + y
		x=x*y
		x = x * (3 + y)

/=	x /= y
	x/=y+2
		x = x / y
		x=x/(y+2)

%=	x %= y
	x%=y+2
		x = x % y
		x=x%(y+2)

7.7.2 Logical Expression

A *logical expression* can be:

• **simple**, with the general syntax:
<variable>[*<relation_operator><variable>*]
or
<variable>[*<relation_operator><constant>*]
<relation_operator>::=<|<=|>|>=|==|!=
Table 7.5 shows and explains the comparison operators.

• **complex**, with the general syntax:
e_1 && e_2 - logical and;
e_1 || e_2 - logical or;
! e_1 - logical not.
<logical_expression$_1$><logical_operator><logical_expression$_2$>
where the *logical_operator* can be:
&& (and; two ampersand character), || (or; two vertical bar character) as binary operators (connectives);
! (not) as unary operator.

The precedence of evaluation of logical operators is Not, And, Or. The logical operator together with the truth table defining the way they operate and usage examples are shown in table 7.6.

Table 7.5 Comparison Operators

Operator	**Description**	**Example Suppose x=2**
==	is equal to	x == 3 returns false
===	Is equal to (checks for both value and data type)	y=”2” x==y returns true x===y return false (x integer; y string)
!=	is not equal	x != 3 returns true
>	is greater than	x > 3 returns false
<	is less than	x < 3 returns true
>=	is greater than or equal to	x >= 3 returns false
<=	is less than or equal to	x <= 3 returns true

Table 7.6 Logical Operators

Operator	Description			Example Suppose x=2; y=3
&&	**x**	**y**	**and**	
	T	T	T	(x < 9 && y > 1) returns true
	T	F	F	(x < 9 && y < 1) returns false
	F	T	F	(x > 9 && y > 1) returns false
	F	F	F	(x > 9 && y < 1) returns false
||	**x**	**y**	**or**	
	T	T	T	(x < 9 || y > 1) returns true
	T	F	F	(x < 9 || y < 1) returns true
	F	T	F	(x > 9 || y > 1) returns true
	F	F	F	(x > 9 || y < 1) returns false
!	**x**		**not**	
	F		T	!(x == y) returns true
	T		F	!(x > y) returns false

7.7.3 String Expression

An *expression on character string* can be built (in Java but not only) using:
- the concatenation operator: +
- intrinsic functions for extracting substrings from a string variable or string constant such as:
Right(*string*,*number_of_characters*) - extracting substring from the end
Left(*string*,*number_of_characters*) - extracting substring from the beginning
- functions that manipulate strings:
Cstr(*expression*) – convert the expression in a character string;
Lcase(*string_expression*) – convert the string in lower case;
Ltrim(*string*), Rtrim(*string*), Trim(*string*) – eliminates the spaces (trailing) from left (leading blanks), right and, respectively left-right;
Str(*number*) – converts number in string;
Ucase(*string*) – converts string to uppercase.

The code sequence below is string concatenation by using the concatenation operator + example.

```
text1 = "Faculty of"
text2         =         "Business
Administration"
text3 = text1 +” “+ text2
```

The variable text3 now contains the string "Faculty of Business Administration". The concatenation with the space character “ “ is realized to separate the strings (generally stored with no trailing blanks).

7.8 Conditional Execution

A script can include branches (If...Then...Else...) alowing the definition of conditional executions similarly to the example in the following sequence:

```
if (navigator.appName.indexOf("Internet Explorer")!=-1)
{
   alert("The used Navigator is Internet Explorer!")
}
else
{
   alert(("The used Navigator is not Internet Explorer!")
}
```

These code sequence will display an alert box containing a diffrent text depending on the type of the used browser in which the page displayed. The conditional expression of the IF sentence searches for the text „Internet Explorer" in the name of the browser (navigator) application. If this text do not appears in the browser name then the function „indexOf" returns the value „-1". The condition evaluates to True only if the return of that function is not „-1". The JavaScript IF sentence, „switch" sentence or the conditional operator „?" can be used to define the conditional execution.

7.9 Decision sentences

The decision sentences are used to model the decision structure and that is used for choosing an alternative (an operation or block of operations) from two possible alternatives. Algorithm steps that select from a choice of actions are called decision steps.

If … Then … Else …

```
    if (condition)
            operation_1;
        else
            operation_2;
```

If one of operations includes a sentences sequence then this sequence will be included in a sentence block:

```
{
    operation_i;
}
```

The decision block can be expressed in a natural language as:
- evaluate the expression that defines the logical condition <*condition*>;

- If the result of evaluation is True

    Then execute *operation*$_1$

    Else execute *operation*$_2$;

- continue the execution with the next step in the flow

If … Then …

```
if (condition) operation;

if (condition) {
      operations;
}
```

if...else if....else statement

This statement allows select one of many blocks of code to be executed.

```
if (condition1)
{
        code to be executed if condition1 is true
}
else if (condition2)
{
        code to be executed if condition2 is true
}
else
{
        code to be executed if condition1 and condition2 are not true
}
```

The logical condition <*condition*$_x$> is a logical expression that will be evaluated either to True or either to False. The logical conditions can be simple or complex logical conditions.

A simple logical condition has the general syntax:

<*variable*> [<*relation_operator* ><*variable*>]

or

<*variable*> [<*relation_operator* ><*constant*>]

The *relation_operator* can be one of:

Relation Operator	Interpretation
<	**Less than. Example:** delta < 0
<=	**Less than or equal. Example:** delta <= 0
>	**Greater than. Example:** delta > 0
>=	**Greater than or equal. Example:** delta >= 0
= =	**Equal to. Example:** a == 0
!=	**Not equal. Example:** a!=0

The simple logical conditions will be connected by the **AND**, **OR, and NOT** logical operators to form complex conditions. The logical operators are

evaluated in the order NOT, AND, and OR. The change of the natural order of evaluation can be done by using parenthesis in the same way for arithmetic expressions.
Example:

```
<html> <head> <title>New Page 1</title> </head><body>
<script type="text/javascript">
// If the time is less than 10,write a "Good morning" greeting
// If time between 10 and 16 write a "Good day" greeting
// Otherwise "Hello world"
// Write the hour and If time <12 write AM else write PM
var computerdate = new Date()
var time = computerdate.getHours()
if (time<10)
{
document.write("<b>Good morning! Now is " +time+((time<12)?' AM':'
PM')+"</b>")
}
else if (time>10 && time<16)
{
document.write("<b>Good day! Now is " +time+((time<12)?' AM':' PM')+"</b>")
}
else
{
document.write("<b>Hello World! Now is " +time+((time<12)?' AM':'
PM')+"</b>")
}
</script> </body>
</html>
```

Switch. Execute one of several groups of statements depending on the value of an expression (called selector). The case structure (and statement) can is especially used when selection is based on the value of a single variable or a simple expression (called the case selector).

```
switch (expression_int) {
        case constant_expression1:
                operations1
        case constant_expression2:
                operations 2
        ⋮
        default:
                operations n
}
```

- *expression_int* is an expression that must produced an integral value (int);
- *constant_expression$_i$* must be a constant expression;
- the label **default:** can be used only once.
The expression_int is also called the selector of instruction Case.

- if the value of the selector don’t fit to a constant the operations specified on branch Default (otherwise) will be executed;
- the values of constants must be unique for a *switch sentence*.

Example:

The sequence below uses the switch statement to find out the Romanian name for the day of the week of a date.

```
<HTML>
<HEAD>
<meta name=vs_defaultClientScript content="JavaScript">
<TITLE></TITLE>
<META NAME="GENERATOR" Content="Microsoft Visual Studio">
<META HTTP-EQUIV="Content-Type" content="text/html” >
<script type="text/javascript">
/* The sequence will write the name of the day in romanian
     (Sunday=0, Monday=1, Tuesday=2, etc)
*/
function RODay(aDayNumber)
{
  switch (aDayNumber)
  {
    case 0:
      return "Duminica"
    case 1:
      return "Luni"
    case 2:
      return "Marti"
    case 3:
      return "Miercuri"
     case 4:
      return "Joi"
     case 5:
      return "Vineri"
    case 6:
      return "Sambata"
    default:
      alert("What day is it? \n The computer is virused or hardware damaged
!")
      return "What day is it? \n The computer is virused or hardware
damaged !"
  }
 }
</script>
<script id=clientEventHandlersJS language=javascript>
<!--
function Button1_onclick() {
  var i=0
  var datadeazi=new Date()
  var ziua=datadeazi.getDay()
  for (i=ziua;ziua<=6;ziua++){
     document.write(ziua+": " + RODay(ziua)+"<br />")
  }
}
//-->
</script>
<script language=javascript for=Button1 event=onclick>
<!--
return Button1_onclick()
//-->
</script>
</HEAD>
<BODY>
<p>This page contains a Java Script exploiting the switch sentence.</p>
<p>
 <input id=Button1 type=button value="Press This"></p>
</BODY>
</HTML>
```

Conditional Operator (?)

The conditional operator has the syntax:

(*conditional_expression*) *? true_case_expression: false_case_expression*

where:

<*conditional_expression*> is a logical expression that will be evaluated either to True or either to False. Is a very good idea to include the expression in parenthesis (to enforce his evaluation).

<*true_case_expression*> is the expression whose evaluation will be returned if the conditional expression evaluates to True

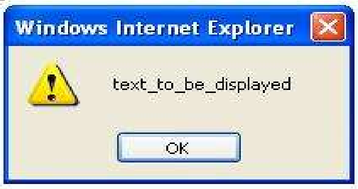
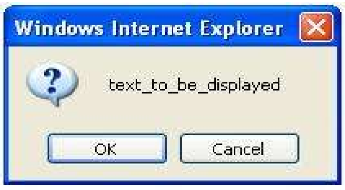
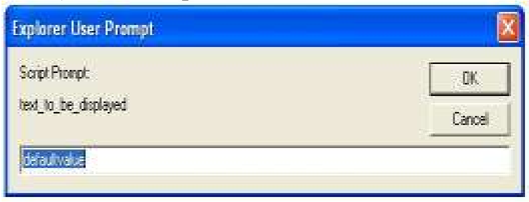
<*false_case_expression*> is the expression whose evaluation will be returned if the conditional expression evaluates to False.

Example:

```
<HTML> <HEAD>
<meta name=vs_defaultClientScript content="JavaScript">
<TITLE></TITLE>
<META NAME="GENERATOR" Content="Microsoft Visual Studio">
<META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-8">
</HEAD>
<BODY>
<script language=javascript>
  var TotalBalance, savings=300
  TotalBalance =(savings==0) ? 0:(savings*1.03)
   // TotalBalance is now 309
   document.write("Total Balance is now: " + TotalBalance)
</script>
</BODY></HTML>
```

7.10 Popup Boxes

In JavaScript we can create three kinds of popup boxes by invoking the associated intrinsic function. This functions are:

Function	Description
alert(”text_to_be_displayed”)	Displays an alert box containing the message passed in argument and an OK button. This call produces the box:  Windows Internet Explorer text_to_be_displayed OK
confirm(”text_to_be_displayed”)	Displays a box containing the message passed in argument provided with an OK (confirm) and Cancel (denny) button. This call produces the box:  Windows Internet Explorer text_to_be_displayed OK Cancel
prompt(”text_to_be_displayed”, ”defaultvalue”)	Displays an input dialog box provided with a text box to fill data and an OK (return the value typed to the caller) and Cancel (return a Null value to caller). This call produces the box:  Explorer User Prompt Script Prompt: text_to_be_displayed OK Cancel defaultvalue

7.11 Cycles

The repeating structure repeats a block of statements while a condition is True or Until a condition becomes True. The repetition of steps in a program is called a **loop**.

The repeated execution of a sequence of instructions (loop) can be done by using the looping sentences „while” , „do...while” and, for.

Example:

```
<html>
<head>
<title>
  Page containing loop
</title>
</head>
<body>
<script type="text/javascript">
<!--
   for (i=0; i<5; i++)
     {
      document.write("Step i: "+i+"<br>")
     }
//-->
</script>
</body>
</html>
```

In this example the values of i are written into the page „Step i:*the_vale_of_i*" from the value 0 to 4.

a) The condition evaluated first

- first syntax:

while (*condition*) *operation*;

- second syntax:

```
while (condition)
 {
   operations;
   [continue;]
   [break;]
 }
```

where:

- **continue** jump to the condition evaluation;
- **break** interrupt the cycle and transfer the execution to the sentence that follows to the end block marker }

Note. The ; character ending sentences is optional if the sentence is written alone on the line. It is necessary if you define mode sentences on the same line.

The executions of such blocks follow the scenario (while): the condition is evaluated and if the condition evaluates to:

• **True** then executes the block of statements;

• **False** then end the execution of the cycle (Loop) and continue the execution of the program.

If for the first time the condition is False the sentence block is simply skipped.

b) the condition evaluated after

```
do
  {
   operations;
} while conditions
```

In this case the operation is executed first and then the condition is evaluated and can be described as:

- the *operations* are executed;
- the *condition* is evaluated;
- **if** the result of the evaluation of the *condition* is False *then loop* to execute again the *operations*;
- **if** the evaluation of the *condition* is True *then* continue the execution of the program (and close the loop).

c) counted loop

for (*expression*$_1$; *expression*$_2$; *expression*$_3$) *operation*;

If many operations desired in the cycle they must be included as block;
expression$_1$ – is an expression that initializes the counter, having the general syntax *counter=startvalue*;
expression$_2$ – contains the definition for ending the loop, generally a logical condition of the form *counter<=endvalue*;
expression$_3$ – is an expression to increment or decrement the value for the counter, for example *counter=counter+increment*.

The cycle can be unconditionally stopped by using the instruction **break** and can be unconditionally restarted by using the sentence: **continue.**

Example:

for (*counter* = i_v; f_v; *s*) {*operations*};

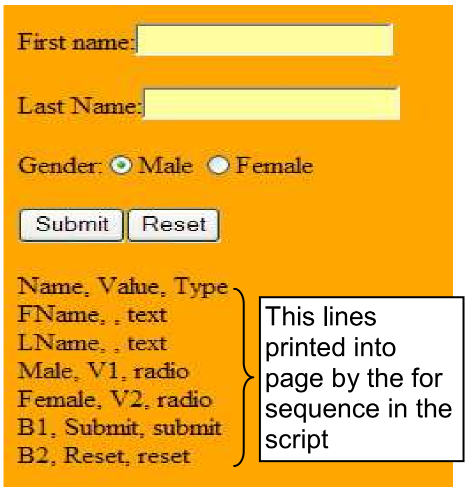
The execution of For sentence follows the scenario:

1. The value i_v is assigned to the variable *counter*;
2. The value of variable *counter* is compared with the end value f_v (the value can be determined by evaluating an expression);
3. The operations are executed;
4. The value of variable *counter* is incremented with the value step (1 if step not specified);
5. Repeat the steps from 2 to 5.

Example :

Figure 7.1 shows the usage of document object for accessing the forms collection and displaying, as comma separated values, the attributes Name, Value and Type.

```
<html>
<head>
<title>A form and
javascript</title>
</head>
<body>
<form method="POST"
action="--WEBBOT-SELF--">
  <!--webbot bot="SaveResults"
u-file="C:\Documents and
Settings\Vio\My Documents\My
Webs\_private\form_results.csv"
   s-format="TEXT/CSV" s-label-
fields="TRUE" -->
  <!-- This is the description of
the form -->
  <p>First name:<input
type="text" name="FName"
size="20"></p>
  <p>Last Name:<input type="text" name="LName" size="20"></p>
  <p>Gender:<input type="radio" value="V1" name="Male" checked>Male
  <input type="radio" name="Female" value="V2">Female</p>
  <p><input type="submit" value="Submit" name="B1">
        <input type="reset" value="Reset" name="B2"></p>
</form>
<script type="text/vbscript">
  document.write("Name, Value, Type "+"<br />")
</script>
<script type="text/javascript">
   for (i=0; i < document.forms[0].elements.length;i++)
     {
        document.write(document.forms[0].elements[i].name + ", ");
        //syntax below uses the name attribute of the form to access the
form's elements
        document.write(document.forms[0].elements[i].value + ", ");
        document.write(document.forms[0].elements[i].type + "<br />");
     }
</script>
</body>
</html>
```



```
Name, Value, Type
FName, , text
LName, , text
Male, V1, radio
Female, V2, radio
B1, Submit, submit
B2, Reset, reset
```

This lines printed into page by the for sequence in the script

Figure 7.1 Accessing HTML form elements

For … In statement

for (*variable in object*)

{ *code to be executed* }

Example :

In this example is defined an array object called divisions and the first three elements initialized. The for…in sentence will fill in the HTML document the lines initialized in the array.

```
html>
<body>

<script type="text/javascript">

var x, nr

var divisions = new Array()
divisions[0] = "English"
divisions[1] = "French"
divisions[2] = "German"

for (x in divisions)
{
nr=x/1+1;
document.write(nr+": "+divisions[x] + "<br />")
}
</script>

</body>
</html>
```

that produces the output :

```
1: English
2: French
3: German
```

7.12 Using events to trigger script execution

Some events that can be associated with HTML pages are represented by the following:

Event	**Occurs when...**
onabort	a user aborts page loading
onblur	a user leaves an object
onchange	a user changes the value of an object
onclick	a user clicks on an object
ondblclick	a user double-clicks on an object
onerror	an error occurs
onfocus	a user makes an object active
onkeydown	a keyboard key is on its way down

onkeypress	a keyboard key is pressed
onkeyup	a keyboard key is released
onload	a page is finished loading (in Netscape, this event occurs during the loading of a page).
onmousedown	a user presses a mouse-button
onmousemove	a cursor moves on an object
onmouseover	a cursor moves over an object
onmouseout	a cursor moves off an object
onmouseup	a user releases a mouse-button
onreset	a user resets a form
onselect	a user selects content on a page
onsubmit	a user submits a form
onunload	a user closes a page; a frequent usage is to deal with cookies.

The table below shows common usage of events:

Event	**Usage**
onload, onunload	The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information. Both events frequently used to deal with cookies.
onfocus, onblur, onchange	Generally used in combination with validation of form fields.
onsubmit	Is used to validate All form fields before submission (is possible to deal with logical validation involving more fields from the form).
onmouseover, onmouseout	Generally used for creating "animated" buttons.

In the following example is shown an inline JavaScript code (without the tags <script> and </script>). The web page contains a button whose property „Caption" has the value „ASE". When the event „onclick" occurs (when clicking the button) is called the function „open" (member of „windows" functions gruop), having in arguments the arguments required to open the ASE site home page in a window called internally „ase_home".

Example:

```
<html>
<head>
<title>
  Command button link
</title>
</head>
<body>
   <form>
   <input type="button" value="ASE" onclick='window.open("http://www.ase.ro",
"ase_home")'>
   </form>
</body>
</html>
```

Example:

```
<html>
<head>
<title>Pagina cu Cronometru </title>
<script type="text/javascript">
      function startTime()
      {
      var today=new Date()
      var h=today.getHours()
      var m=today.getMinutes()
      var s=today.getSeconds()
      // add a zero in front of numbers<10
      m=checkTime(m)
      s=checkTime(s)
      document.getElementById('txt').innerHTML=h+":"+m+":"+s
      t=setTimeout('startTime()',500)
      }
      function checkTime(i)
      {
      if (i<10)
        {i="0" + i}
        return i
      }
</script>
</head>
<body onload="startTime()">
<div id="txt" align=right></div>
<script type="text/javascript">
      var x, nr
      var divisions = new Array()
      divisions[0] = "English"
      divisions[1] = "French"
      divisions[2] = "German"
      for (x in divisions)
      {
      nr=x/1+1;
      document.write(nr+": "+divisions[x] + "<br />")
      }
</script>
</body>
</html>
```

7.13 Handling errors

In JavaScript are two ways for catching errors in a Web page:

- by using the **try…catch** statement;
- by using the **onerror** event.

Try…catch

The code you want prevent harassing user by error messages is included between try sentence and catch(err) sentence:

```
try
{
//Run some code here
}
catch(err)
{
//Handle errors here
}
```

Throw

Throw statement allows user to create an exception that can be catch and processed later on by try..catch. The syntax is:

```
throw(exception)
```

onerror event

The syntax for the **onerror** event and his associated error handler is:

```
Onerror=handleError

function handleError(msg, url, l)
{
// handle the error here
return true (success) or false (failure)
}
```

7.14 Commented samples

1. Displaying web content in new browser window

Below is a couple of two general functions:

ShowSmallWindow(from_uri,win_name, win_height, win_width, resize,scroll)
and
MakeCallString(to_uri,win_name, win_height, win_width, resize)

The function ShowSmallWindow opens a new window with a specified dimension and behavior to user interaction in which displays the content at the address indicated by the from_uri argument.
Required arguments:

- from_uri - in general the URL address of the resource to be displayed;
- win_name - the name of the window;
- win_high, win_width the high and, respectively, width of the window;
- resize - if resizing of the window allowed (Yes) or not (no);
- scroll - if the scroll bars will be available and viewable (Yes) or not (no).

```
function ShowSmallWindow(from_uri,win_name, win_height, win_width,
resize,scroll)
 {
    var sFeature;
    sFeature='directories=no, location=no, status=no, menubar=no, scrollbars=' +
    scroll + ', resizable=' + resize + ', toolbar=no, height=' +
            win_height + ', width=' + win_width;
    window.open(from_uri, win_name, sFeature);
 }
```

The function uses the open method available via DOM. The opened window do not have menu bar, status line, and toolbar and do not have the directory pannel and don’t displays the current location. A call example of that function is from the the index.htm page of the site http://www.avrams.ro:

```
<td title="The image is from Avrams Collection and called 'Horizon in the sunshine',
click to enlarge" valign="top" style="height: 46px; width: 746px; margin-top:0px;"
class="avbkgtop" onclick= 'ShowSmallWindow("http://avrams.ro/descript-
horizon.html", "HorizonInSunshine","640","656","no","no")'>
```

The function MakeCallString opens a new window with a specified dimension and behavior to user interaction in which displays the content at the relative address indicated by the user in a form field called ‘dnx’ prefixed by the from_uri argument, as a base address.

The function MakeCallString requires in arguments as follows:

- to_uri specifying the resource base address to which concatenate what user types in the text box called ‘dnx’ (the name=”dnx” and/or id=”dnx)”;
- win_name the name given to the opened window;
- the dimensions of that window (win_height, win_width);
- resize - if resizable (yes) or not (no).

```
function MakeCallString(to_uri,win_name, win_height, win_width, resize)
 {
   var from_uri;
   from_uri=to_uri + '/' + document.getElementById('dnx').value;
```

```
  ShowSmallWindow(from_uri,win_name, win_height, win_width,"yes","yes")
 }
```

2. Convert decimal integer numbers to a new power of two number base

This is the definition of a general function called Dec_To_NewBase(oldNumber, newBase, twoPower) whose required arguments means:

- oldNumber the decimal integer number you want convert in a new base;
- newBase the new base in which you want express the number;
- twoPower which power of two represents the number.

The variable Digits is a global variable (defined outside of any function body) of type string containing the digits in the new bases (here taken only 0 to f, as to cover binary, octal, and hexadecimal number bases).

The function can be changed to either:

- validate if truly the power of two is the power of the integer;
- determine if the integer is a power of two and which one, and to eliminate from the call model of twoPower argument.

```
var Digits="0123456789abcdef";

function Dec_To_NewBase(oldNumber, newBase, twoPower)
{
 var newNumber = Digits.substr(oldNumber & (newBase-1), 1);
 while(oldNumber > (newBase-1))
  {
    oldNumber >>= twoPower;
    newNumber = Digits.substr(oldNumber & (newBase-1), 1) + newNumber;
  }
 return newNumber;
}
```

The variable newNumber is reserved and initialized to the leftmost character of the result of applying the logical and ("anding") between the oldNumber and the highest digit in the new base, (newBase-1). If the oldNumber is greather than the (newBase-1) then a cycle computation started until oldNumber becomes less than (newBase-1).
In the cycle we have two operations:

a) oldNumber>>twoPower (shift right) which means divide the value of the content of variable called oldNumber to two at the power twoPower and store the quotient of the division in the variable oldNumber;
b) the new content of the variable newNumber is the concatenation between:
- the leftmost character of the string obtained by "anding" the oldNumber and the highest digit in the new base, (newBase-1);
- the old content of the variable newNumber.

When the cycle ends the new number value is returned to the caller and the function ends his execution.

The following functions are specialized and they implement a particular conversion, for that reason the values for newBase and twoPower are defined locally in the body of the function. They implemented in that way in the page *Convert-integer-numbers.html* of the website http://www.avrams.ro with the purpose to easy understanding the algorithm.
The code of that function is:

```
<script type="text/javascript" language=javascript>
<!--

    var Digits="0123456789abcdef";

/* Convert Decimal Integers (base 10) to Hexadecimal (base 16), two power 4 */
function dec2hex(d)
{
     var h = Digits.substr(d & 15, 1);
     while(d > 15)
       {
        d >>= 4;
        h = Digits.substr(d & 15, 1) + h;
       }
     return h;
}

/* Convert Decimal Integers (base 10) to Octal (base 8), two power 3 */
function dec2oct(d)
{
        var o = Digits.substr(d & 7, 1);
        while(d > 7)
        {
             d >>= 3;
             o = Digits.substr(d & 7, 1) + o;
        }
        return o;
}

/* Convert Decimal Integers (base 10) to Binary (base 2), two power 1 */
function dec2bin(d)
{
        var b = Digits.substr(d & 1, 1);
        while(d > 1)
        {
             d >>= 1;
             b = Digits.substr(d & 1, 1) + b;
        }
        return b;
}
 -->  </script>
```

3. Compute Easter date for a wanted year

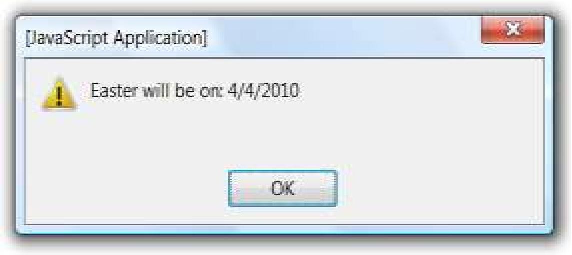
The function requires as in argument Wanted_Year containing the decimal integer representing the year for which you compute when Easter will be, computes the Easter date based on Gauss algorithm, and displays it. You can find another implementation and a description of the modeled algorithm in the page *compute-easter-date.html* of the website http://www.avrams.ro.

```
function Easter_Date(Wanted_Year){
    var D;
    var E;
    if (Wanted_Year<0){
        alert("The value for Year must be a positive number!");
        return -1;
    }
    D = ((Wanted_Year % 19) * 19 + 15) % 30;
    E = (((Wanted_Year % 4) * 2 + (Wanted_Year % 7) * 4) + 6 * D + 6) % 7;
    D=D+E+4;
    if(D>30){
      alert( ‘Easter will be on: ‘+'5/'+(D-30.)+'/'+Wanted_Year);
      }
      else
      {
      alert( ‘Easter will be on: ‘+'4/'+(D)+'/'+Wanted_Year);
      }
}
```

In the line bellow the press of the button will trigger the call of the Easter_Date function (the event onclick) having in argument the year 2010:

```
<input          name="callEaster"          type="button"          value="Easter"
onclick="Easter_Date(2010)" />
```

and will produce the output (the alert call):


[JavaScript Application]

Easter will be on: 4/4/2010

Annexes

Annex 1. List of VBScript intrinsic functions

Date/Time Functions

Function	Description
CDate	Converts a valid date and time expression to the variant of subtype Date
Date	Returns the current system date
DateAdd	Returns a date to which a specified time interval has been added
DateDiff	Returns the number of intervals between two dates
DatePart	Returns the specified part of a given date
DateSerial	Returns the date for a specified year, month, and day
DateValue	Returns a date
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
FormatDateTime	Returns an expression formatted as a date or time
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
MonthName	Returns the name of a specified month
Now	Returns the current system date and time
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
Time	Returns the current system time
Timer	Returns the number of seconds since 12:00 AM
TimeSerial	Returns the time for a specific hour, minute, and second
TimeValue	Returns a time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)
WeekdayName	Returns the weekday name of a specified day of the week
Year	Returns a number that represents the year

Conversion Functions

Function	Description
Asc	Converts the first letter in a string to ANSI code
CBool	Converts an expression to a variant of subtype Boolean
CByte	Converts an expression to a variant of subtype Byte
CCur	Converts an expression to a variant of subtype Currency
CDate	Converts a valid date and time expression to the variant of subtype Date
CDbl	Converts an expression to a variant of subtype Double
Chr	Converts the specified ANSI code to a character
CInt	Converts an expression to a variant of subtype Integer
CLng	Converts an expression to a variant of subtype Long
CSng	Converts an expression to a variant of subtype Single
CStr	Converts an expression to a variant of subtype String
Hex	Returns the hexadecimal value of a specified number
Oct	Returns the octal value of a specified number

Format Functions

Function	Description
FormatCurrency	Returns an expression formatted as a currency value
FormatDateTime	Returns an expression formatted as a date or time
FormatNumber	Returns an expression formatted as a number
FormatPercent	Returns an expression formatted as a percentage

Math Functions

Function	Description
Abs	Returns the absolute value of a specified number
Atn	Returns the arctangent of a specified number
Cos	Returns the cosine of a specified number (angle)
Exp	Returns e raised to a power
Hex	Returns the hexadecimal value of a specified number
Int	Returns the integer part of a specified number
Fix	Returns the integer part of a specified number
Log	Returns the natural logarithm of a specified number
Oct	Returns the octal value of a specified number
Rnd	Returns a random number less than 1 but greater or equal to 0
Sgn	Returns an integer that indicates the sign of a specified number
Sin	Returns the sine of a specified number (angle)
Sqr	Returns the square root of a specified number
Tan	Returns the tangent of a specified number (angle)

Array Functions

Function	Description
Array	Returns a variant containing an array
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria
IsArray	Returns a Boolean value that indicates whether a specified variable is an array
Join	Returns a string that consists of a number of substrings in an array
LBound	Returns the smallest subscript for the indicated dimension of an array
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings
UBound	Returns the largest subscript for the indicated dimension of an array

String Functions

Function	Description
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string
LCase	Converts a specified string to lowercase
Left	Returns a specified number of characters from the left side of a string
Len	Returns the number of characters in a string
LTrim	Removes spaces on the left side of a string
RTrim	Removes spaces on the right side of a string
Trim	Removes spaces on both the left and the right side of a string
Mid	Returns a specified number of characters from a string
Replace	Replaces a specified part of a string with another string a specified number of times
Right	Returns a specified number of characters from the right side of a string
Space	Returns a string that consists of a specified number of spaces
StrComp	Compares two strings and returns a value that represents the result of the comparison
String	Returns a string that contains a repeating character of a specified length
StrReverse	Reverses a string
UCase	Converts a specified string to uppercase

Other Functions

Function	Description
CreateObject	Creates an object of a specified type
Eval	Evaluates an expression and returns the result
GetLocale	Returns the current locale ID
GetObject	Returns a reference to an automation object from a file
GetRef	Allows you to connect a VBScript procedure to a DHTML event on your pages
InputBox	Displays a dialog box, where the user can write some input and/or click on a button, and returns the contents
IsEmpty	Returns a Boolean value that indicates whether a specified variable has been initialized or not
IsNull	Returns a Boolean value that indicates whether a specified expression contains no valid data (Null)
IsNumeric	Returns a Boolean value that indicates whether a specified expression can be evaluated as a number
IsObject	Returns a Boolean value that indicates whether the specified expression is an automation object
LoadPicture	Returns a picture object. Available only on 32-bit platforms
MsgBox	Displays a message box, waits for the user to click a button, and returns a value that indicates which button the user clicked
RGB	Returns a number that represents an RGB color value
Round	Rounds a number
ScriptEngine	Returns the scripting language in use
ScriptEngineBuildVersion	Returns the build version number of the scripting engine in use
ScriptEngineMajorVersion	Returns the major version number of the scripting engine in use
ScriptEngineMinorVersion	Returns the minor version number of the scripting engine in use
SetLocale	Sets the locale ID and returns the previous locale ID
TypeName	Returns the subtype of a specified variable
VarType	Returns a value that indicates the subtype of a specified variable

Annex 2 VBScript naming conventions

Object type	Prefix	Example
3D Panel	pnl	pnlGroup
Animated button	ani	aniMailBox
Check box	chk	chkReadOnly
Combo box, drop-down list box	cbo	cboEnglish
Command button	cmd	cmdExit
Common dialog	dlg	dlgFileOpen
Frame	fra	fraLanguage
Horizontal scroll bar	hsb	hsbVolume
Image	img	imgIcon
Label	lbl	lblHelpMessage
Line	lin	linVertical
List Box	lst	lstPolicyCodes
Spin	spn	spnPages
Text box	txt	txtLastName
Vertical scroll bar	vsb	vsbRate
Slider	sld	sldScale

Annex 3 Character sets and RGB color

A. Character entities

The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;
'	apostrophe	&apos; (does not work in IE)	&#39;

Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

B. ISO Latin-1 Character Set

Character	Decimal code	Named entity	Description
---	&#00;	---	Unused
---	&#01;	---	Unused
---	&#02;	---	Unused
---	&#03;	---	Unused
---	&#04;	---	Unused
---	&#05;	---	Unused
---	&#06;	---	Unused
---	&#07;	---	Unused
---	&#08;	---	Unused
---	&#09;	---	Horizontal tab
---	&#10;	---	Line feed
---	&#11;	---	Unused
---	&#12;	---	Unused
---	&#13;	---	Carriage Return

---	&#14;	---	Unused
---	&#15;	---	Unused
---	&#16;	---	Unused
---	&#17;	---	Unused
---	&#18;	---	Unused
---	&#19;	---	Unused
---	&#20;	---	Unused
---	&#21;	---	Unused
---	&#22;	---	Unused
---	&#23;	---	Unused
---	&#24;	---	Unused
---	&#25;	---	Unused
---	&#26;	---	Unused
---	&#27;	---	Unused
---	&#28;	---	Unused
---	&#29;	---	Unused
---	&#30;	---	Unused
---	&#31;	---	Unused
	&#32;	---	Space
!	&#33;	---	Exclamation mark
"	&#34;	&quot;	Quotation mark
#	&#35;	---	Number sign
$	&#36;	---	Dollar sign
%	&#37;	---	Percent sign
&	&#38;	&amp;	Ampersand
'	&#39;	---	Apostrophe
(	&#40;	---	Left parenthesis
)	&#41;	---	Right parenthesis
*	&#42;	---	Asterisk
+	&#43;	---	Plus sign
,	&#44;	---	Comma
-	&#45;	---	Hyphen
.	&#46;	---	Period (fullstop)
/	&#47;	---	Solidus (slash)
0	&#48;	---	Digit 0
1	&#49;	---	Digit 1
2	&#50;	---	Digit 2
3	&#51;	---	Digit 3
4	&#52;	---	Digit 4
5	&#53;	---	Digit 5
6	&#54;	---	Digit 6
7	&#55;	---	Digit 7
8	&#56;	---	Digit 8
9	&#57;	---	Digit 9
:	&#58;	---	Colon
;	&#59;	---	Semicolon
<	&#60;	&lt;	Less than

=	&#61;	---	Equals sign
>	&#62;	&gt;	Greater than
?	&#63;	---	Question mark
@	&#64;	---	Commercial at
A	&#65;	---	Capital A
B	&#66;	---	Capital B
C	&#67;	---	Capital C
D	&#68;	---	Capital D
E	&#69;	---	Capital E
F	&#70;	---	Capital F
G	&#71;	---	Capital G
H	&#72;	---	Capital H
I	&#73;	---	Capital I
J	&#74;	---	Capital J
K	&#75;	---	Capital K
L	&#76;	---	Capital L
M	&#77;	---	Capital M
N	&#78;	---	Capital N
O	&#79;	---	Capital O
P	&#80;	---	Capital P
Q	&#81;	---	Capital Q
R	&#82;	---	Capital R
S	&#83;	---	Capital S
T	&#84;	---	Capital T
U	&#85;	---	Capital U
V	&#86;	---	Capital V
W	&#87;	---	Capital W
X	&#88;	---	Capital X
Y	&#89;	---	Capital Y
Z	&#90;	---	Capital Z
[	&#91;	---	Left square bracket
\	&#92;	---	Reverse solidus (backslash)
]	&#93;	---	Right square bracket
^	&#94;	---	Caret
_	&#95;	---	Horizontal bar (underscore)
`	&#96;	---	Acute accent
a	&#97;	---	Small a
b	&#98;	---	Small b
c	&#99;	---	Small c
d	&#100;	---	Small d
e	&#101;	---	Small e
f	&#102;	---	Small f
g	&#103;	---	Small g
h	&#104;	---	Small h
i	&#105;	---	Small i

j	&#106;	---	Small j
k	&#107;	---	Small k
l	&#108;	---	Small l
m	&#109;	---	Small m
n	&#110;	---	Small n
o	&#111;	---	Small o
p	&#112;	---	Small p
q	&#113;	---	Small q
r	&#114;	---	Small r
s	&#115;	---	Small s
t	&#116;	---	Small t
u	&#117;	---	Small u
v	&#118;	---	Small v
w	&#119;	---	Small w
x	&#120;	---	Small x
y	&#121;	---	Small y
z	&#122;	---	Small z
{	&#123;	---	Left curly brace
|	&#124;	---	Vertical bar
}	&#125;	---	Right curly brace
~	&#126;	---	Tilde
☐	&#127;	---	Unused
€	&#128;	---	Unused
	&#160;	&nbsp;	Nonbreaking space
¡	&#161;	&iexcl;	Inverted exclamation
¢	&#162;	&cent;	Cent sign
£	&#163;	&pound;	Pound sterling
¤	&#164;	&curren;	General currency sign
¥	&#165;	&yen;	Yen sign
¦	&#166;	&brvbar; or &brkbar;	Broken vertical bar
§	&#167;	&&sect;	Section sign
¨	&#168;	&&um; or &&die;	Diæresis / Umlaut
©	&#169;	&&copy;	Copyright
ª	&#170;	&&ordf;	Feminine ordinal
«	&#171;	&&laquo;	Left angle quote, guillemot left
¬	&#172;	&&not	Not sign
	&#173;	&shy;	Soft hyphen
®	&#174;	&reg;	Registered trademark
¯	&#175;	&macr; or &hibar;	Macron accent
°	&#176;	&deg;	Degree sign
±	&#177;	&plusmn;	Plus or minus
²	&#178;	&sup2;	Superscript two
³	&#179;	&sup3;	Superscript three

´	&#180;	&acute;	Acute accent
µ	&#181;	&micro;	Micro sign
¶	&#182;	&para;	Paragraph sign
·	&#183;	&middot;	Middle dot
¸	&#184;	&cedil;	Cedilla
¹	&#185;	&sup1;	Superscript one
º	&#186;	&ordm;	Masculine ordinal
»	&#187;	&raquo;	Right angle quote, guillemot right
¼	&#188;	&frac14;	Fraction one-fourth
½	&#189;	&frac12;	Fraction one-half
¾	&#190;	&frac34;	Fraction three-fourths
¿	&#191;	&iquest;	Inverted question mark
À	&#192;	&Agrave;	Capital A, grave accent
Á	&#193;	&Aacute;	Capital A, acute accent
Â	&#194;	&Acirc;	Capital A, circumflex
Ã	&#195;	&Atilde;	Capital A, tilde
Ä	&#196;	&Auml;	Capital A, diæresis / umlaut
Å	&#197;	&Aring;	Capital A, ring
Æ	&#198;	&AElig;	Capital AE ligature
Ç	&#199;	&Ccedil;	Capital C, cedilla
È	&#200;	&Egrave;	Capital E, grave accent
É	&#201;	&Eacute;	Capital E, acute accent
Ê	&#202;	&Ecirc;	Capital E, circumflex
Ë	&#203;	&Euml;	Capital E, diæresis / umlaut
Ì	&#204;	&Igrave;	Capital I, grave accent
Í	&#205;	&Iacute;	Capital I, acute accent
Î	&#206;	&Icirc;	Capital I, circumflex
Ï	&#207;	&Iuml;	Capital I, diæresis / umlaut
Ð	&#208;	&ETH;	Capital Eth, Icelandic
Ñ	&#209;	&Ntilde;	Capital N, tilde
Ò	&#210;	&Ograve;	Capital O, grave accent
Ó	&#211;	&Oacute;	Capital O, acute accent
Ô	&#212;	&Ocirc;	Capital O, circumflex
Õ	&#213;	&Otilde;	Capital O, tilde
Ö	&#214;	&Ouml;	Capital O, diæresis / umlaut
×	&#215;	&times;	Multiply sign

Ø	&#216;	&Oslash;	Capital O, slash
Ù	&#217;	&Ugrave;	Capital U, grave accent
Ú	&#218;	&Uacute;	Capital U, acute accent
Û	&#219;	&Ucirc;	Capital U, circumflex
Ü	&#220;	&Uuml;	Capital U, diæresis / umlaut
Ý	&#221;	&Yacute;	Capital Y, acute accent
Þ	&#222;	&THORN;	Capital Thorn, Icelandic
ß	&#223;	&szlig;	Small sharp s, German sz
à	&#224;	&agrave;	Small a, grave accent
á	&#225;	&aacute;	Small a, acute accent
â	&#226;	&acirc;	Small a, circumflex
ã	&#227;	&atilde;	Small a, tilde
ä	&#228;	&auml;	Small a, diæresis / umlaut
å	&#229;	&aring;	Small a, ring
æ	&#230;	&aelig;	Small ae ligature
ç	&#231;	&ccedil;	Small c, cedilla
è	&#232;	&egrave;	Small e, grave accent
é	&#233;	&eacute;	Small e, acute accent
ê	&#234;	&ecirc;	Small e, circumflex
ë	&#235;	&euml;	Small e, diæresis / umlaut
ì	&#236;	&igrave;	Small i, grave accent
í	&#237;	&iacute;	Small i, acute accent
î	&#238;	&icirc;	Small i, circumflex
ï	&#239;	&iuml;	Small i, diæresis / umlaut
ð	&#240;	&eth;	Small eth, Icelandic
ñ	&#241;	&ntilde;	Small n, tilde
ò	&#242;	&ograve;	Small o, grave accent
ó	&#243;	&oacute;	Small o, acute accent
ô	&#244;	&ocirc;	Small o, circumflex
õ	&#245;	&otilde;	Small o, tilde
ö	&#246;	&ouml;	Small o, diæresis / umlaut
÷	&#247;	&divide;	Division sign
ø	&#248;	&oslash;	Small o, slash
ù	&#249;	&ugrave;	Small u, grave accent
ú	&#250;	&uacute;	Small u, acute accent
û	&#251;	&ucirc;	Small u, circumflex
ü	&#252;	&uuml;	Small u, diæresis /

			umlaut
ý	&#253;	&yacute;	Small y, acute accent
þ	&#254;	&thorn;	Small thorn, Icelandic
ÿ	&#255;	&yuml;	Small y, diæresis / umlaut

To use one of the three characters in an HTML document, you must enter its escape sequence instead:

&lt;	the escape sequence for <
&gt;	the escape sequence for >
&amp;	the escape sequence for &

Additional escape sequences support accented characters, such as:

&ouml;	a lowercase o with an umlaut: ö
&ntilde;	a lowercase n with a tilde: ñ
&Egrave;	an uppercase E with a grave accent: È

You can substitute other letters for the o, n, and E shown above. Visit the World Wide Web Consortium for a complete list of special characters.

NOTE: Unlike the rest of HTML, the escape sequences are case sensitive. You cannot, for instance, use &LT; instead of &lt;.

Character Entities for Special Symbols and BIDI Text:

C0 Controls and Basic Latin			
&quot;	&quot	&#34;	quotation mark, =apl quote, U0022 ISOnum
&	&amp	&#38;	ampersand, U0026 ISOnum
<	&lt	&#60;	less-than sign, U003C ISOnum
>	&gt	&#62;	greater-than sign, U003E ISOnum
Latin Extended-A			
&OElig;	&OElig	&#338;	latin capital ligature oe, U0152 ISOlat2
&oelig;	&oelig	&#339;	latin small ligature oe, U0153 ISOlat2
&Scaron;	&Scaron	&#352;	latin capital letter s with caron, U0160 ISOlat2
&scaron;	&scaron	&#353;	latin small letter s with caron, U0161 ISOlat2
&Yuml;	&Yuml	&#376;	latin capital letter y with diaeresis, U0178 ISOlat2
Spacing Modifier Letters			
&circ;	&circ	&#710;	modifier letter circumflex accent, U02C6 ISOpub
&tilde;	&tilde	&#732;	small tilde, U02DC ISOdia
General Punctuation			
&ensp;	&ensp	&#8194;	en space, U2002 ISOpub
&emsp;	&emsp	&#8195;	em space, U2003 ISOpub
&thinsp;	&thinsp	&#8201;	thin space, U2009 ISOpub
&zwnj;	&zwnj	&#8204;	zero width non-joiner, U200C NEW RFC 2070
&zwj;	&zwj	&#8205;	zero width joiner, U200D NEW RFC 2070

&lrm;	&lrm	&#8206;	left-to-right mark, U200E NEW RFC 2070
&rlm;	&rlm	&#8207;	right-to-left mark, U200F NEW RFC 2070
&ndash;	&ndash	&#8211;	en dash, U2013 ISOpub
&mdash;	&mdash	&#8212;	em dash, U2014 ISOpub
&lsquo;	&lsquo	&#8216;	left single quotation mark, U2018 ISOnum
&rsquo;	&rsquo	&#8217;	right single quotation mark, U2019 ISOnum
&sbquo;	&sbquo	&#8218;	single low-9 quotation mark, U201A NEW
&ldquo;	&ldquo	&#8220;	left double quotation mark, U201C ISOnum
&rdquo;	&rdquo	&#8221;	right double quotation mark, U201D ISOnum
&bdquo;	&bdquo	&#8222;	double low-9 quotation mark, U201E NEW
&dagger;	&dagger	&#8224;	dagger, U2020 ISOpub
&Dagger;	&Dagger	&#8225;	double dagger, U2021 ISOpub
&permil;	&permil	&#8240;	per mille sign, U2030 ISOtech
&lsaquo;	&lsaquo	&#8249;	single left-pointing angle quotation mark, U2039 ISO proposed
&rsaquo;	&rsaquo	&#8250;	single right-pointing angle quotation mark, U203A ISO proposed

C. Additional Named Entities for HTML

Character	Named entity	Numeric character reference	Description
Latin Extended-B			
&fnof;	&fnof	&#402;	latin small f with hook, =function, =flavia, U0192 ISOtech
Greek			
&Alpha;	&Alpha	&#913;	greek capital letter alpha, U0391
&Beta;	&Beta	&#914;	greek capital letter beta, U0392
&Gamma;	&Gamma	&#915;	greek capital letter gamma, U0393 ISOgrk3
&Delta;	&Delta	&#916;	greek capital letter delta, U0394 ISOgrk3
&Epsilon;	&Epsilon	&#917;	greek capital letter epsilon, U0395
&Zeta;	&Zeta	&#918;	greek capital letter zeta, U0396
&Eta;	&Eta	&#919;	greek capital letter eta, U0397
&Theta;	&Theta	&#920;	greek capital letter theta, U0398 ISOgrk3
&Iota;	&Iota	&#921;	greek capital letter iota, U0399
&Kappa;	&Kappa	&#922;	greek capital letter kappa, U039A
&Lambda;	&Lambda	&#923;	greek capital letter lambda, U039B ISOgrk3
&Mu;	&Mu	&#924;	greek capital letter mu, U039C
&Nu;	&Nu	&#925;	greek capital letter nu, U039D
&Xi;	&Xi	&#926;	greek capital letter xi, U039E ISOgrk3

&Omicron;	&Omicron	&#927;	greek capital letter omicron, U039F
&Pi;	&Pi	&#928;	greek capital letter pi, U03A0 ISOgrk3
&Rho;	&Rho	&#929;	greek capital letter rho, U03A1
&Sigma;	&Sigma	&#931;	greek capital letter sigma, U03A3 ISOgrk3
&Tau;	&Tau	&#932;	greek capital letter tau, U03A4
&Upsilon;	&Upsilon	&#933;	greek capital letter upsilon, U03A5 ISOgrk3
&Phi;	&Phi	&#934;	greek capital letter phi, U03A6 ISOgrk3
&Chi;	&Chi	&#935;	greek capital letter chi, U03A7
&Psi;	&Psi	&#936;	greek capital letter psi, U03A8 ISOgrk3
&Omega;	&Omega	&#937;	greek capital letter omega, U03A9 ISOgrk3
a	&alpha	&#945;	greek small letter alpha, U03B1 ISOgrk3
b	&beta	&#946;	greek small letter beta, U03B2 ISOgrk3
g	&gamma	&#947;	greek small letter gamma, U03B3 ISOgrk3
d	&delta	&#948;	greek small letter delta, U03B4 ISOgrk3
&epsilon;	&epsilon	&#949;	greek small letter epsilon, U03B5 ISOgrk3
&zeta;	&zeta	&#950;	greek small letter zeta, U03B6 ISOgrk3
&eta;	&eta	&#951;	greek small letter eta, U03B7 ISOgrk3
&theta;	&theta	&#952;	greek small letter theta, U03B8 ISOgrk3
&iota;	&iota	&#953;	greek small letter iota, U03B9 ISOgrk3
&kappa;	&kappa	&#954;	greek small letter kappa, U03BA ISOgrk3
&lambda;	&lambda	&#955;	greek small letter lambda, U03BB ISOgrk3
&mu;	&mu	&#956;	greek small letter mu, U03BC ISOgrk3
&nu;	&nu	&#957;	greek small letter nu, U03BD ISOgrk3
&xi;	&xi	&#958;	greek small letter xi, U03BE ISOgrk3
&omicron;	&omicron	&#959;	greek small letter omicron, U03BF NEW
p	&pi	&#960;	greek small letter pi, U03C0

			ISOgrk3
&rho;	&rho	&#961;	greek small letter rho, U03C1 ISOgrk3
&sigmaf;	&sigmaf	&#962;	greek small letter final sigma, U03C2 ISOgrk3
&sigma;	&sigma	&#963;	greek small letter sigma, U03C3 ISOgrk3
&tau;	&tau	&#964;	greek small letter tau, U03C4 ISOgrk3
&upsilon;	&upsilon	&#965;	greek small letter upsilon, U03C5 ISOgrk3
&phi;	&phi	&#966;	greek small letter phi, U03C6 ISOgrk3
&chi;	&chi	&#967;	greek small letter chi, U03C7 ISOgrk3
&psi;	&psi	&#968;	greek small letter psi, U03C8 ISOgrk3
&omega;	&omega	&#969;	greek small letter omega, U03C9 ISOgrk3
&thetasym;	&thetasym	&#977;	greek small letter theta symbol, U03D1 NEW
&upsih;	&upsih	&#978;	greek upsilon with hook symbol, U03D2 NEW
&piv;	&piv	&#982;	greek pi symbol, U03D6 ISOgrk3
General Punctuation			
&bull;	&bull	&#8226;	bullet, =black small circle, U2022 ISOpub
&hellip;	&hellip	&#8230;	horizontal ellipsis, =three dot leader, U2026 ISOpub
&prime;	&prime	&#8242;	prime, =minutes, =feet, U2032 ISOtech
&Prime;	&Prime	&#8243;	double prime, =seconds, =inches, U2033 ISOtech
&oline;	&oline	&#8254;	overline, =spacing overscore, U203E NEW
&frasl;	&frasl	&#8260;	fraction slash, U2044 NEW
Letterlike Symbols			
&weierp;	&weierp	&#8472;	script capital P, =power set, =Weierstrass p, U2118 ISOamso
&image;	&image	&#8465;	blackletter capital I, =imaginary part, U2111 ISOamso
&real;	&real	&#8476;	blackletter capital R, =real part symbol, U211C ISOamso
&trade;	&trade	&#8482;	trade mark sign, U2122 ISOnum
&alefsym;	&alefsym	&#8501;	alef symbol, =first transfinite cardinal, U2135 NEW

Arrows			
&larr;	&larr	&#8592;	leftward arrow, U2190 ISOnum
&uarr;	&uarr	&#8593;	upward arrow, U2191 ISOnum
&rarr;	&rarr	&#8594;	rightward arrow, U2192 ISOnum
&darr;	&darr	&#8595;	downward arrow, U2193 ISOnum
&harr;	&harr	&#8596;	left right arrow, U2194 ISOamsa
&crarr;	&crarr	&#8629;	downward arrow with corner leftward, =carriage return, U21B5 NEW
&lArr;	&lArr	&#8656;	leftward double arrow, U21D0 ISOtech
&uArr;	&uArr	&#8657;	upward double arrow, U21D1 ISOamsa
&rArr;	&rArr	&#8658;	rightward double arrow, U21D2 ISOtech
&dArr;	&dArr	&#8659;	downward double arrow, U21D3 ISOamsa
&hArr;	&hArr	&#8660;	left right double arrow, U21D4 ISOamsa
Mathematical Operators			
&forall;	&forall	&#8704;	for all, U2200 ISOtech
&part;	&part	&#8706;	partial differential, U2202 ISOtech
&exist;	&exist	&#8707;	there exists, U2203 ISOtech
&empty;	&empty	&#8709;	empty set, =null set, =diameter, U2205 ISOamso
&nabla;	&nabla	&#8711;	nabla, =backward difference, U2207 ISOtech
&isin;	&isin	&#8712;	element of, U2208 ISOtech
&notin;	&notin	&#8713;	not an element of, U2209 ISOtech
&ni;	&ni	&#8715;	contains as member, U220B ISOtech
&prod;	&prod	&#8719;	n-ary product, =product sign, U220F ISOamsb
&sum;	&sum	&#8722;	n-ary sumation, U2211 ISOamsb
&minus;	&minus	&#8722;	minus sign, U2212 ISOtech
&lowast;	&lowast	&#8727;	asterisk operator, U2217 ISOtech
&radic;	&radic	&#8730;	square root, =radical sign, U221A ISOtech
&prop;	&prop	&#8733;	proportional to, U221D ISOtech
&infin;	&infin	&#8734;	infinity, U221E ISOtech
&ang;	&ang	&#8736;	angle, U2220 ISOamso
&and;	&and	&#8869;	logical and, =wedge, U2227

			ISOtech
&or;	&or	&#8870;	logical or, =vee, U2228 ISOtech
&cap;	&cap	&#8745;	intersection, =cap, U2229 ISOtech
&cup;	&cup	&#8746;	union, =cup, U222A ISOtech
&int;	&int	&#8747;	integral, U222B ISOtech
&there4;	&there4	&#8756;	therefore, U2234 ISOtech
&sim;	&sim	&#8764;	tilde operator, =varies with, =similar to, U223C ISOtech
&cong;	&cong	&#8773;	approximately equal to, U2245 ISOtech
&asymp;	&asymp	&#8773;	almost equal to, =asymptotic to, U2248 ISOamsr
&ne;	&ne	&#8800;	not equal to, U2260 ISOtech
&equiv;	&equiv	&#8801;	identical to, U2261 ISOtech
&le;	&le	&#8804;	less-than or equal to, U2264 ISOtech
&ge;	&ge	&#8805;	greater-than or equal to, U2265 ISOtech
&sub;	&sub	&#8834;	subset of, U2282 ISOtech
&sup;	&sup	&#8835;	superset of, U2283 ISOtech
&nsub;	&nsub	&#8836;	not a subset of, U2284 ISOamsn
&sube;	&sube	&#8838;	subset of or equal to, U2286 ISOtech
&supe;	&supe	&#8839;	superset of or equal to, U2287 ISOtech
&oplus;	&oplus	&#8853;	circled plus, =direct sum, U2295 ISOamsb
&otimes;	&otimes	&#8855;	circled times, =vector product, U2297 ISOamsb
&perp;	&perp	&#8869;	up tack, =orthogonal to, =perpendicular, U22A5 ISOtech
&sdot;	&sdot	&#8901;	dot operator, U22C5 ISOamsb
Miscellaneous Technical			
&lceil;	&lceil	&#8968;	left ceiling, =apl upstile, U2308, ISOamsc
&rceil;	&rceil	&#8969;	right ceiling, U2309, ISOamsc
&lfloor;	&lfloor	&#8970;	left floor, =apl downstile, U230A, ISOamsc
&rfloor;	&rfloor	&#8971;	right floor, U230B, ISOamsc
&lang;	&lang	&#9001;	left-pointing angle bracket, =bra, U2329 ISOtech
&rang;	&rang	&#9002;	right-pointing angle bracket, =ket, U232A ISOtech
Geometric Shapes			
&loz;	&loz	&#9674;	lozenge, U25CA ISOpub

Miscellaneous Symbols			
&spades;	&spades	&#9824;	black spade suit, U2660 ISOpub
&clubs;	&clubs	&#9827;	black club suit, =shamrock, U2663 ISOpub
&hearts;	&hearts	&#9829;	black heart suit, =valentine, U2665 ISOpub
&diams;	&diams	&#9830;	black diamond suit, U2666 ISOpub

D. RGB Color codes

NAME	**RGB Value**	**NAME**	**RGB Value**
ALICEBLUE	#F0F8FF	ANTIQUEWHITE	#FAEBD7
AQUA	#00FFFF	AQUAMARINE	#7FFFD4
AZURE	#F0FFFF	BEIGE	#F5F5DC
BISQUE	#FFE4C4	BLACK	#000000
BLANCHEDALMOND	#FFEBCD	BLUE	#0000FF
BLUEVIOLET	#8A2BE2	BROWN	#A52A2A
BURLYWOOD	#DEB887	CADETBLUE	#5F9EA0
CHARTREUSE	#7FFF00	CHOCOLATE	#D2691E
CORAL	#FF7F50	CORNFLOWER	#6495ED
CORNSILK	#FFF8DC	CRIMSON	#DC143C
CYAN	#00FFFF	DARKBLUE	#00008B
DARKCYAN	#008B8B	DARKGOLDENROD	#B8860B
DARKGRAY	#A9A9A9	DARKGREEN	#006400
DARKKHAKI	#BDB76B	DARKMAGENTA	#8B008B
DARKOLIVEGREEN	#556B2F	DARKORANGE	#FF8C00
DARKORCHID	#9932CC	DARKRED	#8B0000
DARKSALMON	#E9967A	DARKSEAGREEN	#8FBC8B
DARKSLATEBLUE	#483D8B	DARKSLATEGREY	#2F4F4F
DARKTURQUOISE	#00CED1	DARKVIOLET	#9400D3
DEEPPINK	#FF1493	DEEPSKYBLUE	#00BFFF
DIMGRAY	#696969	DODGERBLUE	#1E90FF
FIREBRICK	#B22222	FLORALWHITE	#FFFAF0
FORESTGREEN	#228B22	FUCHIA	#FF00FF
GAINSBORO	#DCDCDC	GHOSTWHITE	#F8F8FF
GOLD	#FFD700	GOLDENROD	#DAA520
GRAY	#808080	GREEN	#008000
GREENYELLOW	#ADFF2F	HONEYDEW	#F0FFF0
HOTPINK	#FF69B4	INDIANRED	#CD5C5C
INDIGO	#4B0082	IVORY	#FFFFF0
KHAKI	#F0E68C	LAVENDER	#E6E6FA
LAVENDERBLUSH	#FFF0F5	LAWNGREEN	#7CFC00
LEMONCHIFFON	#FFFACD	LIGHTBLUE	#ADD8E6
LIGHTCORAL	#F08080	LIGHTCYAN	#E0FFFF
LIGHTGOLDENRODYELLOW	#FAFAD2	LIGHTGREEN	#90EE90
LIGHTGREY	#D3D3D3	LIGHTPINK	#FFB6C1
LIGHTSALMON	#FFA07A	LIGHTSEAGREEN	#20B2AA
LIGHTSKYBLUE	#87CEFA	LIGHTSLATEGRAY	#778899
LIGHTSTEELBLUE	#B0C4DE	LIGHTYELLOW	#FFFFE0

LIME	#00FF00	LIMEGREEN	#32CD32
LINEN	#FAF0E6	MAGENTA	#FF00FF
MAROON	#800000	MEDIUMAQUAMARINE	#66CDAA
MEDIUMBLUE	#0000CD	MEDIUMORCHID	#BA55D3
MEDIUMPURPLE	#9370DB	MEDIUMSEAGREEN	#3CB371
MEDIUMSLATEBLUE	#7B68EE	MEDIUMSPRINGGREEN	#00FA9A
MEDIUMTURQUOISE	#48D1CC	MEDIUMVIOLETRED	#C71585
MIDNIGHTBLUE	#191970	MINTCREAM	#F5FFFA
MISTYROSE	#FFE4E1	MOCCASIN	#FFE4B5
NAVAJOWHITE	#FFDEAD	NAVY	#000080
OLDLACE	#FDF5E6	OLIVE	#808000
OLIVEDRAB	#6B8E23	ORANGE	#FFA500
ORANGERED	#FF4500	ORCHID	#DA70D6
PALEGOLDENROD	#EEE8AA	PALEGREEN	#98FB98
PALETURQUOISE	#AFEEEE	PALEVIOLETRED	#DB7093
PAPAYAWHIP	#FFEFD5	PEACHPUFF	#FFDAB9
PERU	#CD853F	PINK	#FFC0CB
PLUM	#DDA0DD	POWDERBLUE	#B0E0E6
PURPLE	#800080	RED	#FF0000
ROSYBROWN	#BC8F8F	ROYALBLUE	#4169E1
SADDLEBROWN	#8B4513	SALMON	#FA8072
SANDYBROWN	#F4A460	SEAGREEN	#2E8B57
SEASHELL	#FFF5EE	SIENNA	#A0522D
SILVER	#C0C0C0	SKYBLUE	#87CEEB
SLATEBLUE	#6A5ACD	SLATEGRAY	#708090
SNOW	#FFFAFA	SPRINGGREEN	#00FF7F
STEELBLUE	#4682B4	TAN	#D2B48C
TEAL	#008080	THISTLE	#D8BFD8
TOMATO	#FF6347	TURQUOISE	#40E0D0
VIOLET	#EE82EE	WHEAT	#F5DEB3
WHITE	#FFFFFF	WHITESMOKE	#F5F5F5
YELLOW	#FFFF00	YELLOWGREEN	#9ACD32

References

1.	[AaBr-09]	Al Anderson, Ryan Benedetti,	Head First Networking, O'Reilly Media, Inc., 2009, ISBN 978-0-596-52155-4
2.	[AvDg03]	Vasile Avram, Gheorghe Dodescu	Informatics: Computer Hardware and Programming in Visual Basic, Ed. Economică, Bucureşti, 2003, 2007 (Chp. 1.6, 1.7, 1.8, 7.11.3 and 7.11.4)
3.	[Av-01]	Vasile Avram	Sisteme de Calcul si Operare/Computer Systems and Operating Systems, , Editura Dacia Europa Nova, vol. 1 - 2000, vol. 2 - 2001
4.	[AvDg-97]	Vasile Avram, Gheorghe Dodescu	General Informatics, by Vasile Avram and Gheorghe Dodescu, Editura Economica, 1997
5.	[BIS-TDM]	Dave Chaffey, Paul Bocij, Andrew Greasley, Simon Hickie	Business Information Systems-Technology, Development and Management for the e-business, Prentice Hall, London, second edition, 2003
6.	[BF01]	Benjamin Faraggi	Architectures marchandes et portails B to B, Ed. DUNOD, Paris, 2001
7.	[CheRo]	Chesbrough, Henry and Rosenbloom, Richard	The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies. Industrial and Corporate Change, 11, no. 3 (June 2002), 529 – 555.
8.	[DgAv05]	Gheorghe Dodescu, Vasile Avram	Informatics: Operating Systems and Application Software, Ed. Economică, Bucureşti, 2005 (Chp. 10.1, 10.2 and 10.3)
9.	[DMVA]	Daniel A. Menascé, Virgilio A. F. Almeida	Scaling for E-Business Prentice Hall, 2000

10.	[DOS_03]	Daconta, Michael C., Leo J. Obrst, and Kevin T. Smith.	The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. John Wiley & Sons. © 2003. Books24x7. <http://common.books24x7.com /book/id_6073
11.	[HKSU]	Chris Hart, John Kauffman, David Sussman, Chris Ullman	Beginning ASP.NET 2.0 with C#, Wrox Press 2006
12.	[JLMT]	Jerri Ledford, Mary E. Tyler	Google™ Analytics 2.0, John Wiley & Sons, August 27, 2007, ISBN-13: 978-0-47017501-9
13.	[KLJL]	Kenneth C. Laudon, Jane P. Laudon	Essentials of Management Information Systems – Managing the Digital Firm, Prentice Hall, fifth edition, 2003
14.	[LCT]	Lijun Mei, W.K. Chan, T.H. Tse	A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues, 2008 IEEE Asia-Pacific Services Computing Conference 978-0-7695-3473-2/08 $25.00 © 2008 IEEE DOI 10.1109/APSCC.2008.168
15.	[Lowe-05]	Doug Lowe	Networking All-in-One Desk Reference for Dummies, 2nd Edition, John Wiley & Sons © 2005 (http://www.acm.com)
16.	[MNSS]	Todd Miller, Matthew L. Nelson, Stella Ying Shen and Michael J. Shaw	e-Business Management Models: A Services Perspective and Case Studies, Revere Group
17.	[OLS07]	Phillip Olson et al	PHP manual, http://www.php.net/docs.php, 2007
18.	[OPT]	A. Ostenwalder, Y. Pigneur, and C.L. Tucci	Communications of AIS, Volume 15, Article 4 Clarifying Business Models: Origins, Present, and Future of the Concept by http://www.businessmodeldesign.com/ publications/ Preprint Clarifying Business Models Origins, Present, and Future of the Concept.pdf

19.	[PG-07]	Petter Gottschalk	Business Dynamics in Information Technologyby IGI Publishing © 2007
20.	[RFC 1630]	T. Berners-Lee	RFC 1630 - Universal Resource Identifiers in WWW, Network Working Group, CERN, June 1994
21.	[RFC3986]	T. Berners-Lee W3C/MIT, R. Fielding Day Software, L. Masinter Adobe Systems	Uniform Resource Identifier (URI): Generic Syntax, January 2005
22.	[RRSD]	Robert Reinhardt, Snow Dowd	Macromedia Flash 8 Bible, John Wiley & Sons © 2006
23.	[SS05]	Steve Schafer	Web Standards Programmer's Reference: HTML, CSS, JavaScript, Perl, Python, and PHP Wrox Press © 2005
24.	[Tan-02]	Andrew S. Tanenbaum	Computer Networks, Fourth Edition, Prentice Hall PTR, 2002
25.	[WV-01]	Weill P., Vitale M. R.	Place to space: Migrating to e-business models, Harvard Business School Press, Boston, 2001
26.	[W3C]	www.w3c.org	World Wide Web Consortium, Web standards collection
27.	[MSDN]	Microsoft Press	Microsoft Developer Network - VBScript
28.	[MSTcN]	Microsoft TechNet	Introduction to Windows Peer-to-Peer Networking, www.microsoft.com
29.	[W3C] World Wide Web Consortium, Web standards collection, www.w3c.org		
30.	[IEEE-802.11] IEEE Standard for Information technology Telecommunications and information exchange between systems, Local and metropolitan area networks, Specific requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-2007 (Revision ofIEEE Std 802.11-1999), 12 June 2007, http://standards.ieee.org/getieee802/download/802.11-2007.pdf		
31.	[IEEE] RFCs – Request For Comment – protocol standards, www.ieee.org		
32.	[Google] http://www.google.com/analytics Google Analitics		
33.	[BROCADE] Communication, www.brocade.com		
34.	[IBM] Hardware and software, www.ibm.com		

35.	[IBM-09] Seeding the Clouds: Key Infrastructure Elements for Cloud Computing, www.ibm.com
36.	[INTEL] Microprocessors, www.intel.com
37.	[HP] Hardware, www.hp.com
38.	[CISCO] Communication devices, www.cisco.com
39.	[SUN] Hardware and software, www.sun.com
40.	[SSL] SSL specification, http://www.openssl.org , www.modssl.org
41.	E-commerce business models http://www.iusmentis.com http://www.iusmentis.com/business/ecommerce/businessmodels/
42.	http://digitalenterprise.org/models/models.html Professor Michael Rappa, North Carolina State University
43.	http://reference.sitepoint.com/css/css SitePoint CSS reference
44.	[W3schools] Tutorials for HTML, XHTML, XML, CSS, JavaScript, VBScript, Ajax, http://www.w3schools.com
45.	[avrams.ro] Internet Technologies for Business, Lecture Notes, Handouts, and Examples. The site pages are implementations of different topics, http://www.avrams.ro

ISBN 978-606-505-274-1

9786065052741

Preţ 24,40 lei