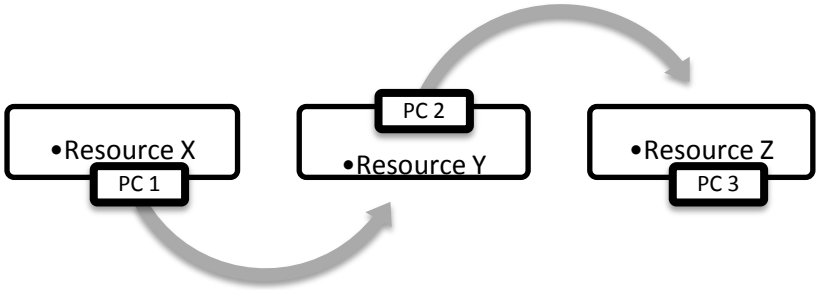
Unit 1

Introduction to Computer Network

Introduction

- Network is the concept of sharing resources and services
- Network of computer is a group of interconnected system sharing resources using a common communication link.
- Shared resource can be data printer, fax modem services such as email and database.
- To complete a network following are main thing which will be required
  1. Resource to share
  2. Pathway to transfer data (Medium)
  3. A set of rules governing the network (Protocol)



Reason of Network

- Sharing files.
- Sharing printer and other devices.
- Enabling centralized administration and security of resources.
- Supporting network application such as electronic mail and database.

Types of Network

- There are three main primary categories of network: Local Area Network, Metropolitan Area Network and Wide Area Network.
- Into which category a network falls is determined by its size, ownership, distance it covers and its physical architecture.

Local Area Network

- LAN is usually privately owned and links the devices in a single office, building or campus.
- Depending on the need of an organization and the types of technology used, LAN can be even of two PCs and Printer in some office or home.
- LANs are designed to allow resource to be shared between personal computers or workstation.
- One of computer may be given a large capacity disk drive and may become a server to the other clients. Software can be stored on this central server and used as needed by whole group.
- In addition to size LAN are distinguished from other types of networks by their transmission media and topology. Generally LAN uses only one type of transmission medium and common topology star.

- Traditionally LAN have data rates in 4 to 16 megabits per second(Mbps), however today's LAN have speed up to 100Mbps to 1Gbps (Gigabits per seconds).

Metropolitan Area Network

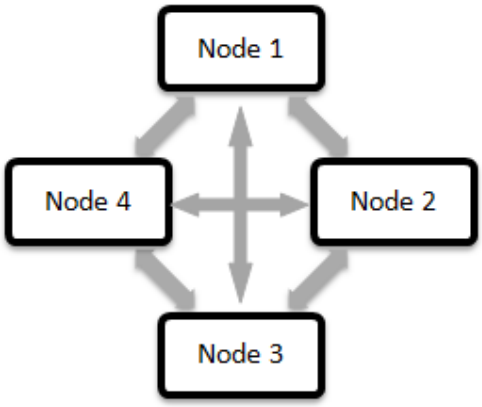
- MAN is designed to extend over an entire city. It may be single network such as cable television network or it may be means of connecting a number of LANs into a larger network so that resource may be shared to LAN to LAN.
- MAN may be wholly owned and operated by a private company, or it may be service provided by a public company such as local telephone company.

Wide Area Network

- It provides long distance transmission of data, voice, image and video information over large geographic areas that may comprise a country, continent or even whole world
- It contrasts to LANs which depends on their own hardware for transmission.
- WAN may utilize public, leased or private communication equipment usually in combination and therefore span an unlimited number of miles
- WAN that is wholly owned and used by a single company is often referred to as enterprise network.

Topologies

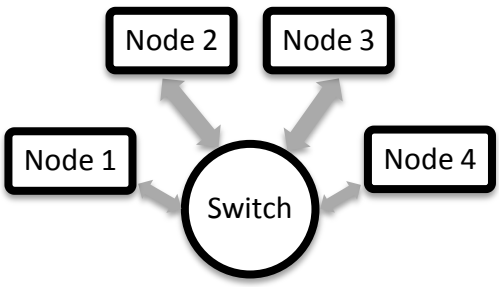
Mesh



- In the mesh topology every device has a dedicated point to point link to every other device.
- The term dedicated means that the link carries the traffic only between the two devices.
- The full connected mesh network always has n (n-1)/2 a physical channel to link every other device.
- The advantage of mesh topology can be as follows
  - The use of dedicated links generated that each connection can carry their own data so no traffic problem will exist.
  - It is the robust network if one link fails it does not harm a entire network.
  - Another advantage is privacy and security as the message travel in the dedicated lines. It prevents the access of data from other users.
- The disadvantages of mesh topology are as under
  - High amount of cabling is required than any other topology.
  - As more cabling is required the installation of cable is not easy to make fit in current shell.
  - More hardware is also required as each machine is having n-1 input output ports.
  - Upgrading the network is very hard as add one more device means more n-1 cables and I/O ports will be needed.

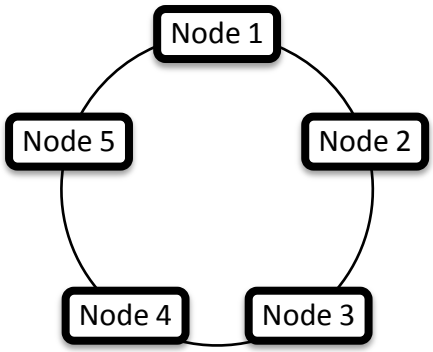
Star

- Each device has dedicated point to point link only to the central controller generally known as switch.



- As devices are not directly connected to one another, the exchange of message or data would be made through a central device switch.
- Sender will pass message or data to central device, which will then relay that message to another connected device i.e. receiver.
- The advantage of star topology are as follows
  - Less expensive comparing to the mess topology as less cabling and only one input output port is required.
  - It is also robust as the mesh topology; if one link fails then all other links will remain active because the internal design of switch is same as mesh topology.
  - It is also helpful in identification of the fault as long as switch is working it can also be used for monitoring the link.
  - The network can also be easily upgraded by adding extra switch in the network and inter connecting them.

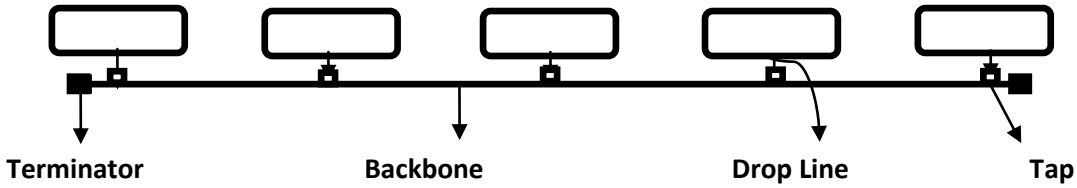
Ring



- A dedicated point to point connection is made only with neighbor device on each side of it.
- A signal, message or data will move in only one direction from device to device until it reaches to its destination.
- Each device will regenerate signal (repeater) and pass it to another device.
- The advantage of ring topology are as follows
  - Fault isolation is simplified as signal is circulating at all time if one device does not receive the signal within the specify period of time.
- The disadvantage of ring topology are as follows
  - The break in the ring, such as any system is down can disable whole network but this can be solved using dual ring.

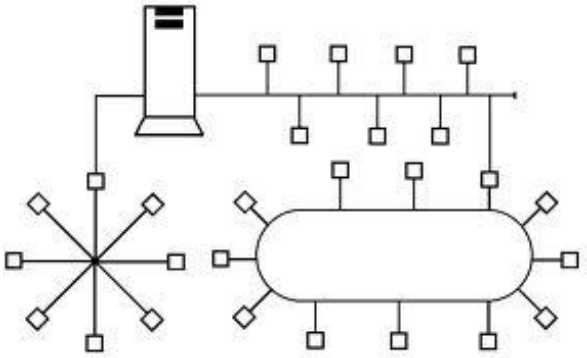
Bus

- One cable act as back bone to all devices in the network.
- Every device is connected to the bus cable by drop lines and tapes.
- The drop lines are connection running between device and main cable.
- Tape is the connection that either slice into the main cable or puncher into the main cable to connect with.
- A special connector called a terminator must be place at the end of backbone cable to prevent signal from reflecting back and causing the interference.
- The advantage of bus topology are as follows
  - The topology is having easy installation, comparing to star and mesh topologies.
  - It also required less able compare to star and mesh as only one backbone main cable much be stretched through the entire network.
- The disadvantage of bus topology are as follows
  - It is very hard to reconnection of any device or solving fault.
  - To add any new device will bring down whole network for that time period.
  - Any fault in main cable will bring down whole network.



Hybrid

- Hybrid networks use a combination of any two or more topologies in such a way that the resulting network does not exhibit one of the standard topologies (e.g., bus, star, ring, etc.).
- A hybrid topology is always produced when two different basic network topologies are connected. Two common examples for Hybrid network are: star-ring network and star bus network.
- A Star-Ring network consists of two or more star topologies connected using a multi station access unit (MAU) as a centralized hub.
- A Star-Bus network consists of two or more star topologies connected using a bus trunk (the bus trunk serves as the network's backbone).



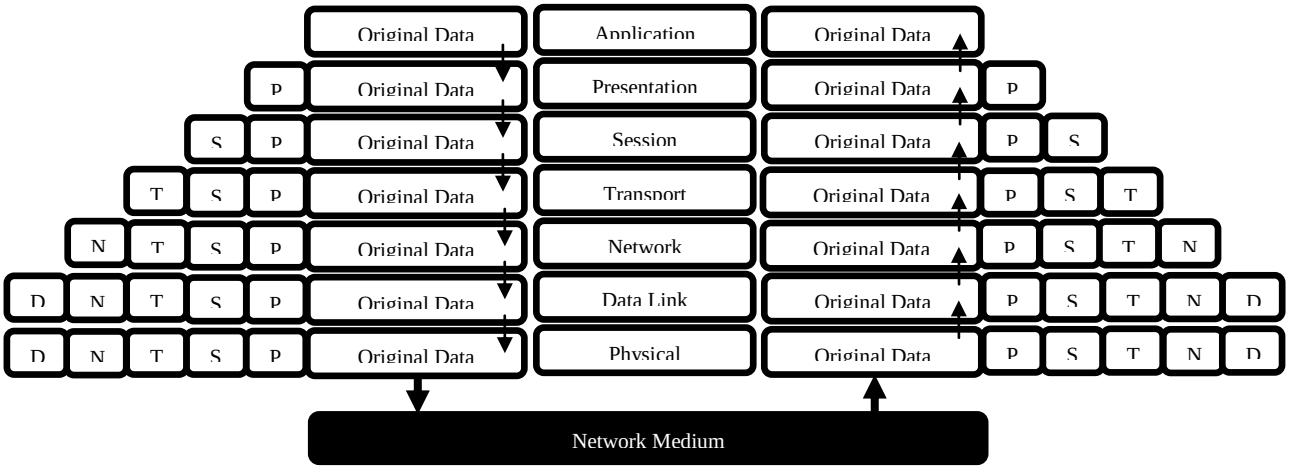
- Advantages of Hybrid Network Topology
  - Reliable: Unlike other networks, fault detection and troubleshooting is easy in this type of topology. The part in which fault is detected can be isolated from the rest of network and required corrective measures can be taken, WITHOUT affecting the functioning of rest of the network.
  - Scalable: It's easy to increase the size of network by adding new components, without disturbing existing architecture.
  - Flexible: Hybrid Network can be designed according to the requirements of the organization and by optimizing the available resources. Special care can be given to nodes where traffic is high as well as where chances of fault are high.
  - Effective: Hybrid topology is the combination of two or more topologies, so we can design it in such a way that strengths of constituent topologies are maximized while there weaknesses are neutralized. For example we saw Ring Topology has good data reliability (achieved by use of tokens) and Star topology has high tolerance capability (as each node is not directly connected to other but through central device), so these two can be used effectively in hybrid star-ring topology.
- Disadvantages of Hybrid Topology
  - Complexity of Design: One of the biggest drawbacks of hybrid topology is its design. It's not easy to design this type of architecture and it's a tough job for designers. Configuration and installation process needs to be very efficient.
  - Costly Hub: The hubs used to connect two distinct networks, are very expensive. These hubs are different from usual hubs as they need to be intelligent enough

to work with different architectures and should be function even if a part of network is down.

- Costly Infrastructure: As hybrid architectures are usually larger in scale, they require a lot of cables; cooling systems, sophisticate network devices, etc.

OSI Reference Model

- The most commonly used model is open system inter connection. The OSI model released in 1984 by international standard organization (ISO)
- OSI model is the blue print for vendors while developing a protocol.
- OSI model having different 7 layers, each having their special work to complete the process of communication.
- Layer 1 the physical layer or hardware layer consist of protocols that control communication of network media.
- Layer 7 the application layer interface the network services with the application of the computer. The service can be file service, printing service, etc.
- The other layers data link layer, network layer, transportation layer, session layer and presentation layer.
- When information is passed with the OSI model on a computer, each protocol layer add its own information to message which is being send.
- The information takes the form of header added to beginning of original message. The sending of a message always goes down in OSI model, hence the header are added top to bottom.
- When the message is received by destination computer every layer's information will be removed one after another.
- The header are removed in reverse order i.e. header which is added last will be removed first, and first added header will be removed last.
- The information of layer pass vertically, but the information between the computer communicate horizontal because each layer of sender computer will talks to it respective layer of receiving computer.
- Note should be made that the physical layer does not add any header information because the layer deals with providing a transmission routes between the computers.



Physical Layer

- OSI physical layer does not define the media which should be used. The layer is concern with all aspect of transmitting or receiving data or network media
- The layer is not responsible for saying whether a cable should be made a silver, copper or gold.
- A layer is concern with transmitting and receiving bits.
- This layer defines several key characteristics of the physical network, including the Following
  - Physical structure of the network (physical topology)
  - Mechanical and electrical specifications for using the medium (not the medium itself)
  - Bit transmission encoding and timing

Data Link Layer

- In the network communication the main thing involved is transferring the bits from one machine to another.
- Dozens of steps must be performing before transferring message to one device to another.
- The real message is not consist of a single bit but group of bits known as frame received from upper layer.
- It is the responsibility of data link layer to disassemble the frames into the bits from sender side and reassemble bits into frame at receiver side.
- The other function of data link layer is addressing, error control and flow control for a single link between the network devices.

1. The Data Link layer into two sub layers:
   - Media Access Control (MAC). The MAC sub layer controls the means by which multiple devices share the same media channel. This includes contention methods and other media access details. The MAC layer also provides addressing information for communication between network devices.
   - Logical Link Control (LLC). The LLC sub layer establishes and maintains links between communicating devices.

Network layer

- The data link layer deals with communication between the devices on the same network.
- The network layer handle the communication between devices, which are logically on the different network but are connected as the inter network.
- As the inter network can be off large size and may be constructed with the different types of network. The network layer will guide the packet by using the routing algorithm after reading their destination address.

Transportation Layer

- The layer ensures the reliable delivery of message to the destination device.
- The term reliable does not mean that the error never occurs instead it means that if error occurs they are detected.
- If error such as lost of data are detected the transformation layer either request the retransmission or notifies upper layer protocol so that they can take corrective action

- The main function of the transportation layer is to break large message in to segments suitable for network delivery.

Activities of transportation layer

- Repacking: When the large message divided into segment for the transportation, the transportation layer must be package the segments when they are receive before reassembling the original message.
- Error Control: When segments are lost during the transmission or when segment have duplicate ID the transportation layer initiate error recovery. The transportation layer also detects corrupted segments by managing end to end error control technique.
- End to End flow control: The transportation layer manage end to end flow control by using acknowledgment between two connected devices. If no acknowledgement receive or negative acknowledgment is detected the transportation layer can request the retransmission of most recent segment.

Session Layer

- The layer manage dialog between two computers by establishing, managing and terminating communication.
- There are three types of communication which are as follows.

Types of Session

- Simple dialog: It is responsible for one way data transfer only, for example a fire alarm which send a alarm message to the fire station but cannot receive any message from fire station
- Half Duplex dialog: This dialog handles two way transmissions but only one way at a time. When one device completes transmission then only another can transmit the message.
- Full Duplex dialog: It permits two way transmissions by providing a separate communication channel to the both side. The sending and receiving of the message can be done at parallel.

Presentation Layer

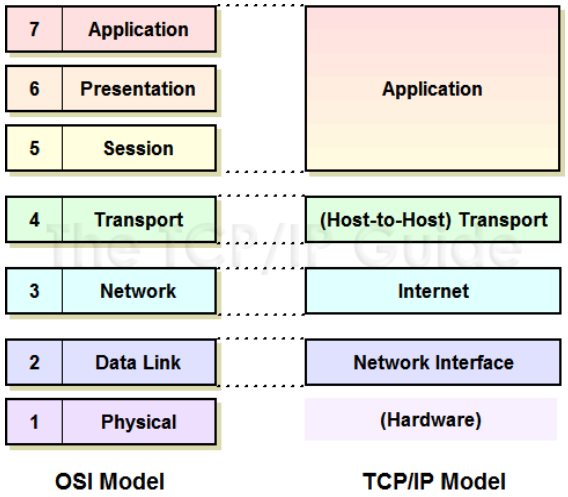
- The presentation layer deals with the syntax, or grammatical rules, needed for communication between two computers.
- The presentation layer converts system specific data from the application layer into a common, machine independent format that will support a more standardized design for lower protocol layers.
- On the receiving end, the presentation layer converts the machine independent data from the network into the format required for the local system.
- The conversion could use following
  - Bit order translation
  - Byte order translation
  - Character code translation
  - File syntax translation

Application Layer

- The application layer of the OSI reference model is concerned with providing services on the network, including file services, printing services, email services, database service, etc
- Common misunderstanding is that the application layer is responsible for running user application such as word processors.
- The application layer provides an interface whereby applications can communicate with the network.
- The application layer also advertises the available services to network

TCP/IP Model

- The Internet Protocol Suite, TCP/IP, is a suite of protocols used for communication over the internet. The TCP/IP model was created after the OSI 7 layer model for two major reasons.
- First, the foundation of the Internet was built using the TCP/IP suite and through the spread of the World Wide Web and Internet, TCP/IP has been preferred.
- Second, a project researched by the Department of Defense (DOD) consisted of creating the TCP/IP protocols.
- The DOD's goal was to bring international standards which could not be met by the OSI model. Since the DOD was the largest software consumer and they preferred the TCP/IP suite, most vendors used this model rather than the OSI.



- The TCP/IP model, similar to the OSI model, is comprised of layers. The OSI has seven layers and the TCP/IP model has four or five layers depending on different preferences.
- TCP/IP stands for Transmission Control Protocol/Internet Protocol
- TCP/IP defines how electronic devices (like computer) should be connected to internet and how data should be transmitted between them.
- TCP is known as fixed connection, communication between application
- If one application wants to communicate with another, its sends a communication request. TCP will set up 'full-duplex' communication between two application
- The 'full-duplex' communication will occupy separate communication line between the two computer until it is closed by one of the two application
- IP is connection less, communication between computers
- IP does not occupy the communication line between two computers, each line can be used by multiple computers at the same time
- With the help of IP message divide into small independents 'packets' and sent between computers via internet.
- IP is responsible for routing each packets to the correct destination
- TCP/IP is a large collection of different communication protocols based upon the two original protocols TCP and IP are having following other protocols

Application

- This layer is comparable to the application, presentation, and session layers of the OSI model all combined into one.
- It provides a way for applications to have access to networked services. This layer also contains the high level protocols. The main issue with this layer is the ability to use both TCP and UDP protocols.
- For example TFTP uses UDP because usually on a LAN the physical links are short enough to ensure quick and reliable packet delivery without many errors.
- SMTP instead uses TCP because of the error checking capabilities. Since we consider our email important information we would like to ensure a safe delivery.

Transport

- This layer acts as the delivery service used by the application layer. Again the two protocols used are TCP and UDP.
- The choice is made based on the application's transmission reliability requirements. The transport layer also handles all error detection and recovery.
- It uses checksums, acknowledgements, and timeouts to control transmissions and end to end verification. Unlike the OSI model, TCP/IP treats reliability as an end-to-end problem.

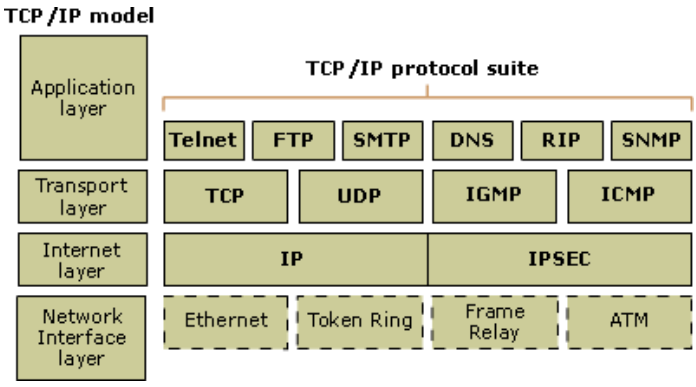
Internet

- The routing and delivery of data is the responsibility of this layer and is the key component of this architecture.
- It allows communication across networks of the same and different types and carries out translations to deal with dissimilar data addressing schemes. It injects packets into any network and delivers them to the destination independently to one another.
- Because the path through the network is not predetermined, the packets may be received out of order.
- The upper layers are responsible for the reordering of the data. This layer can be compared to the network layer of the OSI model. IP and ARP6 are the major protocols used at this layer.

Network access

- This is a combination of the Data Link and Physical layers of the OSI model which consists of the actual hardware.
- This includes wires, network interface cards, etc. Other related details within this layer are connectors, signal strength, and wavelength along with various others. It will use the required LAN operating algorithms, such as Carrier Sense Multiple Access with Collision Detect (CMSA/CD) or Token Passing etc. and is responsible for placing the data within a frame.
- The frame format is dependent on the system being used, for example Ethernet LAN, Frame relay (packet switching protocol for connecting devices on a Wide Area Network.), etc. The frame is the package that holds the data, in the same way as an envelope holds a letter.

- The frame holds the hardware address of the host and checking algorithms for data integrity. This layer has actually not been specified in details because it depends on which technology is being used such as Ethernet. So freedom is given to this layer as far as implementation is concerned.



TCP/IP Protocol suite

1. TCP (Transmission Control Protocol)
   - TCP is used for transmission of data from an application to network
   - TCP is responsible for breaking data down into IP packets before they are sent and for assembling the packets when they arrive
2. IP (Internet Protocol)
   - IP takes care of the communication with other computers
   - IP is responsible for sending and receiving data packets over internal
3. HTTP (Hyper Text Transfer Protocol)
   - HTTP takes care of the communication between a web server and web browser
   - HTTP is used for sending requests from web – client (browser) to web – server, returning web content (web page) from the server back to clients
4. HTTPS (Secure HTTP)
   - HTTPs takes care of secure communication between web server and web browser
   - HTTPs typically handles credit card transaction and other sensitive data
5. SSL (Secure Sockets Layer)
   - SSL protocols is used for encryption of data for secure data transmission
6. SMTP (Simple Mail Transfer Protocol)
   - SMTP is used for transmission email
7. MIME (Multi Purpose Internet Mail Extension)
   - MIME protocol lets SMTP transmits multimedia files including voice, audio, and binary data across TCP/IP network
8. IMAP (Internet Message Access Protocol)
   - IMAP is used for storing and retrieving emails
9. POP (Post Office Protocol)
   - POP is used for downloading emails from an email server to personal computers
10. FTP (File Transfer Protocol)
    - FTP take care of transmission of files between computers
11. NTP (Network Time Protocol)
    - NTP is used to synchronize time of clock between computers
12. DHCP (Dynamic Host Configuration Protocol)
    - DHCP is used for allocation of dynamic IP address to computers in a network
13. SNMP (Simple Network Management Protocol)
    - SNMP is used for administration of computers networks
14. LDAP (Lightweight Directory Access Protocol)
    - LDAP is used for collection information about users and email address from internet
15. ICMP (Internet Control Message Protocol)

- ICMP handles error and care of error in network

16. ARP (Address Resolution Protocol)
    - ARP is used by IP to find the hardware address of computer network card based on IP address
17. RARP (Reverse Address Resolution Protocol)
    - RARP is used by IP to find IP address based on hardware address of computer network card.
18. BOOTP (Boot Protocol)
    - BOOTP is used for booting (starting) computers from network
19. PPTP (Point to Point Tunneling Protocol)
    - PPTP is used for setting up a connection (tunnel) between private networks.

Internet

Internet is the largest and most complete learning tool in the world. Through the Internet we can find knowledge resources that allow you to study virtually (practically) and discipline imaginable. Not only that but you can communicate quickly and effectively with others who are also interested in the same discipline. Teachers, student and other educators can share ideas instantly across vast distances. A variety of programs have been installed on the Internet to use the services.

Individuals companies and institutions use the Internet in many ways. Businesses use it to provide access to complex database such as financial databases. Companies can carry out commerce online including advertising selling buying distributing products and providing after sales services. Business and institution can use the internet for voice and video conferencing and other forms of communication that allow people to telecommute or work from a distance.

The term Internet is comprised of two different term ‘interconnection’ and ‘network’. A network is simply a group of computer that is connected together for sharing information and resources. In other terminology it is also known as ‘network of networks’. But the Internet has become something greater than the sum of its connection. Certainly no one could have predicted how its move from a military computer network (ARPANET) to a public and commercial network (today’s network) would affect the way we think about information and communication. In fact from the last few years the Internet has become one of the most discussed computer technologies. Every day we hear about its content, its future and how we have to get access to it. If it were to continue growing at the current rate, the entire population of the world would be on the Internet in fourth coming years.

Brief history of the internet

During the 1960 the United States Department of Defense saw a need to maintain some kind of communication network. A U.S military think, proposed a decentralized communications network as a solution. This network would use a series of interconnected nodes each with the ability to pass and receive messages. These messages would in turn be divided into smaller chunks called packets. Each packed would be given a specific address with specified source and destination nodes, and would travel through the network on an individual basis. A packet is a series of bytes that are sent in an order set by a particular format. In 1969, the first “Interface Message Processor” a predecessor to today’s routers, was installed at the University of California in Los Angeles (UCLA) and ARPANET came into being. ARPANET included some of the

services that are important features of the present day Internet such as File transfer protocol (FTP), remote login (TELNET) and electronic mail.

ISP

Short for Internet Service Provider, it refers to a company that provides Internet services, including personal and business access to the Internet. For a monthly fee, the service provider usually provides a software package, username, password and access phone number. Equipped with a modem, you can then log on to the Internet and browse the World Wide Web and USENET, and send and receive e-mail.

An ISP is also sometimes referred to as an IAP (Internet Access Provider). As more people move to broadband the traditional dial-up ISP and all of its services is being replaced. Today, if you are using broadband, you have a broadband provider such as Comcast and are probably using other online services to provide other services. For example, using Gmail as your e-mail provider.

ISP History

The first ISP is widely believed to be Telenet, which was the first commercial version of ARPANET introduced in 1974. The first ISP for the Internet we know and use today is considered to be "The World" who started serving customers in 1989.

- Intranet is system in which multiple PCs are connected to each other.
- PCs in intranet are not available to the world outside the intranet.
- Usually each company or organization has their own Intranet network and members/employees of that company can access the computers in their intranet.
- Each computer in Intranet is also identified by an IP Address which is unique among the computers in that Intranet.

Similarities in Internet and Intranet

- Intranet uses the internet protocols such as TCP/IP and FTP.
- Intranet sites are accessible via web browser in similar way as websites in internet. But only members of Intranet network can access intranet hosted sites.
- In Intranet, own instant messengers can be used as similar to yahoo messenger/ gtalk over the internet.

Differences in Internet and Intranet

- Internet is general to PCs all over the world whereas Intranet is specific to few PCs.
- Internet has wider access and provides a better access to websites to large population whereas Intranet is restricted.
- Internet is not as safe as Intranet as Intranet can be safely privatized as per the need.

VSAT - Very Small Aperture Terminal

A Very Small Aperture Terminal (VSAT) is a device - known as a small private earth station - hat is used to transmit & receive data signal through a satellite. The "very small" component of the VSAT acronym refers to the size of the VSAT dish antenna -typically about 4 feet (1.2 m) diameter- that is mounted on a roof, or placed on the ground.

This antenna, along with the attached low-noise converter (LNB-which receives satellite signals) and the transmitter (BUC-which transmits the radio waves signals) make up the VSAT outdoor unit (ODU), one of the two components of a VSAT earth station. The satellite modem which makes the connection between the computers is named the indoor unit (IDU) and does all management of that small private low cost earth station.

Portal

"Portal" has in the recent two years become an increasingly popular term being mentioned and discussed in the IT sector and many organizations.- would give some idea about what a portal is. In short, it is a web system that provides the functions and features to authenticate and identify the users and provide them with an easy, intuitive, personalized and user-customizable web-interface for facilitating access to information and services that are of primary relevance and interests to the users. To the organization that sets up the portal, it is a system that provides versatile functions for the organization to catalogue or organize collections of different and multiple sources of information and service resources for dissemination to many users according to their specific privileges, needs and interest. Hence, the main purpose of setting up a portal is to bring the vast information and service resources available from many sources to many users in an effective manner.

"Many to many in an effective manner" should be the essence of a portal system. As a matter of fact, long before the concept of portal was explored by IT people in trying to reap the potential values and benefits that could be provided by using Web technologies, the term "Portal system" has been consistently used in the field of medical and anatomy studies to represent the important system within our body that very effectively brings blood from the many capillaries connecting to our digestive system to the many fine blood vessels inside our liver.
A portal should support the following desirable functions and features:

- Allowing different information- and service-providing departments to set up and update their own information and services tailored specifically for different user groups according to the common user profiles (such as grades, departments associated, etc.) and the specific needs of these user groups at specific times.
- Presenting automatically the information and services that a user would need according to his profile at the appropriate time.
- Allowing a user to select the information and services that are his interests and to customize their presentation.
- Setting up information and services from users' perspective rather than from the angle of convenience of the services providers.
- Supporting the "Single-sign-on" feature so that a single sign-on step would enable the user to gain access to the different information resource and services that are supported by different application systems provided by different departments.
- Technically, this feature can be facilitated by means of implementing a common organization-wide LDAP (Light-weight Directory Access Protocol) service and CAS (Central Authentication Service).

DNS(Domain Name Server)

- A DNS server translates a computer or domain name to the associated IP address. It provides a way to map friendly host names, or URLs, to IP addresses. A DNS server houses the IP addresses used to access internet resources.

- The internet operates via IP address in the format of 255.255.255.255 (i.e. 192.168.002.015). Since those are hard to remember, DNS servers are responsible for translating the user friendly names like www.faqfarm.com to 64.255.154.135.
- If you type "CMD" in the Run box on Windows XP, then "PING www.faqfarm.com". You can see what the IP address is for the website.
- DNS stands for Domain Name System. A DNS server resolves a name to an IP address, as stated in an earlier answer, but it can also point to multiple IP addresses for load balancing, or for backup servers if one or more is offline or not accepting connections.
- Individual organizations may have their own DNS servers for their local Intranet.

Unit 2
Application of internet

World Wide Web (Www)

Stands for "World Wide Web." It is important to know that this is not a synonym for the Internet. The World Wide Web, or just "the Web," as ordinary people call it, is a subset of the Internet. The Web consists of pages that can be accessed using a Web browser. The Internet is the actual network of networks where all the information resides. Things like Telnet, FTP, Internet gaming, Internet Relay Chat (IRC), and e-mail are all part of the Internet, but are not part of the World Wide Web. The Hyper-Text Transfer Protocol (HTTP) is the method used to transfer Web pages to your computer. With hypertext, a word or phrase can contain a link to another Web site. All Web pages are written in the hyper-text markup language (HTML), which works in conjunction with HTTP.

Web Search Engine

Internet search engines (e.g. Google, AltaVista) help users find web pages on a given subject. The search engines maintain databases of web sites and use programs (often referred to as "spiders" or "robots") to collect information, which is then indexed by the search engine.

Computer software designed to help users of the Internet locate information on the World Wide Web. It collects and indexes Internet resources (Web pages, Usenet Newsgroups, programs, images, etc.) and provides a keyword search system allowing the user to identify and retrieve resources. There are many search engines available and each is different in their scope, search protocols, and appearance. Some common search engines are: Alta Vista, Google, Yahoo, Excite, Lycos, and HotBot.

Web Meta Search Engine

A Meta Search Engine is a website where you can run a single search and it searches all the major search engine's databases on the web and then gives you the best results on a single page.

Remote Login

In simple words Remote Login means to access other computers on the network or on the other network by the use of telnet or rlogin command. In other words we can also say that Remote Login means to access native computer from the other computer on the network when you are connected to the internet.

Telnet

Telnet is an old computer protocol (set of programmatic rules). Telnet is famous for being the original Internet when the Net first launched in 1969. Telnet stands for 'telecommunications network', and was built to be form of remote control to manage mainframe computers from distant terminals. In those original days of large mainframe computers, telnet enabled research students and professors to 'log in' to the university mainframe from any terminal in the building. This remote login saved researchers hours of walking each semester. While telnet pales in comparison to modern networking technology, it was revolutionary in 1969, and telnet helped pave the way for the eventual World Wide Web in 1989. While telnet technology is very old, it is still in some use today by purists. Telnet has evolved into a new modern version of remote

control called 'SSH', something that many modern network administrators use today to manage Linux and Unix computers from a distance. Telnet is a text-based computer protocol. Unlike Firefox or Google Chrome screens, telnet screens are very dull to look at. Very different from Web pages that sport fancy images, animation, and hyperlinks, telnet is about typing on a keyboard. Telnet commands can be rather cryptic commands, with example commands being 'z' and 'prompt% fg'. Most modern users would find telnet screens to be very archaic and slow

Electronic Mail(Email)

ARPANET immediately become a forum for the exchange of information and ideas collaboration among scientists and educators was the number one use of the system and the main incentive for new sites to want to be connected. Thus it is not surprising that the first major application developed for use on the ARPANET was electronic mail. With the advent (invention) of Ray Tomlinson's email system in 1972, researches connected to the net could establish one-on-one communication links with colleagues all over the world and could exchange ideas and research at a pace never before imagined.

An email is a network service that enables users to send and receive messages. In other words, and email is an addressed message sent over computer network to one or more recipients. You can send or receive personal and business related message with attachments like pictures or other documents. It was good for only short message some time ago; you could not send attachments like formatted documents or graphics. With the advent of MIME (Multipurpose internet mail extension) you can send message with formatted documents, photos, sound file and video files as attachments.

E-Commerce and E-Business

Electronic commerce or e-commerce refers to a wide range of online business activities for products and services. It also pertains to "any form of business transaction in which the parties interact electronically rather than by physical exchanges or direct physical contact." E-commerce is usually associated with buying and selling over the Internet, or conducting any transaction involving the transfer of ownership or rights to use goods or services through a computer-mediated network. Though popular, this definition is not comprehensive enough to capture recent developments in this new and revolutionary business phenomenon. A more complete definition is: E-commerce is the use of electronic communications and digital information processing technology in business transactions to create, transform, and redefine relationships for value creation between or among organizations, and between organizations and individuals. The major different types of e-commerce are: business-to-business (B2B); business-to-consumer (B2C); business-to-government (B2G); consumer-to-consumer (C2C);

What is B2B E-Commerce?

B2B e-commerce is simply defined as e-commerce between companies. This is the type of e-commerce that deals with relationships between and among businesses. About 80% of e-commerce is of this type, and most experts predict that B2B e-commerce will continue to grow faster than the B2C segment

What is B2C e-commerce?

Business-to-consumer e-commerce, or commerce between companies and consumers, involves customers gathering information; purchasing physical goods (i.e., tangibles such as books or

consumer products) or information goods (or goods of electronic material or digitized content, such as software, or e-books); and, for information goods, receiving products over an electronic network.12

It is the second largest and the earliest form of e-commerce. Its origins can be traced to online retailing (or e-tailing).13 Thus, the more common B2C business models are the online retailing companies such as Amazon.com, Drugstore.com, Beyond.com, Barnes and Noble and ToysRus.

What is B2G E-Commerce?

Business-to-government e-commerce or B2G is generally defined as commerce between companies and the public sector. It refers to the use of the Internet for public procurement, licensing procedures, and other government-related operations. This kind of e-commerce has two features: first, the public sector assumes a pilot/leading role in establishing e-commerce; and second, it is assumed that the public sector has the greatest need for making its procurement system more effective. Web-based purchasing policies increase the transparency of the procurement process (and reduce the risk of irregularities). To date, however, the size of the B2G e-commerce market as a component of total e-commerce is insignificant, as government e-procurement systems remain undeveloped

What is C2C E-Commerce?

Consumer-to-consumer e-commerce or C2C is simply commerce between private individuals or consumers. This type of e-commerce is characterized by the growth of electronic marketplaces and online auctions, particularly in vertical industries where firms/businesses can bid for what they want from among multiple suppliers.16 It perhaps has the greatest potential for developing new markets

E-Business

E-business (electronic business), derived from such terms as "e-mail" and "e-commerce," is the conduct of business on the Internet, not only buying and selling but also servicing customers and collaborating with business partners. One of the first to use the term was IBM, when, in October, 1997, it launched a thematic campaign built around the term. Today, major corporations are rethinking their businesses in terms of the Internet and its new culture and capabilities. Companies are using the Web to buy parts and supplies from other companies, to collaborate on sales promotions, and to do joint research. Exploiting the convenience, availability, and world-wide reach of the Internet, many companies, such as Amazon.com, the book sellers, have already discovered how to use the Internet successfully.

Increasingly, much direct selling is taking place on the Internet of computer-related equipment and software. One of the first to report sales in the millions of dollars directly from the Web was Dell Computer. Travel bookings directly or indirectly as a result of Web research are becoming significant. Custom-orderable golf clubs and similar specialties are considered good prospects for the immediate future.

With the security built into today's browsers and with digital certificates now available for individuals and companies from Verisign, a certificate issuer, much of the early concern about the security of business transaction on the Web has abated and e-business by whatever name is accelerating.

IBM considers the development of intranets and extranets to be part of e-business. e-business can be said to include e-service, the provision of services and tasks over the Internet by application service providers (ASP).

E-Governance

E-government is the use of information and communications technology(ICT) promote more efficient and cost effective government, facilitate more convenient government services, allow greater public access to information, and make government more accountable to citizens. Therefore a common definition for e-government is a form of organization that integrates the interactions and the interrelations between government and citizens, companies, customers, and public intuitions through the application of modern information and communication technologies.

The three main target groups that can be distinguished in e-governance concepts are government, citizens and businesses. The most common interactions in e-governance are G2G, G2B, G2C, G2E.

- G2E - Interactions and services to public servants and people's representatives
- G2B - Interactions and services to business sectors
- G2C – Interactions and services to citizens
- G2G – Interactions between Government institutions, Governments and International institutions

Mobile Commerce:

M-commerce (mobile commerce) is the buying and selling of goods and services through wireless handheld devices such as cellular telephone and personal digital assistants (PDAs). Known as next-generation e-commerce, m-commerce enables users to access the Internet without needing to find a place to plug in. The emerging technology behind m-commerce, which is based on the Wireless Application Protocol (WAP), has made far greater strides in Europe, where mobile devices equipped with Web-ready micro-browsers are much more common than in the United States.

Website Basics

Web page

A web page is a document that contains text and graphical information that can be accessed through the Internet through a web browser. Usually, web pages are stored as HTML documents on a web server. Web pages are what make up the World Wide Web. These documents are written in HTML (hypertext markup language) and are translated by your Web browser. Web pages can either be static or dynamic. Static pages show the same content each time they are viewed. Dynamic pages have content that can change each time they are accessed. Some pages are typically written in scripting languages such as PHP, Perl, ASP, or JSP.

In web site terms, static means web pages that are not interactive. Because the web site visitor does not have any control over the information provided, the pages and information do not change with each visit. There is not a two-way communication between the user (client) and the web site (server) in a static page.

A web page that uses a database to generate data or information "on the fly", depending on the user's request. Dynamic web pages may use Active Server Pages (using VBScript or JScript) or other scripts. In contrast, static web page renders HTML exactly the way it is written. Although a static page may run a Java applet, the user may interact with the page only by selecting a link.

Hyper Text Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

The standards development of HTTP was coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs). The first definition of HTTP/1.1, the version of HTTP in common use, occurred in RFC 2068 in 1997, although this was obsolete by RFC 2616 in 1999.

A later version, the successor HTTP/2, was standardized in 2015, then supported by major web browsers and already supported by major web servers.

FTP (File transfer protocol)

FTP is the file transfer protocol service for the Internet. At its most basic level FTP simply routes data between two computers on the Internet in order to facilitate file transfer. On the higher end, FTP can create and remove directories, rename files, and perform most other file maintenance tasks.

A domain name is the part of your Internet address that comes after "www". For example, in facebook.com the domain name is facebook.com. A domain name becomes your Business Address so care should be taken to select a domain name. Your domain name should be easy to remember and easy to type.

Domain Name

When you plan to put a site online, this is one of the important steps to buy a domain name. This is always not necessary that whatever domain name you are looking that is available so in that case you will have to opt for any other good domain name.
When you buy a domain name it is registered and when domain names are registered they are added to a large domain name register, and information about your site − including your Internet IP address is stored on a DNS server and your contact information etc. is registered with your registrar.
You can buy domain name from any domain registrar like GoDaddy

Domain Extension Types

There are many types of domain extensions you can choose for your domain name. This depends on your business nature.
For example, if you are going to register a domain name for education purpose then you can choose .edu extension.

Below is a reference of the correct usage of certain extensions. But there is no hard and fast rule to go for any extension. Most commonly used is .com

- .com − Stands for company/commercial, but it can be used for any website.
- .net − Stands for network and is usually used for a network of sites.
- .org − Stands for organization and is supposed to be for non-profit bodies.
- .us, .in − They are based on your country names so that you can go for country specific domain extensions
- .biz − A newer extension on the Internet and can be used to indicate that this site is purely related to business.
- .info − Stands for information. This domain name extension can be very useful, and as a new comer it's doing well.
- .tv − Stands for Television and are more appropriate for TV channel sites.

Newer domain extensions such as .biz .info and .us etc. have more name choices available as many of the popular domains have yet to be taken and most of the them are available at very nominal prices.

Choosing a Domain Name

- The domain name will be your business address. Hence, it is imperative that you choose the domain name with utmost care.
- Many people think it is important to have keywords in a domain. Keywords in the domain name are usually important, but it usually can be done while keeping the domain name short, memorable, and free of hyphens.
- Using keywords in your domain name gives you a strong competitive advantage over your competitors. Having your keywords in your domain name can increase click through rates on search engine listings and paid ads as well as make it easier to using your keywords in get keyword rich descriptive inbound links.
- Avoid buying long and confusing domain names. May people separate the words in their domain names using dashes or hyphen; In the past the domain name itself was a significant ranking factor but now with advanced search engines, it is not a significant factor anymore.
- Keep two to three words in your domain name − it will be more memorable. Some of the most memorable websites do a great job of branding by creating their own words. Examples include eBay, Yahoo!, Expedia, Slashdot, Fark, Wikipedia, Google...
- You should be able to say it over the telephone once and the other person should know how to spell it and they should know what you sell. If you can do that AND work keywords in there, good for you. If you can't, skip the keywords.

Sub-Domains

You can divide your domain into many sub domains based on your requirement. If you are doing multiple businesses using the same domain, then it would be useful to have sub-domains for every business. Following are examples of some sub-domains −

You must have seen google.com as a main domain but Google has created many sub domains based on their business. Some of them are as follows −

- adwords.google.com − this sub domain is being used for Google Ad words.
- groups.google.com − this sub domain is being used for Google Groups.
- images.google.com − this sub domain is being used for Google Images.

This way, you can present your different business sections in a very good segregated way. It is not a big thing to create sub domains. If you already have registered a domain, then your registrar will provide you a way to create sub-domains. You may need to talk to your registrar for more detail.

URL

Uniform Resource Locator, the global address of documents and other resources on the World Wide Web. The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located.

The first part is called scheme. It describes the protocol that the client should use to access the resource. The protocol is usually followed by :// except in the case of the file which look likes file:///.

The second part is Host (domain name), it the name of Internet host on which the resource resides.

The third part is path. The path is the full path and possible the file name of a document. Although protocol and host are always required, the path is not. If you are opening an URL leaving off the path causes the web server to return the default home page for the site. e.g. http://www.jjkcc.org/index.php

scheme://prefix.domain:port/path/filename

- scheme - defines the type of Internet service (most common is http)
- prefix - defines a domain prefix (default for http is www)
- domain - defines the Internet domain name (facebook.com)
- port - defines the port number at the host (default for http is 80)
- path - defines a path at the server (If omitted: the root directory of the site)
- filename - defines the name of a document or resource

Common URL Schemes

Scheme	Short for	Used for
http	HyperText Transfer Protocol	Common web pages. Not encrypted
https	Secure HyperText Transfer Protocol	Secure web pages. Encrypted
ftp	File Transfer Protocol	Downloading or uploading files
file		A file on your computer

Protocol Address:

An Internet Protocol address (IP address) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing. Its role has been characterized as follows: "A name indicates what we seek. An address indicates where it is. A route indicates how to get there."

The designers of the Internet Protocol defined an IP address as a 32-bit number and this system, known as Internet Protocol Version 4 (IPv4), is still in use today. However, because of the growth of the Internet and the predicted depletion of available addresses, a new version of IP (IPv6),

using 128 bits for the address, was developed in 1995.IPv6 was standardized as RFC 2460 in 1998, and its deployment has been ongoing since the mid-2000s.

IP addresses are usually written and displayed in human-readable notations, such as 172.16.254.1 (IPv4), and 2001:db8:0:1234:0:567:8:1 (IPv6).

The Internet Assigned Numbers Authority (IANA) manages the IP address space allocations globally and delegates five regional Internet registries (RIRs) to allocate IP address blocks to local Internet registries (Internet service providers) and other entities.

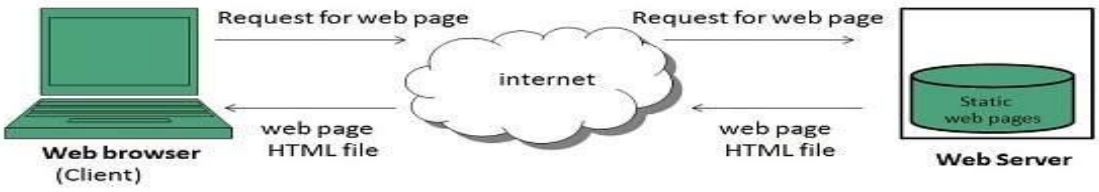
Web Site

A collection of files Web files, including a home page. The home page is usually the address that is given, as it is usually the easiest to remember, this is the first document user see when they enter the site. The site might also contain additional documents and files i.e. ww.greencomputer.com, in actuality the first page of the site is www.greencomputer.com/index.html, the server is told that www.greencomputer.com really means www.greencomputer.com/index.html making it easier to remember web addresses.

Static web site

Static websites are also known as flat or stationary websites. They are loaded on the client's browser as exactly they are stored on the web server. Such websites contain only static information. User can only read the information but can't do any modification or interact with the information.

Static websites are created using only HTML. Static websites are only used when the information is no more required to be modified.



Dynamic Websites

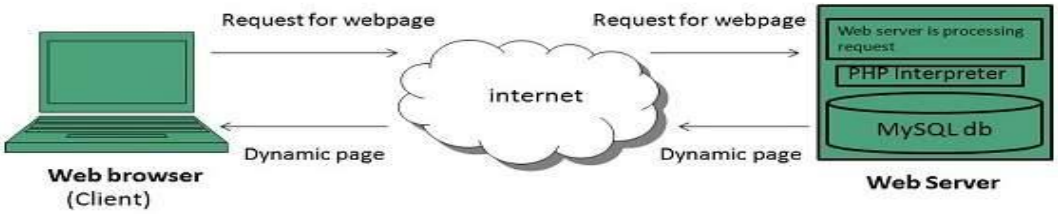
Dynamic websites shows different information at different point of time. It is possible to change a portion of a web page without loading the entire web page. It has been made possible using Ajax technology.

Server-side dynamic web page

It is created by using server-side scripting. There are server-side scripting parameters that determine how to assemble a new web page which also include setting up of more client-side processing.

Client-side dynamic web page

It is processed using client side scripting such as JavaScript. And then passed in to Document Object Model (DOM).



Responsive website:

A responsive design means a type of design where the characteristics of the website (such as width, alignment of data, etc) will get adjusted according to the width of the screen. This means that you are serving different webpage to users with a smart phone, a tablet or a laptop. This is a characteristic implemented on the code. This has got nothing to do with a specific design pattern.

Irrespective of how your website look, you can make a responsive design. This is rather a functional aspect than a design aspect, even though the functionality is achieved by the designer.

Responsive design is needed because of two reasons :

1. Usability : With the spike in the number of smartphone users, it gets more and more necessary for websites to provide content in such a way that it is easier for them to use.
2. Speed Matters :Google announced mobile page load time as a factor for SEO. So with a design specific for mobile devices, you can get a better page load times for mobile devices and hence better rankings.

Web browser

A browser is an application that provides a way to look at and interact with all the information on the World Wide Web. Technically, a web browser uses HTTP to make requests of web servers throughout the Internet on behalf of the browser user. A commercial version of the original browser; Mosaic, is in use.

A program used to view, download, upload, surf or otherwise access documents (pages) on the World Wide Web. Netscape Navigator and Microsoft Internet Explorer are examples of Web browsers. Simple Web browsers, such as Lynx, allow users to access the text only content

Web server

It is computer that has the high processing power and is connected to the Internet and also runs programs such as the Internet Information service. It stores, distributes and retrieves the file of the web site.

Web hosting

Web hosting is a service that allows organizations and individuals to post a website or web page on to the Internet. A web host, or web hosting service provider, is a business that provides the technologies and services needed for the website or webpage to be viewed in the Internet.

Websites are hosted, or stored, on special computers called servers.

When Internet users want to view your website, all they need to do is type your website address into their browser. Their computer will then connect to your server and your webpages will be delivered to them through the browser.

Most hosting companies require that you own your domain name in order to host with them. If you do not have a domain name, the hosting companies will help you purchase one.

Network Security Concepts

Cyber Law

In Simple way we can say that cyber crime is unlawful acts wherein the computer is either a tool or a target or both Cyber crimes can involve criminal activities that are traditional in nature, such as theft, fraud, forgery, defamation and mischief, all of which are subject to the Indian Penal Code. The abuse of computers has also given birth to a gamut of new age crimes that are addressed by the Information Technology Act, 2000.

We can categorize Cyber crimes in two ways

The Computer As A Target :- using a computer to attack other computers.
e.g. Hacking, Virus/Worm attacks, DOS attack etc.
The Computer As A Weapon :-using a computer to commit real world crimes.
e.g. Cyber Terrorism, IPR violations, Credit card frauds, EFT frauds, Pornography etc. Cyber Crime regulated by Cyber Laws or Internet Laws.

Technical Aspects

Technological advancements have created new possibilities for criminal activity, in particular the criminal misuse of information technologies such as:

Unauthorized Access & Hacking

Access means gaining entry into, instructing or communicating with the logical, arithmetical, or memory function resources of a computer, computer system or computer network.

Unauthorized access would therefore mean any kind of access without the permission of either the rightful owner or the person in charge of a computer, computer system or computer network.

Every act committed towards breaking into a computer and/or network is hacking. Hackers write or use ready-made computer programs to attack the target computer. They possess the desire to destruct and they get the kick out of such destruction. Some hackers hack for personal monetary gains, such as to stealing the credit card information, transferring money from various bank accounts to their own account followed by withdrawal of money. By hacking web server taking control on another persons website called as web hijacking

Trojan Attack :- The program that act like something useful but do the things that are quiet damping. The programs of this kind are called as Trojans. The name Trojan Horse is popular.
Trojans come in two parts, a Client part and a Server part. When the victim (unknowingly) runs the server on its machine, the attacker will then use the Client to connect to the Server and start using the trojan. TCP/IP protocol is the usual protocol type used for communications, but some functions of the trojans use the UDP protocol as well.

Virus and Worm attack :- A program that has capability to infect other programs and make copies of itself and spread into other programs is called virus.
Programs that multiply like viruses but spread from computer to computer are called as worms.

E-Mail & IRC Related Crimes

- **Email spoofing:** Email spoofing refers to email that appears to have been originated from one source when it was actually sent from another source.

- **Email Spamming:** Email "spamming" refers to sending email to thousands and thousands of users - similar to a chain letter.
- **Sending malicious codes through email:** E-mails are used to send viruses, Trojans etc through emails as an attachment or by sending a link of website which on visiting downloads malicious code.
- **Email bombing:** E-mail "bombing" is characterized by abusers repeatedly sending an identical email message to a particular address.

Firewall

A firewall is a network security system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both.

How are Firewalls Used?

Network firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

Hardware and Software Firewalls

Firewalls can be either hardware or software but the ideal firewall configuration will consist of both. In addition to limiting access to your computer and network, a firewall is also useful for allowing remote access to a private network through secure authentication certificates and logins.

Hardware firewalls can be purchased as a stand-alone product but are also typically found in broadband routers, and should be considered an important part of your system and network set-up. Most hardware firewalls will have a minimum of four network ports to connect other computers, but for larger networks, business networking firewall solutions are available.

Software firewalls are installed on your computer (like any software) and you can customize it; allowing you some control over its function and protection features. A software firewall will protect your computer from outside attempts to control or gain access your computer.

Common Firewall Filtering Techniques

Firewalls are used to protect both home and corporate networks. A typical firewall program or hardware device filters all information coming through the Internet to your network or computer system. There are several types of firewall techniques that will prevent potentially harmful information from getting through:

Packet Filter :- Looks at each packet entering or leaving the network and accepts or rejects it based on user-defined rules. Packet filtering is fairly effective and transparent to users, but it is difficult to configure. In addition, it is susceptible to IP spoofing.

Application Gateway :- Applies security mechanisms to specific applications, such as FTP and Telnet servers. This is very effective, but can impose a performance degradation.

Circuit-level Gateway :- Applies security mechanisms when a TCP or UDP connection is established. Once the connection has been made, packets can flow between the hosts without further checking.

Proxy Server :- Intercepts all messages entering and leaving the network. The proxy server effectively hides the true network addresses.

In practice, many firewalls use two or more of these techniques in concert. A firewall is considered a first line of defense in protecting private information. For greater security, data can be encrypted.

Next Generation Firewall (NGFW)

A newer class of firewalls, next generation firewall - NGFW, filters network and Internet traffic based upon the applications or traffic types using specific ports. Next Generation Firewalls (NGFWs) blend the features of a standard firewall with quality of service (QoS) functionalities in order to provide smarter and deeper inspection.

Cookies

A cookie is information that a Web site puts on your hard disk so that it can remember something about you at a later time. (More technically, it is information for future use that is stored by the server on the client side of a client/server communication.) Typically, a cookie records your preferences when using a particular site. Using the Web's Hypertext Transfer Protocol (HTTP), each request for a Web page is independent of all other requests. For this reason, the Web page server has no memory of what pages it has sent to a user previously or anything about your previous visits. A cookie is a mechanism that allows the server to store its own information about a user on the user's own computer. You can view the cookies that have been stored on your hard disk (although the content stored in each cookie may not make much sense to you). The location of the cookies depends on the browser. Internet Explorer stores each cookie as a separate file under a Windows subdirectory. Netscape stores all cookies in a single cookies.txt fle. Opera stores them in a single cookies.dat file.

Hackers and Crackers

A cracker (also known as a black hat hacker) is an individual with extensive computer knowledge whose purpose is to breach or bypass internet security or gain access to software without paying royalties. The general view is that, while hackers build things, crackers break things. Cracker is the name given to hackers who break into computers for criminal gain; whereas, hackers can also be internet security experts hired to find vulnerabilities in systems. These hackers are also known as white hat hackers. Crackers' motivations can range from profit, a cause they believe in, general maliciousness or just because they like the challenge. They may steal credit card numbers, leave viruses, destroy files or collect personal information to sell.

Crackers can also refer to those who reverse engineer software and modify it for their own amusement. The most common way crackers gain access to networks or systems is through social engineering, whereby the cracker contacts employees at a company and tricks them into divulging passwords and other information that allows a cracker to gain access.

Types Of Payment System

Digital cash

Digital Cash (also known as e-currency, e-money, electronic cash, electronic currency, digital money, digital currency, cyber currency) refers to a system in which a person can securely pay for goods or services electronically without necessarily involving a bank to mediate the transaction. This article briefly describes the concept of Digital Cash and its features.

In ancient times, people would simply barter to obtain the goods and services they needed. The theory was quite simple: two parties each possessing a commodity the other wanted would enter an agreement to trade their commodities (goods or services). If party A had a Sheep and needs Cereals, it must not only find a party which has Cereals but also wants a Sheep (probably for its Wool). Further there might be party B which needs a Sheep, but has no Cereals to barter. This system was confusing, tiring and inefficient. Moreover if party A happens to find someone with whom they can trade their Sheep with Cereals, they might not think that few bags of grains are worth a Sheep. Both parties have to devise a formula to divide the Sheep and come to a conclusin, how many bags of Cereals is party A willing to accept for certain parts of the Sheep.

To solve such problems money came to resuce as a medium of exchange that facilitates trade. Various forms of Money were used in colonial times like beaver pelts, dried corn, copper coins, gold and so on. Paper money was introduced to lessen the burden of carrying and exchanging thousands of copper coins and also for the need of credit. Paper cash is popular amongst people as it is easy to carry, they can make payments with the received cash and they don't need a third party like a bank to intermediate. One of its major drawback is it can be stolen or lost and no one compensates the for the lost or stolen money.

Having gradually replaced central bank notes and bank checks with plastic cards, electronic fund transfers and automated clearing-houses(ACH), the world is now readying itself for the next step in the automation of money.

Credit cards do reduce the risk of lost cash to some extent, but at the same time there is a risk of losing ones privacy. Each year, credit cardcompanies and banks incure a huge loss since they are required to compensate for lost cards along with the costs associated with fraudulent transactions.
The solutions to the above problems faced by paper cash and credit cards is Digital Cash which is secure and protects people's privacy.

Electronic money is broadly defined as an electronic store of monetary value on a technical device that may be widely used for making payments to undertakings other than the issuer without necessarily involving bank accounts in the transaction, but acting as a prepaid bearer instrument.)

Basic Model of Digital Cash transaction

A Digital Cash transaction usually involves three types of users:

- a Payer (P) or consumer
- a Payee (R), such as a merchant
- a financial network like a Bank with whom both Payer and Payee have accounts.

And usually involves three transaction:

- Withdrawl, the Payer (P) transfer some money (token) from his/her account to her wallet (which could be a computer or smart case)
- Payment, the Payer (P) transfer the withdrawn money (token) to the Payee's (R) wallet
- Deposit, the Payee (R) transfers the received money (token) to his/her account.

Important properties of a Digital Cash system

- **Security:** The most important feature of a Digital Cash system is that it should ensure a high-level of security through sophisticated authentification techniques, Which means it should not be copied or reused by the payer, the payee or anyone else.
- **Anonymity:** It should be able to maintain the anonymity of the person, i.e the transaction carried out should not be traceable.
- **Portability:** The use of such a system should be independent of the location. The transactions can be carried over computer networks and into storage devices and vice versa.
- **Tranferability:** The user can spend the money received in payment without having to contact a bank for authentication
- **Divisibility:** This allows the digital cash to be sub-divided into smaller denominations and the customer can choose to spend only a part of it.
- **User friendly:** Both the payer and payee should be able to use it with ease which would make it widely acceptable.

Electronic Cheque

There was a time — not so very long ago — that when a person wrote a personal check to make a purchase, the recipient delivered it to their bank for deposit and it was processed manually. It was a process that could take days to complete, which meant the depositor had to wait for the funds to clear.

Today, new technology has eliminated much of the time and effort of the past by turning a paper check into an electronic transfer (debit), also known as an electronic check or e-check. If you're a merchant or service provider, this means funds are electronically transferred from a customer or client's bank account directly into your bank account through the Federal Reserve Bank's Automated Clearing House (ACH) system.

Electronic check payment processing is check cashing simplified, a hassle-free and inexpensive way to get paid quicker. It is also just one of the many important merchant services that TransFirst® provides to its clients to meet all their payment processing needs, including credit cards and debit cards. We offer electronic check payment processing through CrossCheck.

The e-Check Process

The e-check process begins when a customer writes a paper check at the point of sale. The clerk runs the check through a reader or imager that captures the required information, including the check number, account number and the number identifying the financial institution (routing number). The clerk enters additional merchant-related information to complete the one-time electronic payment from the designated account.

Depending on the merchant's agreement with their payment processor, the check is verified or guaranteed by the provider. A receipt is generated and printed for the check writer to sign; the check is voided and returned to them. The transaction will appear on the customer's bank statement as a debit, not a check.

The merchant uploads the captured check information to their payment provider for processing, and the proceeds are deposited into the merchant's account within 1 to 2 business days.

The process is similar for a check that is mailed in payment of a bill, except the check is retained by the merchant after it is voided.

Benefits of Electronic Check Processing

Electronic check processing offers numerous benefits for merchants. It eliminates the need to take checks to the bank and speeds up deposits. There are no check deposit slips to complete, and it reduces the chance of checks being lost or stolen before they are deposited.

For merchants working out of more than one location, checks can be converted at each location and funds can be centralized in one main account. Guarantee programs expedite the collection process by discovering NSF (insufficient funds) checks and other returned items quickly. And some payment providers make account activity available online for easy record keeping.

Smart Card

A smart card is a small, tamperproof computer. The smart card itself contains a CPU and some non-volatile storage. In most cards, some of the storage is tamperproof while the rest is accessible to any application that can talk to the card. This capability makes it possible for the card to keep some secrets, such as the private keys associated with any certificates it holds. The card itself actually performs its own cryptographic operations.

Although smart cards are often compared to hard drives, they're "secured drives with a brain"—they store and process information. Smart cards are storage devices with the core mechanics to facilitate communication with a reader or coupler. They have file-system configurations and the ability to be partitioned into public and private spaces that can be made available or locked.

They also have segregated areas for protected information, such as certificates, e-purses, and entire operating systems. In addition to traditional data storage states, such as read-only and read/write, some vendors are working with sub states best described as "add only" and "update only."

Smart cards currently come in two forms, contact and contactless.

- Contact cards require a reader to facilitate the bidirectional connection. The card must be inserted into a device that touches the contact points on the card, which facilitate communication with the card's chip. Contact cards come in 3-volt and 5-volt models, as do current desktop CPUs. Contact card readers are commonly built into company or vendor-owned buildings and assets, cellular phones, handheld devices, stand-alone devices that connect to a computer desktop's serial or Universal Serial Bus (USB) port, laptop card slots, and keyboards.
- Contactless cards use proximity couplers to get information to and from the card's chip. An antenna is wound around the circumference of the card and activated when the card is radiated in a specific distance from the coupler. The configuration of the card's antenna and the coupler facilitate connected states from a couple of centimeters to a couple of feet. The bidirectional transmission is encoded and can be encrypted by using a combination of a card vendor's hard-coded chip algorithms; randomly generated session numbers; and the card holder's certificate, secret key, or personal identification number (PIN). The sophistication of the connection can facilitate separate and discrete connections with multiple cards should they be within range of the coupler. Because

contactless cards don't require physical contact with a reader, the usability range is expanded tremendously.

Debit Card and Credit Card

Debit card

A debit card lets you pay for things with money from your bank account without needing cash. To use it you'll need your PIN number – normally the same PIN to use the card in a cash machine – which you enter into a card terminal in shops, or into a website when buying things online. The money is normally deducted from your bank balance in one to two days.

Credit card

Although using a credit card is similar to using a debit card, with a credit card money isn't immediately deducted from your bank account. This is because – instead of spending your own money – credit cards let you spend money the bank has lent you, up to a certain limit every month. The bank then sends you a bill for the total amount of everything you've bought once a month. You don't have to pay it back all at once, but you pay extra interest on any money you haven't paid back when it's added to the next month's total

Unit 3
Basic of HTML and Advance HTML 5

Introduction

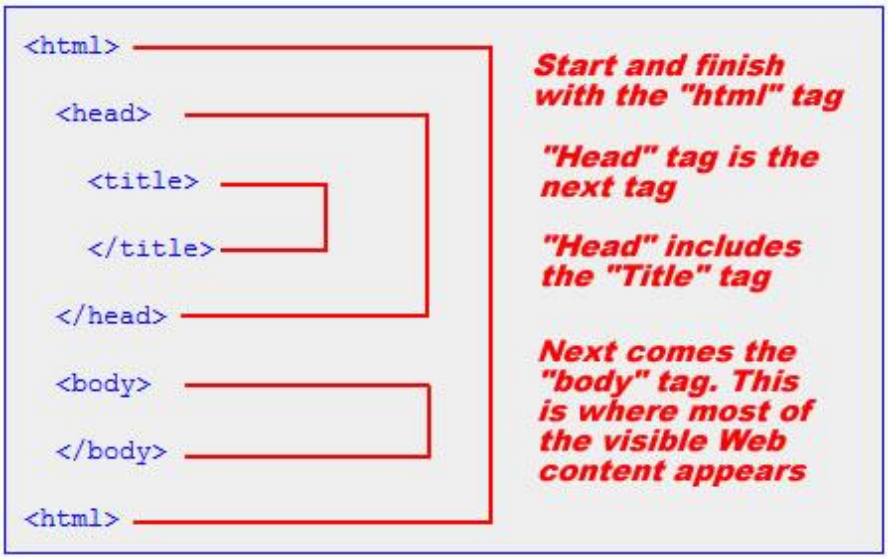
The Html stands for Hypertext Markup Language. The main goal of html is to be universal language for classifying the function of different section of a document. In other words html is used to define different parts of your page. To indicates which part of your document is tittled,which part of your document is addressed, which part of your document should be emphasized,which part of your document should include an image and so on.

Html is neither a page layout language nor a printing language,the only thing Html does is classifying parts of your documents so that a browser can display it correctly.This allow documents to be displayed on many kind of platform.
Although Html has evolved to the point that it contains many layout and formatting commands.This function are secondary to Html role in classifying the logical parts of your documents.

The important thing to remember is that Html is designed to work on a wide variety of platform not just ordinary personal computers platform like pcs.But Html is also designed to work on wide variety of graphical workstation.

The idea behind Html is that if you markup your document by indicating the parts of your document,then you should be able to trust that your document will be attractive and correctly displayed by any browser,or any computer anywhere in the world.This means that Html can be used to put a document on not just computer screen but also on printers,faxmachine,Tvsets and consoles.Imagine for example a browser that accept input from the touch tone buttons on a telephone line and then reads a webpage back to you over the phone(text to speech machine).

Structure of HTML



How to run

- Open the Start Screen (the window symbol at the bottom left on your screen). Type Notepad; and open it.
- Write or copy some HTML into Notepad.
- Save the file on your computer. Select File > Save as in the Notepad menu.
- Name the file "index.html" or any other name ending with html or htm.
- Press "Windows-E" to launch Windows Explorer.
- Navigate to the folder that contains your HTML file.
- Double-click the file. Your default browser displays the HTML document. If the browser is not open, Windows launches it.

Document Tags

Html

The <html> tag tells the browser that this is an HTML document. The <html> tag represents the root of an HTML document. The <html> tag is the container for all other HTML elements (except for the <!DOCTYPE> tag).

Head

The <head> element is a container for all the head elements. The <head> element can include a title for the document, scripts, styles, meta information, and more. The following elements can go inside the <head> element:

- <title> (this element is required in an HTML document)
- <style>
- <base>
- <link>
- <meta>
- <script>
- <noscript>

```
<html>
	<head>
		<title>Title of the document</title>
	</head>
	<body>
		The content of the document......
	</body>
</html>
```

Body

The <body> tag defines the document's body. The <body> element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

Attributes:

- **background:** It is used to set an image as a background of your page. Not supported in HTML5.

```
<html>
	<body background="bgimage.jpg">
		<h1>Hello world!</h1>
		<p>Welcome to Kundalia college</p>
	</body>
```

```
</html>
```

- **bgcolor :** It is used to set background color of your web page. Not supported in HTML5.

```
<html>
	<body bgcolor="#E6E6FA">
		<h1>Hello world!</h1>
		<p>Welcome to Kundalia college</p>
	</body>
</html>
```

- **text:** It is used to set a fore(text) color of your web page. Not supported in HTML5.

```
<html>
	<body text="green">
		<h1>Hello world!</h1>
		<p>Welcome to Kundalia college</p>
	</body>
</html>
```

```
<html>
	<body text="#AACC00">
		<h1>Hello world!</h1>
		<p>Welcome to Kundalia college</p>
	</body>
</html>
```

- **link:** It is used to set a fore color of your link. The default link color is blue. Not supported in HTML5.

```
<html>
	<body link="green">
		<a href="http://http://jjkcc.org/">Smt J J Kundalia Commerce
College</a>
	</body>
</html>
```

- **vlink :** [visited link] It is used to specify the color used for hypertext link that have been previous visited by user. Not supported in HTML5.

```
<html>
	<body link="green">
		<a href="http://http://jjkcc.org/">Smt J J Kundalia Commerce
College</p>
		<a href="http://http://jjkcc.org/">Smt J J Kundalia Commerce
College</p>
	</body>
</html>
```

- **alink:**[active link] It changes the color of active link color is displayed only while the mouse is held down on a link. The default color is red. Not supported in HTML5.

```
<html>
	<body link="green">
		<a href="http://http://jjkcc.org/">Smt J J Kundalia Commerce
College</p>
		<a href="http://http://jjkcc.org/">Smt J J Kundalia Commerce
College</p>
	</body>
</html>
```

```
<html>
	<body  vlink="#00AAFF"  alink="FFBBCC" text="yellow"
bgcolor="Black">
		Basic HTML Tutorials for Beginner
	</body>
</html>
```

Title

The <title> tag is required in all HTML documents and it defines the title of the document. The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search-engine results

```
<html>
	<head>
		<title>HTML Reference</title>
	</head>
	<body text="yellow"  bgcolor="Black">
		Basic HTML Tutorials for Beginner
	</body>
</html>
```

Comment

The comment tag is used to insert comments in the source code. Comments are not displayed in the browsers. You can use comments to explain your code, which can help you when you edit the source code at a later date. This is especially useful if you have a lot of code. Comments are also used to hide stylesheet and active scripts from older browser.

```
<html>
	<head>
		<title>HTML Reference</title>
	</head>
	<body text="yellow"  bgcolor="Black">
		<!--This is a comment. Comments are not displayed in the
browser-->
		<p>This is a paragraph.</p>
	</body>
</html>
script type="text/javascript">
	<!--
	function displayMsg() {
		alert("Hello World!")
	}
	//-->
</script>
```

Text Formatting Tags

<h1> to <h6>

The <h1> to <h6> tags are used to define HTML headings. <h1> defines the most important heading. <h6> defines the least important heading. Each heading tag will break a line.

```
<html>
	<body>
		<h1>This is heading 1</h1>
		<h2>This is heading 2</h2>
		<h3>This is heading 3</h3>
		<h4>This is heading 4</h4>
		<h5>This is heading 5</h5>
		<h6>This is heading 6</h6>
	</body>
</html>
```

Attributes

- **Align:** It is used to set alignment of the heading. The values can only be center, left, right or justify. Not supported in HTML5.

```
<html>
	<body>
		<h1 align="center">This is heading 1</h1>
	</body>
</html>
```

P

The <p> tag defines a paragraph. Browsers automatically add some space (margin) before and after each <p> element. The margins can be modified with CSS (with the margin properties).

```
<html>
	<body>
		<p>This is some text in a paragraph. This is some text in a
paragraph. This is some text in a paragraph. This is some text in a
paragraph.</p>
	</body>
</html>
```

- **Align:** It is used to set alignment of the paragraph. The values can only be center, left, right or justify. Not supported in HTML5.

```
<html>
	<body>
		<p align="right">This is some text in a paragraph.</p>
	</body>
</html>
```

Pre

Creates a block of pre formatted text(in which the normal rule of collapsing white space do not apply)so that you can arrange lines in a particular way. Most browsers will display the pre element using moonscape fonts such as courier. Note:- white space: more than one space, space by tab, space by enter

```
<html>
	<body>
		<pre>
Text in a pre element
is displayed in a fixed-width
font, and it preserves
both      spaces and
line breaks
</pre>
	</body>
</html>
```

B

The <b> tag specifies bold text. According to the HTML 5 specification, the <b> tag should be used as a LAST resort when no other tag is more appropriate. The HTML 5 specification states that headings should be denoted with the <h1> to <h6> tags, emphasized text should be denoted with the tag, important text should be denoted with the tag, and marked/highlighted text should use the <mark> tag. You can also use the CSS "font-weight" property to set bold text.

```
<html>
	<body>
		<p>This is normal text - <b>and this is bold text</b>.</p>
	</body>
</html>
```

U

The <u> tag represents some text that should be stylistically different from normal text, such as misspelled words or proper nouns in Chinese. Avoid using the <u> element where it could be confused for a hyperlink. The <u> element was deprecated in HTML 4.01. (the <u> element was used to define underlined text).
The <u> element is redefined in HTML5, to represent text that should be stylistically different from normal text, such as misspelled words or proper nouns in Chinese.

```
<html>
	<body>
		<p>This is a <u>paragraph</u>.</p>
	</body>
</html>
```

I

The <i> tag defines a part of text in an alternate voice or mood. The content of the <i> tag is usually displayed in italic. The <i> tag can be used to indicate a technical term, a phrase from another language, a thought, or a ship name, etc. Use the <i> element only when there is not a more appropriate semantic element, such as:

- (emphasized text)
- (important text)
- <mark> (marked/highlighted text)
- <cite> (the title of a work)
- <dfn> (a definition term)

```
<html>
	<body>
		<p>He named his car <i>The lightning</i>, because it was
very fast.</p>
	</body>
</html>
```

EM

The tag is a phrase tag. It renders as emphasized text. This tag is not deprecated, but it is possible to achieve richer effect with CSS.

```
<html>
	<body>
		<p>He named his car <em>The lightning</em>, because it was
very fast.</p>
	</body>
</html>
```

TT

The text appears in fixed width and in type writer fonts such as courier. Not supported in HTML5.

```
<html>
	<body>
		<p>He named his car <tt>The lightning</tt>, because it was
very fast.</p>
	</body>
</html>
```

Strike

The text appears with a strike out the middle. Not supported in HTML5. Use <del> or <s> instead.

```
<html>
	<body>
		<p>Version 2.0 is <strike>not yet available!</strike> now
available!</p>
	</body>
</html>
```

Sub

The <sub> tag defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H_2O.

```
<html>
	<body>
		<p>This text contains <sub>subscript</sub> text.</p>
	</body>
</html>
```

Sup

The <sup> tag defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW[1].

```
<html>
	<body>
		<p>This text contains <sup>superscript</sup> text.</p>
	</body>
</html>
```

Big

The big tag is used to increase the size of fonts by one to normal size of your text.

```
<html>
	<body>
		<p>Normal text ... <big>Bigger text</big></p>
	</body>
</html>
```

Small

The small tag is used to decrease the size of fonts by one to normal size of your text.

```
<html>
	<body>
		<p>Normal text ... <small>Smaller text</small></p>
	</body>
</html>
```

Strong

The strong tag is alternative of B tag. It defines important text. This tag is not deprecated, but it is possible to achieve richer effect with CSS.

```
<html>
	<body>
		<p>Normal text ... <strong>Strong text</strong></p>
	</body>
</html>
```

Font

It is used to set font color, font size and font face(font name). Not supported in HTML5. Use CSS instead.

Attributes

- **Color:** The color attribute is used to set text color of content within starting and ending tab of font.

```
<font color="color_name|hex_number|rgb_number">
color_name  Specifies the text color with a color name (like "red")
hex_number  Specifies the text color with a hex code (like "#ff0000")
rgb_number  Specifies the text color with an rgb code (like
"rgb(255,0,0)")
```

```
<html>
	<body>
		<p>Normal text ... <font color="red">This is some text with
red color</font></p>
	</body>
</html>
```

- **Size:** The Size will set size of the text. A number from 1 to 7 that defines the size of the text. Browser default is 3. The increment and decrement of the font size using + and – sign respectively will be in compare of your browser.

```
<html>
	<body>
		Your name
		<font color=”Red”  size=”7”>
			Your name or any content u like
		</font size=”+2”>
			Another content
		<font size=”-2”>
			More content
		</font>
	</body>
</html>
```

- **Face:** It is used to change font face(font name).

```
<font face="font_family">
font_family     The font of the text. To specify a prioritized
list of several fonts, separate the names with a comma (like this
<font face="verdana,arial,sans-serif">
```

```
<html>
      <body>
            <p>Your name...<font face="verdana">Your name</font></p>
      </body>
</html>
```

Marquee

The HTML <marquee> tag is used for scrolling piece of text or image displayed either horizontally across or vertically down your web site page depending on the settings

Attributes

- **direction** : Specifies the direction of text to scroll, the values can be left, right, up or down. The default direction is right to left.

```
<html>
      <body>
            <marquee direction=”up”>
                  Your text
            </marquee>
            <marquee direction=”right”>
                  Your text
            </marquee>
      </body>
</html>
```

- **scrolldelay :** Specifies the time in milliseconds before the marquee text begins to move. The default value 1 millisecond.

```
<html>
      <body>
            <marquee scrolldelay=”999”>
                  Your text with maximum value of delay value
            </marquee>
            <marquee scrolldelay=”1”>
                  Your text with minimum value of delay value
            </marquee>
      </body>
</html>
```

- **scrollamount** : It will increment or decrement the position of text in pixel. The default value is 1 pixel

```
 <html>
      <body>
            <marquee scrollamount=”50” direction=”right”>
                  Your text
            </marquee>
            <marquee scrollamount=”50” direction=”left”>
                  Your text
            </marquee>
      </body>
</html>
```

- **behaviour**: Specifies the type of motion used to animate the text in marquee. The values can only be scroll, slide and alternate.

```
<html>
	<body>
		<marquee>your text</marquee>
		<marquee behavior=”scroll” direction=”Right”>
			Your text
		</marquee>
		<marquee behavior=”Alternate” direction=”Left”>
			Your text
		</marquee>
		<marquee behavior=”slide”>
			Your text
		</marquee>
	</body>
</html>
```

- **bgcolor :** To set the background color only for the working area of marquee text.

```
<html>
	<body>
		<marquee bgcolor=”red”>
			Your text
		</marquee>
	</body>
</html>
```

- **height : width:** Specifies the height and width of a marquee in pixel or percentage(working area).

```
<html>
	<body>
		<marquee behavior=”up” height=”25%”>
			Your text
		</marquee>
		<marquee behavior=”left”  width=”50%”>
			Your text
		</marquee>
	</body>
<html>
```

- **loop:** Specifies the number of times marquee text should scroll. If attribute is not passed then text will scroll continuously only if the direction is scroll is scroll or alternate.

```
<html>
	<body>
		<marquee  Loop=”3”>
			Your text
		</marquee>
		<marquee behavior=”scroll”  width=”50”>
			Your text
		</marquee>
	</body>
</html>
```

Creating Link with Other Pages

Anchor

The <a> tag defines a hyperlink, which is used to link from one page to another. The most important attribute of the <a> element is the href attribute, which indicates the link's destination. By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

Attributes

- **href :** It stands for hyper text Reference. The value can be a URL or the local path of file with an extension.

```
<html>
      <body>
            <a href=”http:\\www.yahoo.com”>yahoo</a>
            <a href=”http:\\www.gmail.co.in”>Email</a>
            <a href=”c:\windows\xyz.jpg”>wallpaper </a>
      </body>
</html>
```

- **title :** Title attribute allows an advisor title that will explain the resource in more detail. A little box that appears when the mouse pointer is pointing to the link[tool tip] or [balloon help]

```
<html>
      <body>
            <a href="http:\\www.yahoo.com" >yahoo</a>
            <a href="http:\\www.gmail.co.in" title="click  here to check
your mail">Email</a>
            <a href="c:\windows\xyz.jpg">wallpaper </a>
      </body>
</html>
```

- **name :** It is used to link a particular line of any web page or of the same web page. Not supported in HTML5. Use the global id attribute instead.

```
<html>
      <body>
            <p>
                  <a href="#C10">See also Chapter 10</a>
            </p>
            <h2>Chapter 1</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 2</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 3</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 4</a></h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 5</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 6</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 7</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 8</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 9</h2>
            <p>This chapter explains ba bla bla</p>
```

```
            <h2><a name="C10">Chapter 10</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 11</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 12</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 13</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 14</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 15</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 16</h2>
            <p>This chapter explains ba bla bla</p>
            <h2>Chapter 17</h2>
            <p>This chapter explains ba bla bla</p>
      </body>
</html>
```

- **target:** The attribute is normally used with frames, you can use it even if you don't use any king of frames element. Basically it is used to indicate the name of window when you would like to the linked page or linked information to be appearing.

  _blank Opens the linked document in a new window or tab
  _self Opens the linked document in the same frame as it was clicked (this is default)
  _parent Opens the linked document in the parent frame
  _top Opens the linked document in the full body of the window
  framename Opens the linked document in a named frame

```
<html>
      <body>
            <a href="http://www.jjkcc.org" target="_blank">Smt J J
Kundalia Commerce College</a>
      </body>
</html>
```

Line Breaks Tags

Br

The
 tag inserts a single line break. The
 tag is an empty tag which means that it has no end tag. In XHTML, the
 tag must be properly closed, like this:
.

```
<html>
      <body>
            Kundalia
      <br><br>
            Commerce
      <br><br><br>
            College
      </body>
</html>
```

Hr

It stands for horizontal rule. The <hr> tag defines a thematic break in an HTML page (e.g. a shift of topic). The <hr> element is used to separate content (or define a change) in an HTML page.

Attribute

- **width:** The value should be in percentage or pixels. If the value is in percentage then percentage sign is compulsory. Not supported in HTML5.

```
<html>
	<body>
		<hr>
		<hr width=”150”>
			No of pixel alignment control
		<hr width=”25%”>
			25% of web page width.
	</body>
</html>
```

- **align:** The align attribute positions the rule on the page either left, right or center. Normally the rule fills entire width of screen. Aligning of rule is only useful when you have changed width of rule. Not supported in HTML5.

```
<html>
	<body>
		<hr align=”center”>
		<hr width=”15%”  align=”Right”>
		<hr width=”500” align=”left”>
	</body>
</html>
```

- **size:** The size attribute is a measurement of how the rule thick is. The number or the value must be in pixel only. If size is not passed then NV(navigator browser) and IE(Internet Explorer)will display the horizontal rule with the size 2(pixels). Not supported in HTML5.

```
<html>
	<body>
		<hr align=”center” Size=”4”>
		<hr width=”500” align=”left” size=”5”>
	</body>
</html>
```

- **color:** It is used to change the background color for the respective rule. When the color attribute is used it is better to use the size attribute with it. Not supported in HTML5.

```
<html>
	<body>
		<hr size=”4” color=”red”>
		<hr size=”10” color=”green”>
		<hr size=”20” color=”#aabbccc”>
	</body>
</html>
```

List Creation Tags

UL

Creates a unordered list with bullets preceding each list item. The starting ‘ul’ tag at the beginning of list and ending tag at the end of list each list item will be specified using starting and ending ‘li’ tag.(list item tag).

Attributes

- **type:** In an unordered list every list item has a bullet. The bullets can be any symbol in particular list. The symbols can be circle , disc and square.

```
<html>
	<body>
		<ul type="square">
		  <li>Coffee</li>
		  <li>Tea</li>
		  <li>Milk</li>
```

```
            </ul>
      </body>
</html>
```

LI

The <li> tag is used in ordered lists(<ol>), unordered lists (<ul>), and in menu lists (<menu>).

```
<html>
      <body>
            <ol>
              <li>Coffee</li>
              <li>Tea</li>
              <li>Milk</li>
            </ol>

            <ul>
              <li>Coffee</li>
              <li>Tea</li>
              <li>Milk</li>
            </ul>
      </body>
</html>
```

OL

The <ol> tag defines an ordered list. An ordered list can be numerical or alphabetical.

Attributes

- **type:** The type attribute specifies the kind of marker to use in the list (letters or numbers).

```
<ol type="1|a|A|i|I">
1  Default. Decimal numbers (1, 2, 3, 4)
a  Alphabetically ordered list, lowercase (a, b, c, d)
A  Alphabetically ordered list, uppercase (A, B, C, D)
i  Roman numbers, lowercase (i, ii, iii, iv)
I  Roman numbers, uppercase (I, II, III, IV)
```

```
<html>
      <body>
            <ol type="I">
              <li>Coffee</li>
              <li>Tea</li>
              <li>Milk</li>
            </ol>
      </body>
</html>
```

- **start:** The start attribute specifies the start value of the first list item in an ordered list.

```
<html>
      <body>
            <ol start="50">
              <li>Coffee</li>
              <li>Tea</li>
              <li>Milk</li>
            </ol>
      </body>
</html>
```

- **reversed:** The reversed attribute is a boolean attribute. When present, it specifies that the list order should be descending (9,8,7...), instead of ascending (1, 2, 3...). Only in HTML 5

```
<!DOCTYPE html>
<html>
	<body>
		<ol reversed>
		  <li>Coffee</li>
		  <li>Tea</li>
		  <li>Milk</li>
		</ol>
	</body>
</html>
```

DL

Definition list values appear within the <dl> and </dl> tag. The definition list contains two parts definition term and definition description.

DT

The definition term is passed by <dt> tag.

DD

The definition description is passed by <dd> tag.

```
<html>
	<body>
		<dl>
		  <dt>Coffee</dt>
		  <dd>Black hot drink</dd>
		  <dt>Milk</dt>
		  <dd>White cold drink</dd>
		</dl>
	</body>
</html>
```

Image Handling

Img

The <img> tag defines an image in an HTML page. The <img> tag has two required attributes: src and alt. Images are not technically inserted into an HTML page, images are linked to HTML pages. The <img> tag creates a holding space for the referenced image. To link an image to another document, simply nest the <img> tag inside <a> tags.

Attributes

- **src:** The src stands for the source the image which u want to display in the web page. The value could be the URL or the local path with the file name and extension of file.(path name is case sensitive in some case)

```
<html>
	<body>
		<img src="smiley.gif" alt="Smiley face">
	</body>
</html>
```

- **align:** Controls the alignment of the text following the image. The align attribute of <img> is not supported in HTML5. Use CSS instead. For the image to align middle, top, or bottom use the CSS property vertical-align. For the image to align left or right use the CSS property float. Not supported in HTML5.

```
<img align="left|right|middle|top|bottom">
left            Align the image to the left
right           Align the image to the right
middle          Align the image in the middle
top             Align the image at the top
bottom          Align the image at the bottom
```

```
<html>
      <body>
            <p>This is some text. <img src="smiley.gif" alt="Smiley
face" align="middle"> This is some text.</p>
      </body>
</html>
```

- **border:** Specifies the size of the border to place around the image. Not supported in HTML5.

```
<html>
      <body>
            <img src="smiley.gif" alt="Smiley face" border="5">
      </body>
</html>
```

- **width: height:** Specifies the height and width of image in pixel.

```
<html>
      <body>
            <img src="smiley.gif" alt="Smiley face" height="42"
width="42">
      </body>
</html>
```

- **alt:** Indicates the text to be displayed in case the browser is unable to display the image specified in the src attribute.

```
<html>
      <body>
            <img src="unknown.gif" alt="Smiley face">
      </body>
</html>
```

- **hspace:** The hspace attribute specifies the whitespace on left and right side of an image. The hspace attribute of <img> is not supported in HTML5. Use CSS instead.

```
<html>
      <body>
            <p><img src="smiley.gif" alt="Smiley face" hspace="20"> This
is some text. This is some text. This is some text.</p>
      </body>
</html>
```

- **vspace:** The vspace attribute specifies the whitespace on top and bottom of an image. The vspace attribute of <img> is not supported in HTML5. Use CSS instead.

```
<html>
      <body>
            <p><img src="smiley.gif" alt="Smiley face" vspace="50"> This
is some text. This is some text. This is some text.</p>
      </body>
</html>
```

Map

The <map> tag is used to define a client-side image-map. An image-map is an image with clickable areas. The required name attribute of the <map> element is associated with the <img>'s usemap attribute and creates a relationship between the image and the map. The <map> element contains a number of <area> elements that defines the clickable areas in the image map.

Attributes

- **name:** The name attribute is used to reference the map in the html document for an image

Area

The <area> tag defines an area inside an image-map (an image-map is an image with clickable areas). The <area> element is always nested inside a <map> tag. Note: The usemap attribute in the <img> tag is associated with the <map> element's name attribute, and creates a relationship between the image and the map.

Attributes

- **shape:** The shape of a region can be one of following rect, circle and polygon.

```
<area shape="default|rect|circle|poly">
default  Specifies the entire region
rect     Defines a rectangular region
circle   Defines a circular region
poly     Defines a polygonal region
```

- **coords:** The coords attribute specifies the coordinates of an area in an image-map. The coords attribute is used together with the shape attribute to specify the size, shape, and placement of an area. Tip: The coordinates of the top-left corner of an area are 0,0.

```
<area coords="value">
x1,y1,x2,y2    Specifies the coordinates of the left, top, right, bottom corner of the rectangle (for shape="rect")
x,y,radius     Specifies the coordinates of the circle center and the radius (for shape="circle")
x1,y1,x2,y2,..,xn,yn Specifies the coordinates of the edges of the polygon. If the first and last coordinate pairs are not the same, the browser will add the last coordinate pair to close the polygon (for shape="poly")
```

- **href:** Takes the names of the html file that is linked to the particular area on the image. In HTML5, if the <area> tag has no href attribute, it is a placeholder for a hyperlink.
- **alt**: The alt attribute specifies an alternate text for an area, if the image cannot be displayed. The alt attribute provides alternative information for an image if a user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader). The alt attribute is required if the href attribute is present. In HTML 4.01 the alt attribute is required. In HTML5, the alt attribute is only required if the href attribute is present. If the href attribute is not present, the alt attribute should not be used.

```
<!DOCTYPE html>
<html>
      <body>
```

```
	<p>Click on the sun or on one of the planets to watch it
closer:</p>

	<img src="planets.gif" width="145" height="126" alt="Planets"
usemap="#planetmap">

	<map name="planetmap">
		<area shape="rect" coords="0,0,82,126" alt="Sun"
href="sun.htm">
		<area shape="circle" coords="90,58,3" alt="Mercury"
href="mercur.htm">
		<area shape="circle" coords="124,58,8" alt="Venus"
href="venus.htm">
	</map>

	</body>
</html>
```

Tables Creation Tags

Table

The <table> tag defines an HTML table. An HTML table consists of the <table> element and one or more <tr>, <th>, and <td> elements. The <tr> element defines a table row, the <th> element defines a table header, and the <td> element defines a table cell. A more complex HTML table may also include <caption>, <col>, <colgroup>, <thead>, <tfoot>, and <tbody> elements. The "align", "bgcolor", "cellpadding", "cellspacing", "frame", "rules", "summary", and "width" attributes are not supported in HTML5.

Attributes

- **height: width:** Sets the height and width for the particular table. It could be in pixel or in percentage user should use % pattern for the dynamic output.

```
<!DOCTYPE html>
<html>
	<body>
		<table width="400">
			<tr>
				<th>Month</th>	<th>Savings</th>
			</tr>
			<tr>
				<td>January</td>	<td>$100</td>
			</tr>
		</table>
	</body>
</html>
```

- **border:** Sets the border of the rows and columns. The value is in pixel only. The default value is zero and the minimum value is 1.

```
<!DOCTYPE html>
<html>
	<body>
		<table border="3">
			<tr>
				<th>Month</th>	<th>Savings</th>
			</tr>
			<tr>
				<td>January</td>	<td>$100</td>
			</tr>
		</table>
	</body>
```

```
</html>
```

- **align:** The attribute is used for the alignment of the table left, right and center are the values.

```
<!DOCTYPE html>
<html>
	<body>
		<table align="right">
			<tr>
				<th>Month</th>	<th>Savings</th>
			</tr>
			<tr>
				<td>January</td>	<td>$100</td>
			</tr>
		</table>
	</body>
</html>
```

- **cellpadding:** Sets the space of the text from the starting of cell. The value is in pixels only.

```
<html>
	<body>
		<table cellpadding="10">
		  <tr>
			<th>Month</th>	<th>Savings</th>
		  </tr>
		  <tr>
			<td>January</td>	<td>$100</td>
		  </tr>
		</table>
	</body>
</html>
```

- **cellspacing:** Sets the space between two cells. The value is in pixel only.

```
<html>
	<body>
		<table cellspacing="10">
		  <tr>
			<th>Month</th>	<th>Savings</th>
		  </tr>
		  <tr>
			<td>January</td>	<td>$100</td>
		  </tr>
		</table>
	</body>
</html>
```

- **bgcolor:** The bgcolor attribute specifies a background color of a table.

```
<html>
	<body>
		<table bgcolor="#00FF00">
		  <tr>
			<th>Month</th>	<th>Savings</th>
		  </tr>
		  <tr>
			<td>January</td>	<td>$100</td>
		  </tr>
		</table>
	</body>
</html>
```

- **rules:** Used to display or not to display grids of a table. Values are all, rows, cols, none.

```
<html>
	<body>
		<table rules="rows">
		  <tr>
		      <th>Month</th>    <th>Savings</th>
		  </tr>
		  <tr>
		      <td>January</td>  <td>$100</td>
		  </tr>
		</table>
	</body>
</html>
```

TR

The <tr> tag defines a row in an HTML table. A <tr> element contains one or more <th> or <td> elements.

Attribute

- **align:** Aligns the content in a table row. Values left, right, center and justify.
- **bgcolor:** To set the background color of table row.
- **valign:** To set the vertical alignment of text in all cells of the row. The values are top, bottom and middle.

TD

The <td> tag defines a standard cell in an HTML table. An HTML table has two kinds of cells: Header cells - contains header information (created with the <th> element). Standard cells - contains data (created with the <td> element). The text in <th> elements are bold and centered by default. The text in <td> elements are regular and left-aligned by default.

Attribute

- **align**: Aligns the content in a cell. Values left, right, center, justify.
- **valign:** To set the vertical alignment of text in all cells of the row. The values are top, bottom and middle.

```
<html>
	<body>
		<table>
		  <tr>
		      <th>Month</th>    <th>Savings</th>
		  </tr>
		  <tr>
		      <td valign="bottom">January</td>
		      <td valign="bottom">$100</td>
		  </tr>
		</table>
	</body>
</html>
```

- **rowspan:** Sets the number of rows a cell should span.

```
<html>
	<body>
		<table>
		  <tr>
		      <th>Month</th>    <th>Savings</th>
		      <th>Savings for holiday!</th>
```

```
                </tr>
                <tr>
                    <td>January</td>
                    <td>$100</td>
                    <td rowspan="2">$50</td>
                </tr>
                <tr>
                    <td>February</td>        <td>$80</td>
                </tr>
            </table>
        </body>
</html>
```

- **colspan:** Specifies the number of columns a cell should span.

```
<html>
        <body>
              <table>
                <tr>
                    <th>Month</th>    <th>Savings</th>
                </tr>
                <tr>
                    <td>January</td>  <td>$100</td>
                </tr>
                <tr>
                    <td>February</td> <td>$80</td>
                </tr>
                <tr>
                    <td colspan="2">Sum: $180</td>
                </tr>
              </table>
        </body>
</html>
```

- **height: width:** Sets the height and width for the particular table. It could be in pixel or in percentage user should use % pattern for the dynamic output.

TH

The <th> tag defines a header cell in an HTML table. Generates a cell in the current row but the content will be in bold style(cell heading).

Attributes:

- All are same as TD tag.

Caption

The <caption> tag defines a table caption. The <caption> tag must be inserted immediately after the <table> tag. You can specify only one caption per table. By default, a table caption will be center-aligned above a table. However, the CSS properties text-align and caption-side can be used to align and place the caption.

Attributes

- **align:** Values are left, right, top, bottom and center. The top and center will have same output. There is not a much difference

```
<html>
        <body>
              <table>
                <caption>Monthly savings</caption>
                <tr>
```

```
                    <th>Month</th>     <th>Savings</th>
                </tr>
                <tr>
                    <td>January</td>  <td>$100</td>
                </tr>
            </table>
        </body>
</html>
```

Thead

The <thead> tag is used to group header content in an HTML table. The <thead> element is used in conjunction with the <tbody> and <tfoot> elements to specify each part of a table (header, body, footer). Browsers can use these elements to enable scrolling of the table body independently of the header and footer. Also, when printing a large table that spans multiple pages, these elements can enable the table header and footer to be printed at the top and bottom of each page. The <thead> tag must be used in the following context: As a child of a <table> element, after any <caption>, and <colgroup> elements, and before any <tbody>, <tfoot>, and <tr> elements.

Attributes

- **align**: Aligns the content . Values left, right, center, justify.
- **valign:** To set the vertical alignment. The values are top, bottom and middle.

Tfoot

The <tfoot> tag is used to group footer content in an HTML table. The <tfoot> element is used in conjunction with the <thead> and <tbody> elements to specify each part of a table (footer, header, body). The <tfoot> tag must be used in the following context: As a child of a <table> element, after any <caption>, <colgroup>, and <thead> elements and before any <tbody> and <tr> elements.

Attributes

- **align**: Aligns the content . Values left, right, center, justify.
- **valign:** To set the vertical alignment. The values are top, bottom and middle.

Tbody

The <tbody> tag is used to group the body content in an HTML table. The <tbody> element is used in conjunction with the <thead> and <tfoot> elements to specify each part of a table (body, header, footer).

Attributes

- **align**: Aligns the content . Values left, right, center, justify.
- **valign:** To set the vertical alignment. The values are top, bottom and middle.

```
<html>
      <body>
            <table>
              <thead>
              <tr>
                   <th>Month</th>     <th>Savings</th>
              </tr>
              </thead>
              <tfoot>
              <tr>
                    <td>Sum</td>      <td>$180</td>
              </tr>
```

```
                </tfoot>
                <tbody>
                <tr>
                     <td>January</td>  <td>$100</td>
                </tr>
                <tr>
                      <td>February</td>  <td>$80</td>
                </tr>
                </tbody>
              </table>
      </body>
</html>
```

Frames Tags

Frameset

The <frameset> tag is not supported in HTML5. The <frameset> tag defines a frameset. The <frameset> element holds one or more <frame> elements. Each <frame> element can hold a separate document. The <frameset> element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them. Not Supported in HTML5.

Attributes

- **rows:** Specifies the number and size of columns in a frameset
- **cols:** Specifies the number and size of rows in a frameset

Frame

The <frame> tag is not supported in HTML5. The <frame> tag defines one particular window (frame) within a <frameset>. Each <frame> in a <frameset> can have different attributes, such as border, scrolling, the ability to resize, etc. Not Supported in HTML5.

Attribute

- **name:** A name is given to identify it when specifying the target of the html document which is to be displayed in that particular frame.
- **src:** Src stands for the source. It specifies the URL or the local path of the html document to be displayed in a frame.
- **scrolling:** The attribute is used to describes whether the frame should have a scrollbar or not. If the value is yea a scroll bar is displayed if the value is no, no scrollbar is displayed and if the value is auto the browser decides whether the scrollbar should be displayed or not.
- **Noresize:** The attribute has no value to assign. If this is given the frame is not resizable by any user.

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

Forms Tags

Form

In the html document so far studied we designed on one way communication in the web page. The html documents open and show the message. Forms are the tools to improve user interface in the web. Using forms we can design a web page on which a user can communicate his wishes, opinion and suggestions.

Attributes

- **action:** Forms are used to get inputs from users. The user input is submitted to the server. The action attributes informs the browser the location of the server to which form input has to be submitted.
  For example suppose we want a program "xyz/abc/123.exe" to be executed when the form is submitted this means that 123.exe is to be executed when the file is submitted and the location of a file "xyz/abc".
  The name of the executable program and the location of directory depends upon the web server. We can also write a complete url, host name etc. if we leave out the starting part url the browser will submit form to the server which supplied the form.
- **method:** The attribute has only two choices of values. They are get and post. These denote to protocol, the server use in implementing the form features. Usually the value used for the method post.
  This is the recommended protocol with the post method the information from the user is put into the data stream of the http at the backend program can read the data as input through the standard input data steam.
  In the case of get method the data received in the form are placed at the end of url. If the form is very big and has number of inputs the get method cause the url to be very long. Usually get method can pass 25 characters in the total of characters inputted by the user in the form so get method is often discouraged.

Select

A drop down list presents a list to the user. The user can select his choice from the list. The definition for the drop down list is started with the starting select tag and ended with ending select tag.

Attributes

- **name:** The name attribute assigns name for a variable which will hold the selected choice.
- **size:** In the form when we click the arrow in the corner of the drop down list the list appear. When list is long it is difficult to manage on the screen for any user. So we can define the no. of items in the list to be shown. Other items can be seen by pressing the scroll arrows in the list.
- **multiple:** In the drop down list we can normally select only one item. There are certain cases in which the user can be given freedom to select more than one of the option.

Option

Each item of the list is mentioned with the help of option tag.(drop down list options)

Attribute

- **value:** The attribute value will be transferred to a variable specified by the name attribute in the select tag.

```
<html>
      <select>
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
      </select>
</html>
```

Input

The <input> tag specifies an input field where the user can enter data. <input> elements are used within a <form> element to declare input controls that allow users to input data. An input field can vary in many ways, depending on the type attribute. The tag is used for a various kinds of input from a user using values of type attribute

Attributes

- **type:** The type attribute specifies the type of <input> element to display. The default type is: text. *HTML5 has the following new input types: color, date, datetime, datetime-local, month, week, time, email, number, range, search, tel, and url.*

<input type="value">

button Defines a clickable button (mostly used with a JavaScript to activate a script)

```
<input type="button" value="Click me">
```

checkbox Defines a checkbox

```
<input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
<input type="checkbox" name="vehicle2" value="Car"> I have a car
```

color Defines a color picker

```
Select your favorite color: <input type="color" name="favcolor">
```

date Defines a date control (year, month and day (no time))

```
Birthday: <input type="date" name="bday">
```

datetime The input type datetime has been removed from the HTML standard. Use datetime-local instead.

datetime-local Defines a date and time control (year, month, day, hour, minute, second, and fraction of a second (no time zone)

```
Birthday (date and time): <input type="datetime-local" name="bdaytime">
```

email Defines a field for an e-mail address

```
E-mail: <input type="email" name="usremail">
```

file Defines a file-select field and a "Browse..." button (for file uploads)

```
Select a file: <input type="file" name="img">
```

hidden Defines a hidden input field

```
<input type="hidden" name="country" value="Norway">
```

number Defines a field for entering a number

```
Quantity (between 1 and 5): <input type="number" name="quantity" min="1"
max="5" step="2">
```

password Defines a password field (characters are masked)

```
<input type="password" name="pwd">
```

radio Defines a radio button

```
<input type="radio" name="gender" value="male"> Male<br>
<input type="radio" name="gender" value="female"> Female<br>
```

range Defines a control for entering a number whose exact value is not important (like a slider control)

```
<input type="range" name="points" min="0" max="10">
```

reset Defines a reset button (resets all form values to default values)

```
<input type="reset">
```

submit Defines a submit button

```
<input type="submit">
```

text Default. Defines a single-line text field (default width is 20 characters)

```
First name: <input type="text" name="fname"><br>
Last name: <input type="text" name="lname"><br>
```

url Defines a field for entering a URL

```
Add your homepage: <input type="url" name="homepage">
```

- **accept:** The accept attribute specifies the types of files that the server accepts (that can be submitted through a file upload). The accept attribute can only be used with <input type="file">. Do not use this attribute as a validation tool. File uploads should be validated on the server.

  <input accept="file_extension|audio/*|video/*|image/*|media_type">

  file_extension A file extension starting with the STOP character, e.g: .gif, .jpg, .png, .doc

  audio/* All sound files are accepted

  video/* All video files are accepted

  image/* All image files are accepted

- **autofocus**: The autofocus attribute is a boolean attribute. When present, it specifies that an <input> element should automatically get focus when the page loads.

```
<form action="demo_form.asp">
      First name: <input type="text" name="fname" autofocus><br>
      Last name: <input type="text" name="lname"><br>
      <input type="submit">
</form>
```

- **max: min:** The min attribute specifies the minimum value for an <input> element. The max attribute specifies the maximum value for an <input> element. The max and min attributes works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

```
<form action="demo_form.asp">
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-31">

  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-02">

  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5">

  <input type="submit">
</form>
```

- **maxlength:** The maxlength attribute specifies the maximum number of characters allowed in the <input> element. The maximum number of characters allowed in the <input> element. Default value is 524288.

```
<form action="demo_form.asp">
  Username: <input type="text" name="usrname" maxlength="10"><br>
  <input type="submit" value="Submit">
</form>
```

- **name:** The name attribute specifies the name of an <input> element. The name attribute is used to reference elements in a JavaScript, or to reference form data after a form is submitted. Only form elements with a name attribute will have their values passed when submitting a form.

```
<form action="demo_form.asp">
  Name: <input type="text" name="fullname"><br>
  Email: <input type="text" name="email"><br>
  <input type="submit" value="Submit">
</form>
```

- **placeholder:** The placeholder attribute specifies a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The short hint is displayed in the input field before the user enters a value. The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

```
<form action="demo_form.asp">
  <input type="text" name="fname" placeholder="First name"><br>
  <input type="text" name="lname" placeholder="Last name"><br>
  <input type="submit" value="Submit">
</form>
```

- **required:** The required attribute is a boolean attribute. When present, it specifies that an input field must be filled out before submitting the form. The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

```
<form action="demo_form.asp">
  Username: <input type="text" name="usrname" required>
  <input type="submit">
</form>
```

- **readonly:** The readonly attribute is a boolean attribute. When present, it specifies that an input field is read-only. A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it). The readonly attribute can be set to keep a user from changing the value until some other conditions have been met (like selecting a checkbox, etc.). Then, a JavaScript can remove the readonly value, and make the input field editable.

```
<form action="demo_form.asp">
  Country: <input type="text" name="country" value="Norway"
readonly><br>
  <input type="submit" value="Submit">
</form>
```

- **autocomplete:** The autocomplete attribute specifies whether or not an input field should have autocomplete enabled. Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values. The autocomplete attribute works with the following <input> types: text, search, URL, tel, email, password, datepickers, range, and color.

```
<form action="demo_form.asp" autocomplete="on">
  First name:<input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  E-mail: <input type="email" name="email" autocomplete="off"><br>
  <input type="submit">
</form>
```

Textarea

The text field can be used to get only a single line field input. The text area is a multiline area in which the user can input.

Attributes

- **name:** Same as input and select tag.
- **maxlength**: Same as input and select tag.
- **placeholder**: Same as input and select tag.
- **required**: Same as input and select tag.
- **readonly**: Same as input and select tag.
- **rows: cols**: The attribute will tell the number of rows and columns a text field visible at any notice that this is not a restrict for the text type. The text will have scroll bars for view the entire text.

```
<textarea rows="4" cols="50">
Welcome to Smt J J Kundalia Commerce College (Computer Science
Department)
Suchak Road, Near Shastri Maden, Rajkot: 360001. PH: 0281 - 2466007
</textarea>
```

HTML 5

- HTML5 is the latest and most enhanced version of HTML. Technically, HTML is not a programming language, but rather a markup language.
- HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.
- HTML5 is cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).
- The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears.

HTML5 Document – Web Page Structure

The following tags have been introduced for better structure –

- **section** – This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.
- **article** – This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.
- **aside** – This tag represents a piece of content that is only slightly related to the rest of the page.
- **header** – This tag represents the header of a section.
- **footer** – This tag represents a footer for a section and can contain information about the author, copyright information, etc.
- **nav** – This tag represents a section of the document intended for navigation.

- **dialog** – This tag can be used to mark up a conversation.
- **figure** – This tag can be used to associate a caption together with some embedded content, such as a graphic or video.

The markup for an HTM 5 document would look like the following –

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>...</title>
   </head>
   <body>
      <header>...</header>
      <nav>...</nav>
      <article>
         <section>
            ...
         </section>
      </article>

      <aside>...</aside>
      <figure>...</figure>
      <footer>...</footer>
   </body>
</html>
```

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>...</title>
   </head>
   <body>
      <header role="banner">
         <h1>HTML5 Document Structure Example</h1>
         <p>This page should be tried in safari, chrome or Mozila.</p>
      </header>
      <nav>
         <ul>
            <li><a href="#">HTML Tutorial</a></li>
            <li><a href="#">CSS Tutorial</a></li>
            <li><a href="#">JavaScript Tutorial</a></li>
         </ul>
      </nav>
      <article>
         <section>
            <p>Once article can have multiple sections</p>
         </section>
      </article>
      <aside>
         <p>This is  aside part of the web page</p>
      </aside>
       <figure align="right">
        <img src="/html5/images/logo.png" alt="bcafy" width="200"
height="100">
      </figure>
       <footer>
         <p>Created by <a href="#">bcafy students</a></p>
      </footer>
   </body>
</html>
```

Browser Support

The latest versions of Apple Safari, Google Chrome, Mozilla Firefox, and Opera all support many HTML5 features and Internet Explorer 9.0 will also have support for some HTML5 functionality. The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

New Features

HTML5 introduces a number of new elements and attributes that helps in building a modern website. Following are great features introduced in HTML5.

- **New Semantic Elements** – These are like <header>, <footer>, and <section>.
- **Forms 2.0** – Improvements to HTML web forms where new attributes have been introduced for <input> tag.
- **Persistent Local Storage** – To achieve without resorting to third-party plugins.
- **WebSocket** – A a next-generation bidirectional communication technology for web applications.
- **Server-Sent Events** – HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- **Canvas** – This supports a two-dimensional drawing surface that you can program with JavaScript.
- **Audio & Video** – You can embed audio or video on your web pages without resorting to third-party plugins.
- **Geolocation** – Now visitors can choose to share their physical location with your web application.
- **Microdata** – This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
- **Drag and drop** – Drag and drop the items from one location to another location on a the same webpage.

Backward Compatibility

- HTML5 is designed, as much as possible, to be backward compatible with existing web browsers. New features build on existing features and allow you to provide fallback content for older browsers.
- It is suggested to detect support for individual HTML5 features using a few lines of JavaScript.
- The HTML 5 language has a "custom" HTML syntax that is compatible with HTML 4 and XHTML1 documents published on the Web, but is not compatible with the more esoteric SGML features of HTML 4.
- HTML 5 does not have the same syntax rules as XHTML where we needed lower case tag names, quoting our attributes, an attribute had to have a value and to close all empty elements.
- But HTML5 is coming with lots of flexibility and would support the followings –
  - Uppercase tag names.
  - Quotes are optional for attributes.
  - Attribute values are optional.
  - Closing empty elements are optional.

The DOCTYPE

DOCTYPEs in older versions of HTML were longer because the HTML language was SGML based and therefore required a reference to a DTD.

HTML 5 authors would use simple syntax to specify DOCTYPE as follows −

```
<!DOCTYPE html>
```

Character Encoding

HTML 5 authors can use simple syntax to specify Character Encoding as follows −

```
<meta charset="UTF-8">
```

The <script> tag

It's common practice to add a type attribute with a value of "text/javascript" to script elements as follows −

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

HTML 5 removes extra information required and you can use simply following syntax −

```
<script src="scriptfile.js"></script>
```

The <link> tag

So far you were writing <link> as follows −

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

HTML 5 removes extra information required and you can use simply following syntax −

```
<link rel="stylesheet" href="stylefile.css">
```

HTML5 Elements

HTML5 elements are marked up using start tags and end tags. Tags are delimited using angle brackets with the tag name in between. The difference between start tags and end tags is that the latter includes a slash before the tag name.

Following is the example of an HTML5 element −

```
<p>...</p>
```

HTML5 tag names are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lower case.

Most of the elements contain some content like ... contains a paragraph. Some elements, however, are forbidden from containing any content at all and these are known as void elements. For example, br, hr, link and meta etc.

HTML5 Attributes

Elements may contain attributes that are used to set various properties of an element.

Some attributes are defined globally and can be used on any element, while others are defined for specific elements only. All attributes have a name and a value and look like as shown below in the example.

Following is the example of an HTML5 attributes which illustrates how to mark up a div element with an attribute named class using a value of "example" −

```
<div class="example">...</div>
```

Attributes may only be specified within start tags and must never be used in end tags.

HTML5 attributes are case insensitive and may be written in all upper case or mixed case, although the most common convention is to stick with lower case.

Standard Attributes

The attributes listed below are supported by almost all the HTML 5 tags.

Attribute	Options	Function
accesskey	User Defined	Specifies a keyboard shortcut to access an element.
align	right, left, center	Horizontally aligns tags
background	URL	Places an background image behind an element
bgcolor	numeric, hexidecimal, RGB values	Places a background color behind an element
class	User Defined	Classifies an element for use with Cascading Style Sheets.
contenteditable	true, false	Specifies if the user can edit the element's content or not.
contextmenu	Menu id	Specifies the context menu for an element.
data-XXXX	User Defined	Custom attributes. Authors of a HTML document can define their own attributes. Must start with "data-".
draggable	true,false, auto	Specifies whether or not a user is allowed to drag an element.
height	Numeric Value	Specifies the height of tables, images, or table cells.
hidden	hidden	Specifies whether element should be visible or not.
id	User Defined	Names an element for use with Cascading Style Sheets.
item	List of elements	Used to group elements.
itemprop	List of items	Used to group items.
spellcheck	true, false	Specifies if the element must have it's spelling or grammar checked.
style	CSS Style sheet	Specifies an inline style for an element.
subject	User define id	Specifies the element's corresponding item.
tabindex	Tab number	Specifies the tab order of an element.
title	User Defined	"Pop-up" title for your elements.
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
width	Numeric Value	Specifies the width of tables, images, or table cells.

Custom Attributes

A new feature being introduced in HTML 5 is the addition of custom data attributes. A custom data attribute starts with data- and would be named based on your requirement. Following is the simple example –

```
<div class="example" data-subject="physics" data-level="complex">
   ...
</div>
```

The above will be perfectly valid HTML5 with two custom attributes called data-subject and data-level. You would be able to get the values of these attributes using JavaScript APIs or CSS in similar way as you get for standard attributes.

Web Forms 2.0 is an extension to the forms features found in HTML4. Form elements and attributes in HTML5 provide a greater degree of semantic mark-up than HTML4 and remove a great deal of the need for tedious scripting and styling that was required in HTML4.

<output> element

HTML5 introduced a new element <output> which is used to represent the result of different types of output, such as output written by a script. You can use the for attribute to specify a relationship between the output element and other elements in the document that affected

the calculation (for example, as inputs or parameters). The value of the for attribute is a space-separated list of IDs of other elements.

```
<!DOCTYPE HTML>
<html>
<head>
   <script type="text/javascript">
      function showResult()  {
         x = document.forms["myform"]["newinput"].value;
         document.forms["myform"]["result"].value=x;
      }
   </script>
</head>
   <body>
      <form action="/cgi-bin/html5.cgi" method="get" name="myform">
         Enter a value : <input type="text" name="newinput" />
         <input type="button" value="Result"  onclick="showResult();" />
         <output name="result"></output>
      </form>
   </body>
</html>
```

Audio and video

HTML5 features, include native audio and video support without the need for Flash. The HTML5 <audio> and <video> tags make it simple to add media to a website. You need to set src attribute to identify the media source and include a controls attribute so the user can play and pause the media.

Embedding Video

Here is the simplest form of embedding a video file in your webpage –

```
<video src="foo.mp4"  width="300" height="200" controls>
   Your browser does not support the <video> element.
</video>
```

The current HTML5 draft specification does not specify which video formats browsers should support in the video tag. But most commonly used video formats are –

- Ogg – Ogg files with Thedora video codec and Vorbis audio codec.
- mpeg4 – MPEG4 files with H.264 video codec and AAC audio codec.

You can use <source> tag to specify media along with media type and many other attributes. A video element allows multiple source elements and browser will use the first recognized format

```
<!DOCTYPE HTML>
<html>
   <body>

      <video  width="300" height="200" controls autoplay>
         <source src="/html5/foo.ogg" type="video/ogg" />
         <source src="/html5/foo.mp4" type="video/mp4" />
         Your browser does not support the video element.
      </video>

   </body>
</html>
```

Video Attribute Specification

The HTML5 video tag can have a number of attributes to control the look and feel and various functionalities of the control –

Attribute	Description
autoplay	This boolean attribute if specified, the video will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
autobuffer	This boolean attribute if specified, the video will automatically begin buffering even if it's not set to automatically play.
controls	If this attribute is present, it will allow the user to control video playback, including volume, seeking, and pause/resume playback.
height	This attribute specifies the height of the video's display area, in CSS pixels.
loop	This boolean attribute if specified, will allow video automatically seek back to the start after reaching at the end.
preload	This attribute specifies that the video will be loaded at page load, and ready to run. Ignored if autoplay is present.
poster	This is a URL of an image to show until the user plays or seeks.
src	The URL of the video to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed
width	This attribute specifies the width of the video's display area, in CSS pixels.

Embedding Audio

HTML5 supports <audio> tag which is used to embed sound content in an HTML or XHTML document as follows.

```
<audio src="foo.wav" controls autoplay>
   Your browser does not support the <audio> element.
</audio>
```

The current HTML5 draft specification does not specify which audio formats browsers should support in the audio tag. But most commonly used audio formats are ogg, mp3 and wav.

You can use <source> tag to specify media along with media type and many other attributes. An audio element allows multiple source elements and browser will use the first recognized format –

```
<!DOCTYPE HTML>
<html>
   <body>

      <audio controls autoplay>
         <source src="/html5/audio.ogg" type="audio/ogg" />
         <source src="/html5/audio.wav" type="audio/wav" />
         Your browser does not support the audio element.
      </audio>

   </body>
</html>
```

Audio Attribute Specification

The HTML5 audio tag can have a number of attributes to control the look and feel and various functionalities of the control:

Attribute	Description
autoplay	This boolean attribute if specified, the audio will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
autobuffer	This boolean attribute if specified, the audio will automatically begin buffering even if it's not set to automatically play.
controls	If this attribute is present, it will allow the user to control audio playback, including volume, seeking, and pause/resume playback.
loop	This boolean attribute if specified, will allow audio automatically seek back to the start after reaching at the end.
preload	This attribute specifies that the audio will be loaded at page load, and ready to run. Ignored if autoplay is present.
src	The URL of the audio to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed

Handling Media Events

The HTML5 audio and video tag can have a number of attributes to control various functionalities of the control using Javascript –

Event	Description
abort	This event is generated when playback is aborted.
canplay	This event is generated when enough data is available that the media can be played.
ended	This event is generated when playback completes.
error	This event is generated when an error occurs.
loadeddata	This event is generated when the first frame of the media has finished loading.
loadstart	This event is generated when loading of the media begins.
pause	This event is generated when playback is paused.
play	This event is generated when playback starts or resumes.
progress	This event is generated periodically to inform the progress of the downloading the media.
ratechange	This event is generated when the playback speed changes.
seeked	This event is generated when a seek operation completes.
seeking	This event is generated when a seek operation begins.
suspend	This event is generated when loading of the media is suspended.
volumechange	This event is generated when the audio volume changes.
waiting	This event is generated when the requested operation (such as playback) is delayed pending the completion of another operation (such as a seek).

Following is the example which allows to play the given video –

```
<!DOCTYPE HTML>
<html>
   <head>
      <script type="text/javascript">
         function PlayVideo(){
            var v = document.getElementsByTagName("video")[0];
            v.play();
         }
      </script>
   </head>
   <body>
     <form>
         <video  width="300" height="200" src="/html5/foo.mp4">
         Your browser does not support the video element.
         </video>
```

```
         <br />
         <input type="button" onclick="PlayVideo();"  value="Play"/>
      </form>
  </body>
</html>
```

Canvas

The HTML <canvas> element is used to draw graphics on a web page. The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

What is HTML Canvas?

The HTML <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript). The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics. Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content. The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Note: Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute:

```
<!DOCTYPE html>
<html>
      <body>
      <canvas id="myCanvas" width="200" height="100" style="border:1px
solid #000000;">
            Your browser does not support the HTML5 canvas tag.
      </canvas>
      </body>
</html>
```

Drawing with JavaScript

```
<!DOCTYPE html>
<html>
      <body>
            <canvas id="myCanvas" width="200" height="100"
style="border:1px solid #c3c3c3;">
            Your browser does not support the HTML5 canvas tag.
            </canvas>
            <script>
                  var c = document.getElementById("myCanvas");
                  var ctx = c.getContext("2d");
                  ctx.fillStyle = "#FF0000";
                  ctx.fillRect(0,0,150,75);
            </script>
      </body>
</html>
```

Draw a Line

```
<!DOCTYPE html>
<html>
      <body>
            <canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
            Your browser does not support the HTML5 canvas tag.
            </canvas>
```

```
            <script>
                  var c = document.getElementById("myCanvas");
                  var ctx = c.getContext("2d");
                  ctx.moveTo(0,0);
                  ctx.lineTo(200,100);
                  ctx.stroke();
            </script>
      </body>
</html>
```

Draw a Circle

```
<!DOCTYPE html>
<html>
      <body>
            <canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
                  Your browser does not support the HTML5 canvas tag.
            </canvas>
            <script>
                  var c = document.getElementById("myCanvas");
                  var ctx = c.getContext("2d");
                  ctx.beginPath();
                  ctx.arc(95,50,40,0,2*Math.PI);
                  ctx.stroke();
            </script>
      </body>
</html>
```

Draw a Text

```
<!DOCTYPE html>
<html>
      <body>
            <canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
            Your browser does not support the HTML5 canvas tag.
            </canvas>
            <script>
                  var c = document.getElementById("myCanvas");
                  var ctx = c.getContext("2d");
                  ctx.font = "30px Arial";
                  ctx.fillText("Hello World",10,50);
            </script>
      </body>
</html>
```

Stroke Text

```
<!DOCTYPE html>
<html>
      <body>
            <canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
            Your browser does not support the HTML5 canvas tag.
            </canvas>
            <script>
                  var c = document.getElementById("myCanvas");
                  var ctx = c.getContext("2d");
                  ctx.font = "30px Arial";
                  ctx.strokeText("Hello World",10,50);
            </script>
      </body>
</html>
```

Unit 4

Cascading Style Sheet & CSS 3

Introduction

Style sheets are powerful mechanism for adding styles to web documents. They enforce standards and uniformity throughout a web site and provide numerous attributes to create dynamic effects. With style sheets, text and image formatting properties can be predefined in a single line. HTML elements on a web page can then be bound to the style sheet. The advantage of a style sheet includes the ability to make global changes to all documents from a single location. Style sheets are said to Cascade when they combine to specify the appearance of a page.

Advantages of CSS

Using CSS we can keep the content of an html file and the style and layout of the file separate from each other. This gives several advantages to CSS over the traditional HTML.

- CSS keeps the file size smaller: Unlike HTML where we need to write the code to specify the font size, color, style etc. for every single page of a document that consists of more than one page, In CSS we need to specify the attributes only once for each element and the specified style will automatically be applied whenever the element occurs. Since the style and layout elements are separated from the HTML this allows the document to be much smaller in size.
- The pages of the website loads faster: Since the HTML pages have less code after the style and layout is separated from it, the pages will take less time to load.
- Lower bandwidth requirement: When the pages of a website take lesser time to load, the bandwidth requirement of the server is also reduced. This also makes CSS a cost effective option.
- CSS saves time: With CSS, changing the style of a document becomes very easy and it also saves much time. In HTML if we want to make even a small change in the entire website (such as changing the font style), we need to open every single page and alter the font style manually, which makes the process to time consuming. However, with we just need to open the CSS file where the layout of the site is stored and change the font style, which in turn, will change the font style of the entire website.
- CSS is easier to manage: As to make a change throughout a website that uses CSS, one need to make an edit only in a single file, it is easier to manage CSS. This greatly helps when the website has several pages.

Disadvantage of CSS

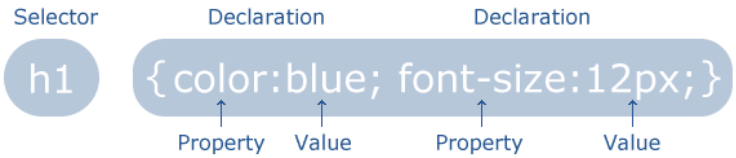
One major disadvantage of CSS is its incompatibility to different browsers. While some browsers show partial compatibility to Style Sheets and support only some of the features of style sheets, very few browsers such as the early versions of Internet Explorer and Netscape show almost complete incompatibility to CSS. However, the latest browsers versions show more standards-compliant and have largely overcome the compatibility problem.

Embedding the style sheet in html document

Between the <style> and </style> html tags, specific style attributes are listed. The <style> and </style> tags are written within the <head> and </head> tags.

Syntax

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



The selector is normally the Html tag, the property is the attribute we can change, and each property can take a value. The property and value are separated by a colon and surrounded by curly braces

- Examples

```
P{color:red}
```

The example will set text color red for all paragraph tag in the web page

- Examples

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

This will set the text color red and align the paragraph as center.

- Example

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

This will do the same as the example 2, user can write style sheet in the multiple lines also to make is more readable.

- Example

```
P,h2,h4{
      Color:red;
      Text-align:center
}
```

If same Dhtml property is too provided to multiple Html tag, each Html tag can be separated with the comma. The example will set style sheet for p, h2 and h4 tags.

Comments

In Dhtml the comments begin with forward slash and an asterisk sign and terminate with in reverse order

```
Dhtml coding
/* the comments goes here */
Coding continues here
```

```
Starting Style tag
      <!- -
            CSS content to be hide from browser which not supporting it.
      -->
Ending Style tag
```

To protect browser that do not support <style> elements, insert comment tags around the declarations within the style elements

Types of Style Sheets

Inline style sheet

When a style is intended for a single place or item on the entire website, a single tag CSS is used. As the term suggests, a single tag CSS affects only a single tag in a HTML document for which it is defined. Inline styles are used for this purpose. In inline style the style attribute must be used in the relevant tag that renders the style. The style attribute may have any CSS property. The following example shows how a change in the color and the right margin of a paragraph is brought which is different from the rest of the page.

```
<html>
      <body>
            <p style="color: teal; margin-right: 40px";>
This paragraph has different style </p>
      </body>
</html>
```

```
<html>
      <body style="background-color:#FF00FF">
            <h2 style="color:red">
                        Here goes your paragraph
            </h2>
      </body>
</html>
```

Internal style sheet

When a style is intended to appear at several places on a single web page, a CSS style is used which is defined once for the entire page. Internal style sheets (or embedded style sheets) are used for this purpose. Internal style sheets are also called embedded style sheets.

```
<html>
      <head>
            <style type="text/css">
                  p{color:#FFCC00}
                  h2{color:#3399FF}
                  h4,h6{color:#66FF00}
            </style>
      </head>
      <body>
            <h2> your first heading </h2>
            <p> your paragraph </p>
            <h4> your second heading </h4>
            <h6> your third heading </h6>
      </body>
</html>
```

External style sheet

The third and the most powerful method of inserting a style sheet is to link the HTML document to an external style sheet. When a style is intended to appear at more than one page in the website, a CSS style is used which is described in an external CSS file. External style sheets (or linked style sheets) are used for this purpose. External style sheets are also called linked style sheets.

Inserting an external style sheet provides complete ease to the designer and also save lot of time whereby the designer can change the look of several pages or of the entire website by making the necessary change in only one file. The external style sheet is linked to each of the pages with the <link> tag which goes inside the head section. The following example shows how an external file is linked to a document:

```
/* CSS Document */
p{color:#FFCC00}
h2{color:#3399FF}
h4,h6{color:#66FF00}
```

```
/*Html Web Page */
<html>
      <head>
```

```
            <link href="external.css" rel="stylesheet" type="text/css">
      </head>
      <body>
            <h2> your first heading </h2>
            <p> your paragraph </p>
            <h4> your second heading </h4>
            <h6> your third heading </h6>
      </body>
</html>
```

In the example save CSS Document with any name having extension .css, in the Html Web Page 'external.css' should be replacing with name of CSS document where it is been called with the help of <link> tag in the Html document. The Html page can be saved with any name.

Note: User should note that in type attribute of <style> tag forward slash should be done, not a back slash "text/css"

Rel attribute defines the relation between the external document and the calling document. In this case, the relation is that the external document is the style sheet for the calling document.

Type attribute refers to MIME type of the external file.

Href attribute is same used in <a>, Href stands for hyperlink reference. It accepts the path of the style sheet document. CSS document 'external.css' can be hyperlink in any number of Html documents, one change to CSS document will affect in every Html document.

The Hierarchy

There are three types of Cascading Style Sheets, which follow a hierarchy.

- Inline
- Embedded and
- Linked/(or) External

The "cascading" refers to the hierarchy of control

- Inline style takes preference
- Then Embedded styles rules follow
- The Linked external style sheet instructions will be used if Inline or Embedded instructions are not present

Whenever any conflict arises as to which style rule should be used first, a style rule with higher preference will get preference.

The following chart explains this principle. The chart is only a brief reference to resolve style conflicts. If multiple style rules are in conflict for a given selector, the scale shown in the chart will help you in determining the order of style rules to be used. A style rule with higher importance will be preferred than an identical style rule with lower importance.

Lowest Importance ←==========→ Highest Importance			
Specification Method:	Linked Style Sheet	Embedded Style Sheet	Inline Styles
Element Selector:	Contextual Selector	Class	ID

Tag Selectors

- By using Tag Selectors we can define styling for existing HTML tags
- Often used to set the basic styles that will appear throughout a Web site

The general syntax for an ID selector is HTML Selector {Property:Value;}

```
b {font-family:Arial, Helvetica, sans-serif; font-size:14px;  color:
red;}
```

This means Arial font of size 14 and red in color is stored within the tag b.

```
body {
font-family: Arial;
font-size: 14px;
color: red;
text-align: left; }
```

The meaning of the CSS code above is: every text inside the body tag will be appeared in Arial font face with size 14 px, red color and left alignment. As the tag selector is defined within the body tag only the text appearing in the body will change. However the paragraph text, header text and others will remain unaffected.

ID Selectors

The id selector uses the id attribute of an HTML element to select a specific element.
The id of an element should be unique within a page, so the id selector is used to select one unique element! To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
<html>
	<head>
		<style type="text/css">
			#a{color:#FFCC00}
			#b{color:#CC00FF}
			#c,#d{color:#9900FF}
		</style>
	</head>
	<body>
		<p id="a">Text with id A</p>
<p id="b">Text with id B</p>
		<p id="c">Text with id C</p>
		<p id="d">Text with id D</p>
		<b id="b">Text with id B and bold tag</b>
	</body>
</html>
```

Class Selectors

The class selector selects elements with a specific class attribute. To select elements with a specific class, write a period (.) character, followed by the name of the class.

```
<html>
	<head>
		<style type="text/css">
			/* CSS Document */
			.a{color:#FFCC00}
			.b{color:#CC00FF}
			.c,.d{color:#9900FF}
		</style>
	</head>
	<body>
		<p class="a">Text with class A</p>
		<p class="b">Text with class B</p>
		<p class="c">Text with class C</p>
		<p class="d">Text with class D</p>
		<b class="b">Text with class B and bold tag</b>
	</body>
</html>
```

Color Properties

Property	Description	CSS
color	Sets the color of text	1
opacity	Sets the opacity level for an element	3

```
h1 {
    color: #00ff00;
}
div {
    opacity: 0.5;
}
```

Background – Border Propertites

Property	Description	CSS
background	A shorthand property for setting all the background properties in one declaration *bg-color bg-image position bg-repeat bg-attachment*	1
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page - *scroll|fixed|local|initial|inherit*	1
background-color	Specifies the background color of an element	1
background-image	Specifies one or more background images for an element	1
background-position	Specifies the position of a background image - *left top|left center|left bottom|right top|right center|right bottom|center top|center center|center bottom|xpos ypos|x% y%|inherit|initial*	1
background-repeat	Sets how a background image will be repeated – *repeat|repeat-x|repeat-y|no-repeat|initial|inherit*	1
border	Sets all the border properties in one declaration *border-width border-style border-color*|initial|inherit;	1
border-bottom	Sets all the bottom border properties in one declaration *border-width border-style border-color|initial|inherit;*	1
border-bottom-color	Sets the color of the bottom border *color|transparent|initial|inherit;*	1
border-bottom-left-radius	Defines the shape of the border of the bottom-left corner *length|% [length|%]|initial|inherit;*	3
border-bottom-right-radius	Defines the shape of the border of the bottom-right corner *length|% [length|%]|initial|inherit;*	3
border-bottom-style	Sets the style of the bottom border *none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset|initial|inherit*	1
border-bottom-width	Sets the width of the bottom border *medium|thin|thick|length|initial|inherit;*	1
border-color	Sets the color of the four borders *color|transparent|initial|inherit;*	1
border-left	Sets all the left border properties in one declaration	1
border-left-color	Sets the color of the left border	1
border-left-style	Sets the style of the left border	1
border-left-width	Sets the width of the left border	1
border-radius	A shorthand property for setting all the four border-*-radius properties *1-4 length*|% / *1-4 length*|%|initial|inherit; **Four values:** first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner **Three values:** first value applies to top-left, second value applies to top-right and bottom-left, and third value applies to bottom-right **Two values:** first value applies to top-left and bottom-right corner, and the second value applies to top-right and bottom-left corner **One value:** all four corners are rounded equally	3
border-right	Sets all the right border properties in one declaration	1
border-right-color	Sets the color of the right border	1
border-right-style	Sets the style of the right border	1

border-right-width	Sets the width of the right border	1
border-style	Sets the style of the four borders	1
border-top	Sets all the top border properties in one declaration	1
border-top-color	Sets the color of the top border	1
border-top-left-radius	Defines the shape of the border of the top-left corner	3
border-top-right-radius	Defines the shape of the border of the top-right corner	3
border-top-style	Sets the style of the top border	1
border-top-width	Sets the width of the top border	1
border-width	Sets the width of the four borders	1
box-shadow	Attaches one or more drop-shadows to the box *none|h-shadow v-shadow blur spread color |inset|initial|inherit;*	3

```
body {
    background: #00ff00 url("smiley.gif") no-repeat fixed center;
}
body {
    background-image: url('w3css.gif');
    background-repeat: no-repeat;
    background-attachment: fixed;
}
h1 {
    background-color: #00ff00;
}

body {
    background-image: url("paper.gif");
}
body {
    background-image: url('smiley.gif');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: center;
}
body {
    background-image: url("paper.gif");
    background-repeat: repeat-y;
}

p {
    border: 5px solid red;
}
p {
    border-style: solid;
    border-bottom: thick dotted #ff0000;
}
div {
    border: 2px solid;
    border-radius: 25px;
}

div {
    box-shadow: 10px 10px 5px #888888;
}
```

Basic Box Properties

Property	Description	CSS
clear	Specifies which sides of an element where other floating elements are not allowed *none|left|right|both|initial|inherit;*	1
display	Specifies how a certain HTML element should be displayed *inline|block|inline-block*	1
float	Specifies whether or not a box should float *none|left|right|initial|inherit;*	1
height	Sets the height of an element *auto|length|initial|inherit;*	1
left	Specifies the left position of a positioned element *auto|length|initial|inherit;*	2
margin	Sets all the margin properties in one declaration *length|auto|initial|inherit;* **margin:10px 5px 15px 20px;** top margin is 10px right margin is 5px bottom margin is 15px left margin is 20px **margin:10px 5px 15px;** top margin is 10px right and left margins are 5px bottom margin is 15px **margin:10px 5px;** top and bottom margins are 10px right and left margins are 5px **margin:10px;** all four margins are 10px	1
margin-bottom	Sets the bottom margin of an element	1
margin-left	Sets the left margin of an element	1
margin-right	Sets the right margin of an element	1
margin-top	Sets the top margin of an element	1
max-height	Sets the maximum height of an element *none|length|initial|inherit;*	2
max-width	Sets the maximum width of an element *none|length|initial|inherit;*	2
min-height	Sets the minimum height of an element	2
min-width	Sets the minimum width of an element	2
overflow	Specifies what happens if content overflows an element's box *visible|hidden|scroll|auto|initial|inherit;*	2
padding	Sets all the padding properties in one declaration *length|initial|inherit;* **padding:10px 5px 15px 20px;** top padding is 10px right padding is 5px bottom padding is 15px left padding is 20px **padding:10px 5px 15px;** top padding is 10px right and left padding are 5px	1

	bottom padding is 15px **padding:10px 5px;** top and bottom padding are 10px right and left padding are 5px **padding:10px;** all four paddings are 10px	
padding-bottom	Sets the bottom padding of an element *length|initial|inherit;*	1
padding-left	Sets the left padding of an element	1
padding-right	Sets the right padding of an element	1
padding-top	Sets the top padding of an element	1
position	Specifies the type of positioning method used for an element *static|absolute|fixed|relative|initial|inherit;*	2
right	Specifies the right position of a positioned element *auto|length|initial|inherit;*	2
top	Specifies the top position of a positioned element *auto|length|initial|inherit;*	2
visibility	Specifies whether or not an element is visible *visible|hidden|collapse|initial|inherit;*	2
width	Sets the width of an element *auto|value|initial|inherit;*	1
vertical-align	Sets the vertical alignment of an element *baseline|length|sub|super|top|text-top|middle|bottom|text-bottom|initial|inherit;*	1
z-index	Sets the stack order of a positioned element *auto|number|initial|inherit;*	2

```
p.clear {
    clear: both;
}

p.inline {
    display: inline;
}

img  {
    float: right;
}

p.ex {
    height: 100px;
    width: 100px;
}


div.absolute {
    position: absolute;
    left: 80px;
    width: 200px;
    height: 120px;
    border: 3px solid #8AC007;
}

p {
    margin: 2cm 4cm 3cm 4cm;
}
```

```
p {
    margin-bottom: 2cm;
}

p {
    max-height: 50px;
}

p {
    max-width: 100px;
}

div {
    width: 150px;
    height: 150px;
    overflow: scroll;
}

p {
    padding: 2cm 4cm 3cm 4cm;
}

p {
    padding-bottom: 2cm;
}

h2 {
    position: absolute;
    left: 100px;
    top: 150px;
}

div.absolute {
    position: absolute;
    right: 20px;
    width: 200px;
    height: 120px;
    border: 3px solid #8AC007;
}

div.absolute {
    position: absolute;
    top: 80px;
    width: 200px;
    height: 100px;
    border: 3px solid #8AC007;
}

h2 {
    visibility: hidden;
}

img {
    vertical-align: text-top;
}
```

CSS3 Extra

Background Size

The CSS3 background-size property allows you to specify the size of background images. Before CSS3, the size of a background image was the actual size of the image. CSS3 allows us to re-use background images in different contexts. The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

```
<html>
	<head>
		<style>
		#example1 {
			border: 1px solid black;
			background:url(img_flwr.gif);
			background-repeat: no-repeat;
			padding:15px;
		}

		#example2 {
			border: 1px solid black;
			background:url(img_flwr.gif);
			background-size: 100px 80px;
			background-repeat: no-repeat;
			padding:15px;
		}
		</style>
	</head>
		<body>
		<p>Original background-image:</p>
		<div id="example1">
			<h2>Lorem Ipsum Dolor</h2>
			<p>Lorem ipsum dolor sit amet, consectetuer adipiscing
elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna
aliquam erat volutpat.</p>
			<p>Ut wisi enim ad minim veniam, quis nostrud exerci
tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
		</div>

		<p>Resized background-image:</p>
			<div id="example2">
			<h2>Lorem Ipsum Dolor</h2>
			<p>Lorem ipsum dolor sit amet, consectetuer adipiscing
elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna
aliquam erat volutpat.</p>
			<p>Ut wisi enim ad minim veniam, quis nostrud exerci
tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
		</div>
	</body>
</html>
```

The two other possible values for background-size are contain and cover. The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.

The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area. The following example illustrates the use of contain and cover:

```
<html>
	<head>
		<style>
		.div1 {
			border: 1px solid black;
			height:150px;
			width:180px;
			background:url(img_flwr.gif);
			background-repeat: no-repeat;
		}

		.div2 {
			border: 1px solid black;
			height:150px;
			width:180px;
			background:url(img_flwr.gif);
			background-repeat: no-repeat;
			background-size: contain;
		}

		.div3 {
			border: 1px solid black;
			height:150px;
			width:180px;
			background:url(img_flwr.gif);
			background-repeat: no-repeat;
			background-size: cover;
		}
		</style>
	</head>
	<body>
		<p>Original image:</p>
		<div class="div1">
		<p>Lorem ipsum dolor sit amet.</p>
		</div>

		<p>Using the "contain" keyword:</p>
		<div class="div2">
		<p>Lorem ipsum dolor sit amet.</p>
		</div>

		<p>Using the "cover" keyword:</p>
		<div class="div3">
		<p>Lorem ipsum dolor sit amet.</p>
		</div>
	</body>
</html>
```

Define Sizes of Multiple Background Images

The background-size property also accepts multiple values for background size (using a comma-separated list), when working with multiple backgrounds. The following example has three background images specified, with different background-size value for each image:

```
<html>
	<head>
		<style>
			#example1 {
				background: url(img_flwr.gif) left top no-
repeat, url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left
top repeat;
				padding: 15px;
				background-size: 50px, 130px, auto;
			}
		</style>
	</head>
	<body>
		<div id="example1">
		<h1>Lorem Ipsum Dolor</h1>
		<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
		<p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
		</div>
	</body>
</html>
```

Full Size Background Image

Now we want to have a background image on a website that covers the entire browser window at all times. The requirements are as follows:

- Fill the entire page with the image (no white space)
- Scale image as needed
- Center image on page
- Do not cause scrollbars

The following example shows how to do it; Use the html element (the html element is always at least the height of the browser window). Then set a fixed and centered background on it. Then adjust its size with the background-size property:

```
<html>
	<head>
		<style>
			html {
			  background: url(img_flower.jpg) no-repeat center
fixed;
			  background-size: cover;
			}
			body {
			  color: white;
			}
		</style>
	</head>
	<body>
		<h1>Full Page Background Image</h1>
		<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
```

```
erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
      </body>
</html>
```

Background Origin

The CSS3 background-origin property specifies where the background image is positioned.

The property takes three different values:

- border-box - the background image starts from the upper left corner of the border
- padding-box - (default) the background image starts from the upper left corner of the padding edge
- content-box - the background image starts from the upper left corner of the content

The following example illustrates the background-origin property:

```
<html>
      <head>
            <style>
            #example1 {
                  border: 10px solid black;
                  padding: 35px;
                  background: url(img_flwr.gif);
                  background-repeat: no-repeat;
            }

            #example2 {
                  border: 10px solid black;
                  padding: 35px;
                  background: url(img_flwr.gif);
                  background-repeat: no-repeat;
                  background-origin: border-box;
            }

            #example3 {
                  border: 10px solid black;
                  padding: 35px;
                  background: url(img_flwr.gif);
                  background-repeat: no-repeat;
                  background-origin: content-box;
            }
            </style>
      </head>
      <body>
            <p>No background-origin (padding-box is default):</p>
            <div id="example1">
            <h2>Lorem Ipsum Dolor</h2>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
            <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
            </div>
            <p>background-origin: border-box:</p>
            <div id="example2">
            <h2>Lorem Ipsum Dolor</h2>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
```

```
            <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
            </div>
            <p>background-origin: content-box:</p>
            <div id="example3">
            <h2>Lorem Ipsum Dolor</h2>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
            <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
            </div>
      </body>
</html>
```

Background Clip

The CSS3 background-clip property specifies the painting area of the background.

The property takes three different values:

- border-box - (default) the background is painted to the outside edge of the border
- padding-box - the background is painted to the outside edge of the padding
- content-box - the background is painted within the content box

The following example illustrates the background-clip property:

```
<html>
      <head>
            <style>
                  #example1 {
                        border: 10px dotted black;
                        padding:35px;
                        background: yellow;
                  }

                  #example2 {
                        border: 10px dotted black;
                        padding:35px;
                        background: yellow;
                        background-clip: padding-box;
                  }

                  #example3 {
                        border: 10px dotted black;
                        padding:35px;
                        background: yellow;
                        background-clip: content-box;
                  }
            </style>
      </head>
      <body>
            <p>No background-clip (border-box is default):</p>
            <div id="example1">
            <h2>Lorem Ipsum Dolor</h2>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
            </div>

            <p>background-clip: padding-box:</p>
            <div id="example2">
            <h2>Lorem Ipsum Dolor</h2>
```

```
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
            </div>

            <p>background-clip: content-box:</p>
            <div id="example3">
            <h2>Lorem Ipsum Dolor</h2>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
            </div>
      </body>
</html>
```

Gradients

CSS3 gradients let you display smooth transitions between two or more specified colors.
Earlier, you had to use images for these effects. However, by using CSS3 gradients you can reduce download time and bandwidth usage. In addition, elements with gradients look better when zoomed, because the gradient is generated by the browser.
CSS3 defines two types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)

CSS3 Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

```
background: linear-gradient(direction, color-stop1, color-stop2, ...);
```

Linear Gradient - Top to Bottom (this is default)

```
<html>
      <head>
            <style>
            #grad1 {
                  height: 200px;
                  background: red; /* For browsers that do not support
gradients */
                  background: -webkit-linear-gradient(red, yellow); /*
For Safari 5.1 to 6.0 */
                  background: -o-linear-gradient(red, yellow); /* For
Opera 11.1 to 12.0 */
                  background: -moz-linear-gradient(red, yellow); /* For
Firefox 3.6 to 15 */
                  background: linear-gradient(red, yellow); /* Standard
syntax (must be last) */
            }
            </style>
      </head>
      <body>
            <h3>Linear Gradient - Top to Bottom</h3>
            <p>This linear gradient starts at the top. It starts red,
transitioning to yellow:</p>
            <div id="grad1"></div>
```

```
            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

Linear Gradient - Left to Right

```
<html>
      <head>
            <style>
                  #grad1 {
                        height: 200px;
                        background: red; /* For browsers that do not
support gradients */
                        background: -webkit-linear-gradient(left, red ,
yellow); /* For Safari 5.1 to 6.0 */
                        background: -o-linear-gradient(right, red,
yellow); /* For Opera 11.1 to 12.0 */
                        background: -moz-linear-gradient(right, red,
yellow); /* For Firefox 3.6 to 15 */
                        background: linear-gradient(to right, red ,
yellow); /* Standard syntax (must be last) */
                  }
            </style>
      </head>
      <body>
            <h3>Linear Gradient - Left to Right</h3>
            <p>This linear gradient starts at the left. It starts red,
transitioning to yellow:</p>
            <div id="grad1"></div>
            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

Linear Gradient - Diagonal

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  #grad1 {
                        height: 200px;
                        background: red; /* For browsers that do not
support gradients */
                        background: -webkit-linear-gradient(left top,
red, yellow); /* For Safari 5.1 to 6.0 */
                        background: -o-linear-gradient(bottom right,
red, yellow); /* For Opera 11.1 to 12.0 */
                        background: -moz-linear-gradient(bottom right,
red, yellow); /* For Firefox 3.6 to 15 */
                        background: linear-gradient(to bottom right,
red, yellow); /* Standard syntax (must be last) */
                  }
            </style>
      </head>
      <body>
            <h3>Linear Gradient - Diagonal</h3>
            <p>This linear gradient starts at top left. It starts red,
transitioning to yellow:</p>
            <div id="grad1"></div>
            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

Using Angles

If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.).

background: linear-gradient(*angle*, *color-stop1*, *color-stop2*);

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			#grad1 {
				height: 100px;
				background: red; /* For browsers that do not
support gradients */
				background: -webkit-linear-gradient(0deg, red,
yellow); /* For Safari 5.1 to 6.0 */
				background: -o-linear-gradient(0deg, red,
yellow); /* For Opera 11.1 to 12.0 */
				background: -moz-linear-gradient(0deg, red,
yellow); /* For Firefox 3.6 to 15 */
				background: linear-gradient(0deg, red, yellow);
/* Standard syntax (must be last) */
			}

			#grad2 {
				height: 100px;
				background: red; /* For browsers that do not
support gradients */
				background: -webkit-linear-gradient(90deg, red,
yellow); /* For Safari 5.1 to 6.0 */
				background: -o-linear-gradient(90deg, red,
yellow); /* For Opera 11.1 to 12.0 */
				background: -moz-linear-gradient(90deg, red,
yellow); /* For Firefox 3.6 to 15 */
				background: linear-gradient(90deg, red, yellow);
/* Standard syntax (must be last) */
			}

			#grad3 {
				height: 100px;
				background: red; /* For browsers that do not
support gradients */
				background: -webkit-linear-gradient(180deg, red,
yellow); /* For Safari 5.1 to 6.0 */
				background: -o-linear-gradient(180deg, red,
yellow); /* For Opera 11.1 to 12.0 */
				background: -moz-linear-gradient(180deg, red,
yellow); /* For Firefox 3.6 to 15 */
				background: linear-gradient(180deg, red,
yellow); /* Standard syntax (must be last) */
			}

			#grad4 {
				height: 100px;
				background: red; /* For browsers that do not
support gradients */
				background: -webkit-linear-gradient(-90deg, red,
yellow); /* For Safari 5.1 to 6.0 */
				background: -o-linear-gradient(-90deg, red,
yellow); /* For Opera 11.1 to 12.0 */
```

```
                        background: -moz-linear-gradient(-90deg, red,
yellow); /* For Firefox 3.6 to 15 */
                        background: linear-gradient(-90deg, red,
yellow); /* Standard syntax (must be last) */
                  }
            </style>
      </head>
      <body>
            <h3>Linear Gradients - Using Different Angles</h3>
            <div id="grad1" style="color:white;text-
align:center;">0deg</div><br>
            <div id="grad2" style="color:white;text-
align:center;">90deg</div><br>
            <div id="grad3" style="color:white;text-
align:center;">180deg</div><br>
            <div id="grad4" style="color:white;text-align:center;">-
90deg</div>

            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

Using Multiple Color Stops

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  #grad1 {
                        height: 200px;
                        background: -webkit-linear-gradient(red, yellow,
green); /* For Safari 5.1 to 6.0 */
                        background: -o-linear-gradient(red, yellow,
green); /* For Opera 11.1 to 12.0 */
                        background: -moz-linear-gradient(red, yellow,
green); /* For Firefox 3.6 to 15 */
                        background: linear-gradient(red, yellow, green);
/* Standard syntax (must be last) */
                  }

                  #grad2 {
                        height: 200px;
                        background: -webkit-linear-gradient(red, orange,
yellow, green, blue, indigo, violet); /* For Safari 5.1 to 6.0 */
                        background: -o-linear-gradient(red, orange,
yellow, green, blue, indigo, violet); /* For Opera 11.1 to 12.0 */
                        background: -moz-linear-gradient(red, orange,
yellow, green, blue, indigo, violet); /* For Firefox 3.6 to 15 */
                        background: linear-gradient(red, orange, yellow,
green, blue, indigo, violet); /* Standard syntax (must be last) */
                  }

                  #grad3 {
                        height: 200px;
                        background: -webkit-linear-gradient(red 10%,
green 85%, blue 90%); /* For Safari 5.1 to 6.0 */
                        background: -o-linear-gradient(red 10%, green
85%, blue 90%); /* For Opera 11.1 to 12.0 */
                        background: -moz-linear-gradient(red 10%, green
85%, blue 90%); /* For Firefox 3.6 to 15 */
                        background: linear-gradient(red 10%, green 85%,
blue 90%); /* Standard syntax (must be last) */
```

```
                }
            </style>
      </head>
      <body>
            <h3>3 Color Stops (evenly spaced)</h3>
            <div id="grad1"></div>

            <h3>7 Color Stops (evenly spaced)</h3>
            <div id="grad2"></div>

            <h3>3 Color Stops (not evenly spaced)</h3>
            <div id="grad3"></div>

            <p><strong>Note:</strong> Color stops are automatically
spaced evenly when no percents are specified.</p>
            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

Using Transparency

CSS3 gradients also support transparency, which can be used to create fading effects. To add transparency, we use the rgba() function to define the color stops. The last parameter in the rgba() function can be a value from 0 to 1, and it defines the transparency of the color: 0 indicates full transparency, 1 indicates full color (no transparency).

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  #grad1 {
                        height: 200px;
                        background: -webkit-linear-gradient(left,
rgba(255,0,0,0), rgba(255,0,0,1)); /* For Safari 5.1 to 6.0 */
                        background: -o-linear-gradient(right,
rgba(255,0,0,0), rgba(255,0,0,1)); /* For Opera 11.1 to 12.0 */
                        background: -moz-linear-gradient(right,
rgba(255,0,0,0), rgba(255,0,0,1)); /* For Firefox 3.6 to 15 */
                        background: linear-gradient(to right,
rgba(255,0,0,0), rgba(255,0,0,1)); /* Standard syntax (must be last) */
                  }
            </style>
      </head>
      <body>
            <h3>Linear Gradient - Transparency</h3>
            <p>To add transparency, we use the rgba() function to define
the color stops. The last parameter in the rgba() function can be a
value from 0 to 1, and it defines the transparency of the color: 0
indicates full transparency, 1 indicates full color (no
transparency).</p>
            <div id="grad1"></div>
            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

Repeating a linear-gradient

The repeating-linear-gradient() function is used to repeat linear gradients:

```
<!DOCTYPE html>
```

```
<html>
	<head>
	<style>
	#grad1 {
		height: 200px;
		background: -webkit-repeating-linear-gradient(red, yellow
10%, green 20%); /* For Safari 5.1 to 6.0 */
		background: -o-repeating-linear-gradient(red, yellow 10%,
green 20%); /* For Opera 11.1 to 12.0 */
		background: -moz-repeating-linear-gradient(red, yellow 10%,
green 20%); /* For Firefox 3.6 to 15 */
		background: repeating-linear-gradient(red, yellow 10%, green
20%); /* Standard syntax (must be last) */
	}

	#grad2 {
		height: 200px;
		background: -webkit-repeating-linear-
gradient(45deg,red,yellow 7%,green 10%); /* For Safari 5.1 to 6.0 */
		background: -o-repeating-linear-gradient(45deg,red,yellow
7%,green 10%); /* For Opera 11.1 to 12.0 */
		background: -moz-repeating-linear-gradient(45deg,red,yellow
7%,green 10%); /* For Firefox 3.6 to 15 */
		background: repeating-linear-gradient(45deg,red,yellow
7%,green 10%); /* Standard syntax (must be last) */
	}

	#grad3 {
		height: 200px;
		background: -webkit-repeating-linear-
gradient(190deg,red,yellow 7%,green 10%); /* For Safari 5.1 to 6.0 */
		background: -o-repeating-linear-gradient(190deg,red,yellow
7%,green 10%); /* For Opera 11.1 to 12.0 */
		background: -moz-repeating-linear-gradient(190deg,red,yellow
7%,green 10%); /* For Firefox 3.6 to 15 */
		background: repeating-linear-gradient(190deg,red,yellow
7%,green 10%); /* Standard syntax (must be last) */
	}

	#grad4 {
		height: 200px;
		background: -webkit-repeating-linear-
gradient(90deg,red,yellow 7%,green 10%); /* For Safari 5.1 to 6.0 */
		background: -o-repeating-linear-gradient(); /* For Opera
11.1 to 12.0 */
		background: -moz-repeating-linear-gradient(90deg,red,yellow
7%,green 10%); /* For Firefox 3.6 to 15 */
		background: repeating-linear-gradient(90deg,red,yellow
7%,green 10%); /* Standard syntax (must be last) */
	}
	</style>
	</head>
	<body>
		<h3>Repeating Linear Gradient</h3>
		<div id="grad1"></div>
		<p>A repeating gradient on 45deg axe starting red and
finishing green:</p>
		<div id="grad2"></div>
		<p>A repeating gradient on 190deg axe starting red and
finishing green:</p>
		<div id="grad3"></div>
		<p>A repeating gradient on 90deg axe starting red and
finishing green:</p>
```

```
            <div id="grad4"></div>
            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

CSS3 Radial Gradients

background: radial-gradient(*shape size* at *position, start-color, ..., last-color*);

By default, shape is ellipse, size is farthest-corner, and position is center.

Radial Gradient - Evenly Spaced Color Stops (this is default)

```
<!DOCTYPE html>
<html>
      <head>
            <style>
            #grad1 {
                  height: 150px;
                  width: 200px;
                  background: red; /* For browsers that do not support
gradients */
                  background: -webkit-radial-gradient(red, yellow,
green); /* Safari 5.1 to 6.0 */
                  background: -o-radial-gradient(red, yellow, green); /*
For Opera 11.6 to 12.0 */
                  background: -moz-radial-gradient(red, yellow, green);
/* For Firefox 3.6 to 15 */
                  background: radial-gradient(red, yellow, green); /*
Standard syntax (must be last) */
            }
            </style>
      </head>
      <body>
            <h3>Radial Gradient - Evenly Spaced Color Stops</h3>
            <div id="grad1"></div>
            <p><strong>Note:</strong> Internet Explorer 9 and earlier
versions do not support gradients.</p>
      </body>
</html>
```

Radial Gradient - Differently Spaced Color Stops

The following example shows a radial gradient with differently spaced color stops:

```
<!DOCTYPE html>
<html>
            <head>
                  <style>
                        #grad1 {
                              height: 150px;
                              width: 200px;
                              background: red; /* For browsers that do
not support gradients */
                              background: -webkit-radial-gradient(red
5%, yellow 15%, green 60%); /* Safari 5.1-6.0 */
                              background: -o-radial-gradient(red 5%,
yellow 15%, green 60%); /* For Opera 11.6 to 12.0 */
                              background: -moz-radial-gradient(red 5%,
yellow 15%, green 60%); /* For Firefox 3.6 to 15 */
                              background: radial-gradient(red 5%, yellow
15%, green 60%); /* Standard syntax (must be last) */
                        }
                  </style>
            </head>
```

```
	<body>
		<h3>Radial Gradient - Differently Spaced Color Stops</h3>
		<div id="grad1"></div>
	</body>
</html>
```

Set Shape

The shape parameter defines the shape. It can take the value circle or ellipse. The default value is ellipse. The following example shows a radial gradient with the shape of a circle:

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			#grad1 {
				height: 150px;
				width: 200px;
				background: -webkit-radial-gradient(red, yellow,
green); /* For Safari 5.1 to 6.0 */
				background: -o-radial-gradient(red, yellow,
green); /* For Opera 11.6 to 12.0 */
				background: -moz-radial-gradient(red, yellow,
green); /* For Fx 3.6 to 15 */
				background: radial-gradient(red, yellow, green);
/* Standard syntax (must be last) */
			}

			#grad2 {
				height: 150px;
				width: 200px;
				background: -webkit-radial-gradient(circle, red,
yellow, green); /* For Safari 5.1 to 6.0 */
				background: -o-radial-gradient(circle, red,
yellow, green); /* For Opera 11.6 to 12.0 */
				background: -moz-radial-gradient(circle, red,
yellow, green); /* For Fx 3.6 to 15 */
				background: radial-gradient(circle, red, yellow,
green); /* Standard syntax (must be last) */
			}
		</style>
	</head>
	<body>
		<h3>Radial Gradient - Shapes</h3>
		<p><strong>Ellipse (this is default):</strong></p>
		<div id="grad1"></div>
		<p><strong>Circle:</strong></p>
		<div id="grad2"></div>
	</body>
</html>
```

Use of Different Size Keywords

The size parameter defines the size of the gradient. It can take four values:

- closest-side
- farthest-side
- closest-corner
- farthest-corner

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			#grad1 {
```

```
                        height: 150px;
                        width: 150px;
                        background: -webkit-radial-gradient(60% 55%,
closest-side, red, yellow, black); /* Safari 5.1 to 6.0 */
                        background: -o-radial-gradient(60% 55%, closest-
side, red, yellow, black); /* For Opera 11.6 to 12.0 */
                        background: -moz-radial-gradient(60% 55%,
closest-side, red, yellow, black); /* For Firefox 3.6 to 15 */
                        background: radial-gradient(closest-side at 60%
55%, red, yellow, black); /* Standard syntax (must be last) */
                  }

                  #grad2 {
                        height: 150px;
                        width: 150px;
                        background: -webkit-radial-gradient(60% 55%,
farthest-side, red, yellow, black); /* Safari 5.1 to 6.0 */
                        background: -o-radial-gradient(60% 55%,
farthest-side, red, yellow, black); /* For Opera 11.6 to 12.0 */
                        background: -moz-radial-gradient(60% 55%,
farthest-side, red, yellow, black); /* For Firefox 3.6 to 15 */
                        background: radial-gradient(farthest-side at 60%
55%, red, yellow, black); /* Standard syntax (must be last) */
                  }

                  #grad3 {
                        height: 150px;
                        width: 150px;
                        background: -webkit-radial-gradient(60% 55%,
closest-corner, red, yellow, black); /* Safari 5.1 to 6.0 */
                        background: -o-radial-gradient(60% 55%, closest-
corner, red, yellow, black); /* For Opera 11.6 to 12.0 */
                        background: -moz-radial-gradient(60% 55%,
closest-corner, red, yellow, black); /* For Firefox 3.6 to 15 */
                        background: radial-gradient(closest-corner at
60% 55%, red, yellow, black); /* Standard syntax (must be last) */
                  }

                  #grad4 {
                        height: 150px;
                        width: 150px;
                        background: -webkit-radial-gradient(60% 55%,
farthest-corner, red, yellow, black); /* Safari 5.1 to 6.0 */
                        background: -o-radial-gradient(60% 55%,
farthest-corner, red, yellow, black); /* For Opera 11.6 to 12.0 */
                        background: -moz-radial-gradient(60% 55%,
farthest-corner, red, yellow, black); /* For Firefox 3.6 to 15 */
                        background: radial-gradient(farthest-corner at
60% 55%, red, yellow, black); /* Standard syntax (must be last) */
                  }
            </style>
      </head>
      <body>
            <h3>Radial Gradients - Use of different size keywords</h3>
            <p><strong>closest-side:</strong></p>
            <div id="grad1"></div>
            <p><strong>farthest-side:</strong></p>
            <div id="grad2"></div>
            <p><strong>closest-corner:</strong></p>
            <div id="grad3"></div>
            <p><strong>farthest-corner (this is default):</strong></p>
            <div id="grad4"></div>
      </body>
```

```
</html>
```

Repeating a radial-gradient

The repeating-radial-gradient() function is used to repeat radial gradients:

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			#grad1 {
				height: 150px;
				width: 200px;
				background: -webkit-repeating-radial-
gradient(red, yellow 10%, green 15%); /* For Safari 5.1 to 6.0 */
				background: -o-repeating-radial-gradient(red,
yellow 10%, green 15%); /* For Opera 11.6 to 12.0 */
				background: -moz-repeating-radial-gradient(red,
yellow 10%, green 15%); /* For Firefox 3.6 to 15 */
				background: repeating-radial-gradient(red,
yellow 10%, green 15%); /* Standard syntax (must be last) */
			}
		</style>
	</head>
	<body>
	<h3>Repeating Radial Gradient</h3>
	<div id="grad1"></div>
	<p><strong>Note:</strong> Internet Explorer 9 and earlier versions
do not support gradients.</p>
	</body>
</html>
```

Box Shadow

The box-shadow property attaches one or more shadows to an element.

Default value:	**none**
Inherited:	no
Animatable:	yes.
Version:	CSS3
JavaScript syntax:	object.style.boxShadow="10px 20px 30px blue"

box-shadow: none|h-shadow v-shadow blur spread color | inset| initial| inherit;

Note: The box-shadow property attaches one or more shadows to an element. The property is a comma-separated list of shadows, each specified by 2-4 length values, an optional color, and an optional inset keyword. Omitted lengths are 0.

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			div {
				width: 300px;
				height: 100px;
				background-color: yellow;
				box-shadow: 10px 10px 5px #888888;
			}
		</style>
	</head>
	<body>
	<div></div>
	</body>
```

```
</html>
```

Transform

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements. To better understand the transform property.

Default value:	**none**
Inherited:	no
Animatable:	yes.
Version:	CSS3
JavaScript syntax:	object.style.transform="rotate(7deg)"

`transform: none|transform-functions|initial|inherit;`

Value	**Description**
none	Defines that there should be no transformation
matrix(n,n,n,n,n,n)	Defines a 2D transformation, using a matrix of six values
matrix3d (n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	Defines a 3D transformation, using a 4x4 matrix of 16 values
translate(x,y)	Defines a 2D translation
translate3d(x,y,z)	Defines a 3D translation
translateX(x)	Defines a translation, using only the value for the X-axis
translateY(y)	Defines a translation, using only the value for the Y-axis
translateZ(z)	Defines a 3D translation, using only the value for the Z-axis
scale(x,y)	Defines a 2D scale transformation
scale3d(x,y,z)	Defines a 3D scale transformation
scaleX(x)	Defines a scale transformation by giving a value for the X-axis
scaleY(y)	Defines a scale transformation by giving a value for the Y-axis
scaleZ(z)	Defines a 3D scale transformation by giving a value for the Z-axis
rotate(angle)	Defines a 2D rotation, the angle is specified in the parameter
rotate3d(x,y,z,angle)	Defines a 3D rotation
rotateX(angle)	Defines a 3D rotation along the X-axis
rotateY(angle)	Defines a 3D rotation along the Y-axis
rotateZ(angle)	Defines a 3D rotation along the Z-axis
skew(x-angle,y-angle)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(angle)	Defines a 2D skew transformation along the X-axis
skewY(angle)	Defines a 2D skew transformation along the Y-axis
perspective(n)	Defines a perspective view for a 3D transformed element
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			div {
				width: 200px;
				height: 100px;
				background-color: yellow;
				/* Rotate div */
				-ms-transform: rotate(7deg); /* IE 9 */
				-webkit-transform: rotate(7deg); /* Chrome,
Safari, Opera */
				transform: rotate(7deg);
```

```
                }
            </style>
      </head>
      <body>
            <div>Hello</div>
            <br>

            <p><b>Note:</b> Internet Explorer 8 and earlier versions do
not support the transform property.</p>

            <p><b>Note:</b> Internet Explorer 9 supports an alternative,
the -ms-transform property. Newer versions of IE support the transform
property (do not need the ms prefix).</p>

            <p><b>Note:</b> Chrome, Safari and Opera supports an
alternative, the -webkit-transform property.</p>
      </body>
</html>
```

Transitions

CSS3 transitions allows you to change property values smoothly (from one value to another), over a given duration. To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The following example shows a 100px * 100px red element. The element has also specified a transition effect for the width property, with a duration of 2 seconds:

The transition effect will start when the specified CSS property (width) changes value.

Property	Description
transition	A shorthand property for setting the four transition properties into a single property
transition-delay	Specifies a delay (in seconds) for the transition effect
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-property	Specifies the name of the CSS property the transition effect is for
transition-timing-function	Specifies the speed curve of the transition effect

Now, let us specify a new value for the width property when a user mouses over the element:

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  div {
                        width: 100px;
                        height: 100px;
                        background: red;
                        -webkit-transition: width 2s; /* For Safari 3.1
to 6.0 */
                        transition: width 2s;
                  }
```

```
                  div:hover {
                        width: 300px;
                  }
            </style>
      </head>
      <body>
      <div></div>
      <p>Hover over the div element above, to see the transition
effect.</p>
      </body>
</html>
```

Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  div {
                        width: 100px;
                        height: 100px;
                        background: red;
                        -webkit-transition: width 2s, height 4s; /* For
Safari 3.1 to 6.0 */
                        transition: width 2s, height 4s;
                  }

                  div:hover {
                        width: 300px;
                        height: 300px;
                  }
            </style>
      </head>
      <body>
            <div></div>
            <p>Hover over the div element above, to see the transition
effect.</p>
      </body>
</html>
```

Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect.
The transition-timing-function property can have the following values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  div {
                        width: 100px;
                        height: 100px;
```

```
                        background: red;
                        -webkit-transition: width 2s; /* Safari */
                        transition: width 2s;
                  }

                  /* For Safari 3.1 to 6.0 */
                  #div1 {-webkit-transition-timing-function: linear;}
                  #div2 {-webkit-transition-timing-function: ease;}
                  #div3 {-webkit-transition-timing-function: ease-in;}
                  #div4 {-webkit-transition-timing-function: ease-out;}
                  #div5 {-webkit-transition-timing-function: ease-in-
out;}

                  /* Standard syntax */
                  #div1 {transition-timing-function: linear;}
                  #div2 {transition-timing-function: ease;}
                  #div3 {transition-timing-function: ease-in;}
                  #div4 {transition-timing-function: ease-out;}
                  #div5 {transition-timing-function: ease-in-out;}

                  div:hover {
                        width: 300px;
                  }
            </style>
      </head>
      <body>
            <div id="div1">linear</div><br>
            <div id="div2">ease</div><br>
            <div id="div3">ease-in</div><br>
            <div id="div4">ease-out</div><br>
            <div id="div5">ease-in-out</div><br>
            <p>Hover over the div elements above, to see the different
speed curves.</p>
      </body>
</html>
```

Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition effect. The following example has a 1 second delay before starting:

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  div {
                        width: 100px;
                        height: 100px;
                        background: red;
                        -webkit-transition: width 3s; /* Safari */
                        -webkit-transition-delay: 1s; /* Safari */
                        transition: width 3s;
                        transition-delay: 1s;
                  }

                  div:hover {
                        width: 300px;
                  }
            </style>
      </head>
      <body>
            <div></div>
            <p>Hover over the div element above, to see the transition
effect.</p>
```

```
            <p><b>Note:</b> The transition effect has a 1 second delay
before starting.</p>
      </body>
</html>
```

Transition + Transformation

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  div {
                        width: 100px;
                        height: 100px;
                        background: red;
                        -webkit-transition: width 2s, height 2s, -
webkit-transform 2s; /* Safari */
                        transition: width 2s, height 2s, transform 2s;
                  }

                  div:hover {
                        width: 300px;
                        height: 300px;
                        -webkit-transform: rotate(180deg); /* Safari */
                        transform: rotate(180deg);
                  }
            </style>
      </head>
      <body>
            <div></div>
            <p>Hover over the div element above, to see the transition
effect.</p>
      </body>
</html>
```

Box Sizing

The CSS3 box-sizing property allows us to include the padding and border in an element's total width and height. *Without the CSS3 box-sizing Property;* By default, the width and height of an element is calculated like this:

```
width + padding + border = actual width of an element
height + padding + border = actual height of an element
```

This means: When you set the width/height of an element, the element often appear bigger than you have set (because the element's border and padding are added to the element's specified width/height).

The two elements above end up with different sizes in the result (because div2 has a padding specified):

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  .div1 {
                        width: 300px;
                        height: 100px;
                        border: 1px solid blue;
                  }

                  .div2 {
                        width: 300px;
```

```
                        height: 100px;
                        padding: 50px;
                        border: 1px solid red;
                  }
            </style>
      </head>
            <body>
            <div class="div1">This div is smaller (width is 300px and
height is 100px).</div>
            <br>
            <div class="div2">This div is bigger (width is also 300px
and height is 100px).</div>
      </body>
</html>
```

So, for a long time web developers have specified a smaller width value than they wanted, because they had to subtract out the padding and borders.
With CSS3, the box-sizing property solves this problem.

The CSS3 box-sizing property allows us to include the padding and border in an element's total width and height.
If you set box-sizing: border-box; on an element padding and border are included in the width and height:
Here is the same example as above, with box-sizing: border-box; added to both elements:

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  .div1 {
                        width: 300px;
                        height: 100px;
                        border: 1px solid blue;
                        box-sizing: border-box;
                  }

                  .div2 {
                        width: 300px;
                        height: 100px;
                        padding: 50px;
                        border: 1px solid red;
                        box-sizing: border-box;
                  }
            </style>
      </head>
      <body>
      <div class="div1">Both divs are the same size now!</div> <br>
      <div class="div2">Hooray!</div>
      </body>
</html>
```

Since the result of using the box-sizing: border-box; is so much better, many developers want all elements on their pages to work this way.

The code below ensures that all elements are sized in this more intuitive way. Many browsers already use box-sizing: border-box; for many form elements (but not all - which is why inputs and text areas look different at width: 100%;).

Applying this to all elements is safe and wise:

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			* {
				box-sizing: border-box;
			}

			input, textarea {
				width: 100%;
			}
		</style>
	</head>
	<body>
		<form action="action_page.html">
			First name:<br>
			<input type="text" name="firstname"
value="Mickey"><br>
			Last name:<br>
			<input type="text" name="lastname" value="Mouse"><br>
			Comments:<br>
			<textarea name="message" rows="5" cols="30">
			</textarea>
			<br><br>
			<input type="submit" value="Submit">
		</form>
		<p><strong>Tip:</strong> Try to remove the box-sizing
property from the style element and look what happens.
		Notice that the input, textarea, and submit button will no
longer have equal widths.</p>
	</body>
</html>
```

Position

The position property specifies the type of positioning method used for an element (static, relative, absolute or fixed).

Default value:	**static**
Inherited:	no
Animatable:	no. Read about *animatable*
Version:	CSS2
JavaScript syntax:	object.style.position="absolute"

`position: static|absolute|fixed|relative|initial|inherit;`

Value	**Description**
static	Default value. Elements render in order, as they appear in the document flow
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element. Read about *inherit*

```
<!DOCTYPE html>
<html>
	<head>
		<style>
```

```
                  h2 {
                        position: absolute;
                        left: 100px;
                        top: 150px;
                  }
            </style>
      </head>
      <body>
            <h2>This is a heading with an absolute position</h2>
            <p>With absolute positioning, an element can be placed
anywhere on a page. The heading below is placed 100px from the left of
the page and 150px from the top of the page.</p>
      </body>
</html>
```

Media Queries

Media queries in CSS3 extend the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to tablets, iPhone, and Androids.

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions) {
    CSS-Code;
}
```

The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.

Unless you use the not or only operators, the media type is optional and the all type will be implied.

You can also have different stylesheets for different media:

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)"
href="print.css">
```

Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  body {
                        background-color: pink;
```

```
                  }

                  @media screen and (min-width: 480px) {
                        body {
                              background-color: lightgreen;
                        }
                  }
            </style>
      </head>
      <body>
            <h1>Resize the browser window to see the effect!</h1>
            <p>The media query will only apply if the media type is
screen and the viewport is 480px wide or wider.</p>
      </body>
</html>
```

```
<!DOCTYPE html>
<html>
      <head>
            <meta name="viewport" content="width=device-width, initial-
scale=1.0">
            <style>
                  .wrapper {overflow:auto;}
                  #main {margin-left: 4px;}
                  #leftsidebar {float: none;width: auto;}
                  #menulist {margin:0;padding:0;}
                  .menuitem {
                        background:#CDF0F6;
                        border:1px solid #d4d4d4;
                        border-radius:4px;
                        list-style-type:none;
                        margin:4px;
                        padding:2px;
                  }
                  @media screen and (min-width: 480px) {
                        #leftsidebar {width:200px;float:left;}
                        #main {margin-left:216px;}
                  }
            </style>
      </head>
      <body>
            <div class="wrapper">
              <div id="leftsidebar">
                  <ul id="menulist">
                    <li class="menuitem">Menu-item 1</li>
                    <li class="menuitem">Menu-item 2</li>
                    <li class="menuitem">Menu-item 3</li>
                    <li class="menuitem">Menu-item 4</li>
                    <li class="menuitem">Menu-item 5</li>
                 </ul>
              </div>
              <div id="main">
                  <h1>Resize the browser window to see the effect!</h1>
                  <p>This example shows a menu that will float to the
left of the page if the viewport is 480 pixels wide or wider. If the
viewport is less than 480 pixels, the menu will be on top of the
content.</p>
              </div>
            </div>
      </body>
</html>
```

Unit 5
JavaScript

Introduction

JavaScript is an object scripting language used on the World Wide Consortium. It is used to create online application which links objects and resources from client and servers. JavaScript is used to create dynamic web pages. JavaScript was originated by Netscape, which was known as LiveScript. In December 1995 Sun Microsystems took over LiveScript and it was renamed JavaScript. In 1996 Microsoft launched Internet Explorer, which was JavaScript enabled. JavaScript Resembles C, Java syntax but does not support all of its functionality.

JavaScript Is

- Language used for scripting.
- Used to create network-centric applications.
- A Lightweight programming language.
- Lines of executable computer code.
- To be inserted into a HTML page.
- An open & cross platform application.

How Does It Work

When a JavaScript is inserted into an HTML document, the Internet browser will read the HTML and interpret the JavaScript. JavaScript can be executed immediately, or at a later event.
Scripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

What Can a JavaScript Do

- Gives HTML designers a programming tool
- Can put dynamic text into an HTML page
- Can react to events
- Can read and write HTML elements
- Can be used to validate data

To insert a script in an HTML document, use the <script> tag. Use the language attribute to define the scripting language.

```
<script language="JavaScript">
</script>
```

A Simple JavaScript

```
<html>
      <body>
            <script language="JavaScript">
            document.write("Hello World!")
            </script>
      </body>
</html>
```

Ending Statements With A Semicolon.

With the traditional programming languages C++ and Java, each code statement has to end with a semicolon. Many programmers continue this habit when writing JavaScript, but in general,

semicolons are optional and are required only if you want to put more than one statement on a single line.

Handling Older Browsers

Older browsers that do not support scripts will display the script as page content. To prevent them from doing this, you can use the HTML comment tag:

```
<script language="JavaScript">
      <!-- some statements //-->
</script>
```

Note that you can not put // in front of the first comment line (like //<!--), because older browser will display it

Comments

The two forward slashes (//) are a JavaScript comment symbol, and prevent the JavaScript from trying to interpret the line. Forward slashes are used to put a single line comment. Forward slash and asterisk sign is used to put double line or multiple line comment

```
<script language="JavaScript">
      // some comment but for single line
      /* put here some
            Multiline comment */
</script>
```

Scripts in The Head Section

When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
<html>
      <head>
            <script language="JavaScript">
                document.write("Head section script")
            </script>
      </head>
</html>
```

Scripts In The Body Section.

When you place a script in the body section it generates the content of the page.

```
<html>
      <body>
            <script language="JavaScript">
                  document.write("Body section script")
            </script>
      </body>
</html>
```

Scripts in both the Body and the Head Section.

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

```
<html>
      <head>
            <script language="JavaScript">
                  document.write("Head section script")
            </script>
      </head>
```

```
      <body>
            <script language="JavaScript">
               document.write("Body section script")
            </script>
      </body>
</html>
```

External Script

Sometimes you might want to run the same script on several pages, without writing the script on each and every page. To simplify this you can write the script in an external file, and save it with a .js file extension.

```
/*External script */
document.write("The script is external")

/* Html Web Page */
<html>
      <body>
            <script src="Ext.js">
            </script>
      </body>
</html>
```

Save the external script and the Html web page in the same directory with any name. In the Html web page 'Ext.js' should be replaced with the name of the file external script by which it is saved.

Variables

Variables are used to store data. A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.

Rules for Variable names:

- Variable names are case sensitive
- They must begin with a letter or the underscore character
- Character, digits and underscore is allowed

Declaring Variables

- You can create a variable with the var statement:

  var strname = "some value"

  var no = 1
- You can also create a variable without the var statement:

  strname = "some value"
- You assign a value to a variable like this:

  var strname = "Hege" or strname = "Hege"

The variable name is on the left side of the expression and the value you want to assign to the variable is on the right.

Scope of variable

When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different function, because each is recognized by the function in which it is declared. If you declare a variable outside a function; all

the function on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

```
<html>
	<body>
		<script language="JavaScript">
			var intNo = 1
			var floatNo = 1.1
			var string = "Hello World"
			var date = Date();
			document.write (intNo + "<br>")
			document.write (floatNo+ "<br>")
			document.write (string+ "<br>")
			document.write (date+ "<br>")
		</script>
	</body>
</html>
```

Data Types In JavaScript

JavaScript primitive data types

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

JavaScript Global Functions

parseInt()

- The parseInt() function parses a string and returns an integer.
- The radix parameter is used to specify which numeral system to be used, for example, a radix of 16 (hexadecimal) indicates that the number in the string should be parsed from a hexadecimal number to a decimal number.
  - If the radix parameter is omitted, JavaScript assumes the following:
  - If the string begins with "0x", the radix is 16 (hexadecimal)
  - If the string begins with "0", the radix is 8 (octal). This feature is deprecated
  - If the string begins with any other value, the radix is 10 (decimal)
- **Note:** Only the first number in the string is returned!
- **Note:** Leading and trailing spaces are allowed.
- **Note:** If the first character cannot be converted to a number, parseInt() returns NaN.
- **Note:** Older browsers will result parseInt("010") as 8, because older versions of ECMAScript, (older than ECMAScript 5, uses the octal radix (8) as default when the string begins with "0". As of ECMAScript 5, the default is the decimal radix (10).

```
<!DOCTYPE html>
<html>
	<body>

		<p>Click the button to parse different strings.</p>
		<button onclick="myFunction()">Try it</button>
		<p id="demo"></p>
		<script>
		function myFunction() {
			var a = parseInt("10") + "<br>";
			var b = parseInt("10.00") + "<br>";
			var c = parseInt("10.33") + "<br>";
			var d = parseInt("34 45 66") + "<br>";
			var e = parseInt("   60   ") + "<br>";
			var f = parseInt("40 years") + "<br>";
			var g = parseInt("He was 40") + "<br>";

			var h = parseInt("10", 10)+ "<br>";
			var i = parseInt("010")+ "<br>";
			var j = parseInt("10", 8)+ "<br>";
			var k = parseInt("0x10")+ "<br>";
			var l = parseInt("10", 16)+ "<br>";

			var n = a + b + c + d + e + f + g + "<br>" + h + i + j
+ k +l;
			document.getElementById("demo").innerHTML = n;
		}
		</script>

	</body>
</html>
```

parseFloat()

- The parseFloat() function parses a string and returns a floating point number.
- This function determines if the first character in the specified string is a number. If it is, it parses the string until it reaches the end of the number, and returns the number as a number, not as a string.
- **Note:** Only the first number in the string is returned!
- **Note:** Leading and trailing spaces are allowed.
- **Note:** If the first character cannot be converted to a number, parseFloat() returns NaN.

```
<!DOCTYPE html>
<html>
	<body>
		<p>Click the button to parse different strings.</p>
		<button onclick="myFunction()">Try it</button>
		<p id="demo"></p>
		<script>
		function myFunction() {
			var a = parseFloat("10") + "<br>";
			var b = parseFloat("10.00") + "<br>";
			var c = parseFloat("10.33") + "<br>";
			var d = parseFloat("34 45 66") + "<br>";
			var e = parseFloat("   60   ") + "<br>";
			var f = parseFloat("40 years") + "<br>";
			var g = parseFloat("He was 40") + "<br>";

			var n = a + b + c + d + e + f + g;
			document.getElementById("demo").innerHTML = n;
```

```
            }
            </script>
      </body>
</html>
```

eval()

- The eval() function evaluates or executes an argument.
- If the argument is an expression, eval() evaluates the expression. If the argument is one or more JavaScript statements, eval() executes the statements.

```
<!DOCTYPE html>
<html>
      <body>
            <p>Click the button to evaluate/execute JavaScript
code/expressions.</p>
            <button onclick="myFunction()">Try it</button>
            <p id="demo"></p>
            <script>
            function myFunction() {
                  var x = 10;
                  var y = 20;
                  var a = eval("x * y") + "<br>";
                  var b = eval("2 + 2") + "<br>";
                  var c = eval("x + 17") + "<br>";

                  var res = a + b + c;
                  document.getElementById("demo").innerHTML = res;
            }
            </script>

      </body>
</html>
```

Expression

The entire combination of the operator(s) and the associated operand(s) is called an Expression.

Operands

The data used by the operator to perform the defined operator are called as the operands.

Operator

An Operator is a simulated transformer, which is used to transform one, or more than one value into a single resultant.

Operator category

- Unary
- Binary
- Ternary
- Special operators.

Unary operators

The operators, which have only one operand and one operator, are called as unary operator.

- -5
- +5

Binary Operators

The operators, which have two operands, and one operator are called as unary operator. They are again categorized into the following.

- Arithmetic operators.
- Logical operators.
- Relational operators
- Comparison operators.
- Assignment operators.
- Bit Manipulation Operators

Operators

Arithmetic operators

Operator	Description	Effect	Example	Result
+	Addition	Sum of variable or values	x=2 x+=2	4
-	Subtraction	Subtraction of variable or values	x=2 5-x	3
*	Multiplication	Multiplication of variable or values	x=4 x*5	20
/	Division	Division of variable or values	15/5 5/2	3 2.5
%	Modulus	Reminder of first variable or value by second	5%2 10%8	1 2 0
++	Increment	(Post) Return and increment. (Pre) Increment and return	x=5 x++ ++x	6 7
--	Decrement	(Post) Return and decrement. (Pre) Decrement and return	x=5 x-- --x	4 3

Logical Operator

Operator	Description	Effect	Example	Result
&&	And	True if both are true	x=6 y=3 (x < 10 && y > 1)	True
||	Or	True if both or any of it true	x=6 y=3 (x==5 || y==5)	False
!	Not	True if false. False if true	x=6 y=3 !(x==y)	True

String Operator

Operator	Description	Effect	Example	Result
+	Plus	Merge two string	x="A" y="B" x += y	"AB"

Relational Operator

Operator	Description	Effect	Example	Result
==	is equal to	True if both value are equal	5=="5"	True
===	strictly equal	True if both values and data type are equal	"5"===5	False
!=	is not equal	True if both are not equal	5!=8	True
>	is greater than	True if first is greater than second	5>8	False
<	is less than	True if first is less then second	5<8	True
>=	is greater than or equal to	True if first is greater than or equal to second	5>=8	False
<=	is less than or equal to	True if first is less than or equal to second	5<=8	True

Assignment Operator

Operator	Description	Effect	Example	Result
=	Assign	Transfer the value	x=y	x=y
+=	Sum	Add and transfer	x+=y	x=x+y
-=	Subtract	Subtract and transfer	x-=y	x=x-y
*=	Multiply	Multiply and transfer	x*=y	x=x*y
/=	Divide	Divide and transfer	x/=y	x=x/y
%=	Modulus	Modulus and transfer	x%=y	x=x%y

Conditional – Ternary Operator

Operator	Description	Effect	Example		Result
?:	Same as if	True and False part in single line	a=5 b=4	max = (a>b)?a:b	Value a

Control Structure (Selection Control) Or (Sequence Control)

If Statement

If statement is used when a part of statement is to be executed or when not to be executed. When the condition following 'if' is true then it will executed the true part. If the condition is false then it executed the else part.

```
if(condition)
    do this;
```

```
if(condition)
{
    do this;
    and this;
}
```

```
if(condition)
{
   if(conditon)
   {
      do this;
      and this;
    }
    also do this;
}
else
   do this;
```

```
if(condition)
   do this;
else
   do this;
```

```
if(condition)
{
   do this;
   and this;
}
else
   do this;
```

```
<html>
      <body>
            <script language="javascript">
                  a=5;  b=6;
                  if(a > b)
                     document.write("a is bigger than b");
                  else
                     document.write("a is less than b");
            </script>
      </body>
</html>
```

Switch

The Switch statement in JavaScript is used to perform one of several different actions based on one of several different conditions. We may compare the same variable with different values and if we want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code. If u don't put a break after each case than the statement in the next statement will be evaluated.

```
switch(expression)
{
    case :
        do this and
exit
}
```

```
switch(expression)
{
    case :
    case :
        do this and
exit
    case :
    case :
        do this and
exit
}
```

```
switch(expression)
{
    case :
        do this and
exit
    case :
        do this and
exit
    case :
        do this and
exit
    default:
        do this
}
```

```
<html>
	<body>
		<script language="javascript">
		i=2;
			switch(i){
				case 0:
				   document.write("i equals 0");
				   break;
				case 1:
				   document.write("i equals 1");
				   break;
				case 2:
				   document.write("i equals 2");
			}
		</script>
	</body>
</html>
```

Control Structure (Iteration Control)

While loop

Loop statements in JAVASCRIPT are used to execute the same block of code a specified number of times. It tells JAVASCRIPT to execute the nested statement(s) repeatedly, as long as the while expression evaluates to TRUE. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time JAVASCRIPT runs the statements in the loop is one iteration). Sometimes, if the while expression evaluates to FALSE from the very beginning, the nested statement(s) won't even be run once.

```
while(expression)
    do this;
```

```
while(expression)
{
    do this;    and this;
}
```

```
while(expression)
{
   do this;
   while(expression)
   {
       do this;
       and this;
   }
}
```

```
<html>
	<body>
		<script language="javascript">
			i=1;
			while(i<=5){
				document.write("The number is ");
				document.write(i+"<br>");
				i++;
			}
		</script>
	</body>
</html>
```

Do- while loop

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true. There is just one syntax for do-while loops.

```
do this;
       and this;
}while(expresion);
```

```
do
{
       do this;       and this;
      do
      {
            do this
       }while(expresion);
}while(expresion);
```

```
<html>
      <body>
            <script language="javascript">
                  i = 5;
                  do{
                        document.write(i);i--;
                  } while (i > 0);
            </script>
      </body>
</html>
```

For Loop

'For statement is used when you know how many times you want to execute a statement or a list of statements. 'For' statement have three parameters. The first parameter initializes variables, the second parameter holds the condition, and the third parameter contains the increments required to implement the loop. If more than one variable is included in the initialization or the increment parameter, they should be separated by commas. The condition must evaluate to true or false.

```
for   (initialization;   condition;
increment)
 do this;
```

```
for   (initialization;   condition;
increment)
{
     do this;
     and do this;
}
```

```
for   (initialization;   condition;
increment)
{
    do this;
    for (initialization; condition;
increment)
         do   this   if   second   for
condition is true
}
```

```
<html>
      <body>
            <script language="javascript">
                  for (i = 1; i <= 10; i++){
                        document.write("Hello every one");
                        document.write(" Ha ha ..!<br>");
                  }
            </script>
      </body>
</html>
```

Control Structure (Unconditional Control)

Break

Break ends execution of the current for, while, do-while or switch structure. Break accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

```
<html>
      <body>
            <script language="javascript">
                  for (i = 1;i<=5; i++) {
                     if (i==5)
                           break;
                     document.write(i);
                  }
            </script>
      </body>
</html>
```

Continue

The continue statement does not entirely terminate the loops execution but only terminates the execution of the statements in the loops current iteration. Whenever a continue statement is encountered the rest if the statement being iterated are skipped and the control returns to the condition of the loop.

```
<html>
      <head>
            <script language="javascript">
                   for (i = 0; i < 5; ++i) {
                       if (i == 2)
                           continue;
                      document.write(i);
                   }
            </script>
      </head>
</html>
```

Using the with Keyword

The “with” keyword is used in conjunction with an object. It eliminates the need of retyping the name of the object repeatedly.

```
<html>
    <body>
        <script language = "JavaScript">
            with (document)
            {
                write("<br> <b> You dont have to type Document")
                write("<br> everytime u output a messages")
                write("<br> You can use the with keywrord")
            }
        </script>
    </body>
</html>
```

Dialog Boxes

Alert Dialog Box

The Alert dialog box is used to display some textual information on the web browser. The dialog box will have only one button ‘OK’. The JavaScript and HTML program will not be continuing

processing until the ok button is pressed. Mostly it is used when user enters some invalid data or some warning is to be displayed.

```
<html>
	<head>
		<script language="javascript">
			alert("You are in head script");
		</script>
	</head>
	<body>
		<script language="javascript">
			alert("You are in body script");
		</script>
	</body>
</html>
```

Confirm Dialog Box

A confirm box is often used if you want the user to verify or accept something. A confirm box display the predefine message and 'OK' and 'CANCEL' button. When confirm box pops up, the user will have press either ok or cancel to proceed. If the ok button is pressed it will return true and if the cancel button is pressed it will return false

```
<html>
	<head>
		<title>
			Javascript
		</title>
		<script language="javascript">
			alert("welcome")
			if(confirm("do you want to display image as
background"))
				document.write("<body
background='friends.jpg'>")

		</script>
	</head>
	<body>
		hello every one
	</body>
</html>
```

Prompt Dialog Box

A prompt box is often used if you want the user to input a value before entering a page. A prompt box display a predefine message, a textbox for user input and display ok and cancel button. When a prompt box pops up, the user will have to click on ok or cancel to proceed after entering an input. If the user clicks ok it will return the value and if cancel is pressed it will return NULL. The prompt is two parts first is the message to the user and second is default value, which will be returned on by pressing ok button without changing it.

```
<html>
	<body>
		<script language="javascript">
			var name = prompt("Enter your name","Firstname")
			document.write("Welcome " + name)
		</script>
	</body>
</html>
```

Array

JavaScript arrays are used to store multiple values in a single variable. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but what about 300? An array can hold many values under a single name, and you can access the values by referring to an index number.

Create Array

Using an array literal is the easiest way to create a JavaScript Array.

```
var array-name = [item1, item2, ...];
var cars = ["Saab", "Volvo", "BMW"];
```

Using the JavaScript Keyword new

```
var cars = new Array("Saab", "Volvo", "BMW");
```

Access the Elements of an Array

You refer to an array element by referring to the index number. This statement accesses the value of the first element in cars:

```
var name = cars[0];
```

This statement modifies the first element in cars:

```
cars[0] = "Opel";
```

```
Note: [0] is the first element in an array. [1] is the second. Array
indexes start with 0.
```

```
<p id="demo"></p>
<script>
      var cars = ["Saab", "Volvo", "BMW"];
      document.getElementById("demo").innerHTML = cars;
</script>
```

The first line (in the script) creates an array named cars. The second line "finds" the element with id="demo", and "displays" the array in the "innerHTML" of it.

Spaces and line breaks are not important. A declaration can span multiple lines:

```
<!DOCTYPE html>
<html>
      <body>
            <p id="demo"></p>
            <script>
                  var cars = [
                        "Saab",
                        "Volvo",
                        "BMW"
                  ];
                  document.getElementById("demo").innerHTML = cars[0];
            </script>
      </body>
</html>
```

Different Objects in One Array

JavaScript variables can be objects. Arrays are special kinds of objects. Because of this, you can have variables of different types in the same Array. You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

Arrays are Objects

Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays. But, JavaScript arrays are best described as arrays. Arrays use numbers to access its "elements". In this example, person[0] returns John:

```
<!DOCTYPE html>
<html>
	<body>
		<p id="demo"></p>
		<script>
			var person = ["John", "Doe", 46];
			document.getElementById("demo").innerHTML = person[0];
		</script>
	</body>
</html>
```

Objects use names to access its "members". In this example, person.firstName returns John:

```
<!DOCTYPE html>
<html>
	<body>
		<p id="demo"></p>
		<script>
			var person = {firstName:"John", lastName:"Doe",
age:46};
			document.getElementById("demo").innerHTML =
person["firstName"];
		</script>
	</body>
</html>
```

Adding Array Elements

The easiest way to add a new element to an array is using the push method:

```
<!DOCTYPE html>
<html>
	<body>
		<button onclick="myFunction()">Try it</button>
		<p id="demo"></p>
		<script>
			var fruits = ["Banana", "Orange", "Apple", "Mango"];
			document.getElementById("demo").innerHTML = fruits;
			function myFunction() {
				fruits.push("Lemon")
				document.getElementById("demo").innerHTML =
fruits;
			}
		</script>
	</body>
</html>
```

New element can also be added to an array using the length property:

```
<!DOCTYPE html>
<html>
	<body>
		<button onclick="myFunction()">Try it</button>
		<p id="demo"></p>
		<script>
			var fruits = ["Banana", "Orange", "Apple", "Mango"];
			document.getElementById("demo").innerHTML = fruits;
			function myFunction() {
				fruits[fruits.length] = "Lemon";
				document.getElementById("demo").innerHTML =
fruits;
			}
		</script>
	</body>
</html>
```

Adding elements with high indexes can create undefined "holes" in an array:

```
<!DOCTYPE html>
<html>
	<body>
		<button onclick="myFunction()">Try it</button>
		<p id="demo"></p>
		<script>
			var fruits = ["Banana", "Orange", "Apple", "Mango"];
			document.getElementById("demo").innerHTML = fruits;
			function myFunction() {
				fruits[10] = "Lemon";
				document.getElementById("demo").innerHTML =
fruits[8];
			}
		</script>
	</body>
</html>
```

Looping Array Elements

The best way to loop through an array, is using a "for" loop:

```
<!DOCTYPE html>
<html>
	<body>
		<p id="demo"></p>
		<script>
			var fruits, text, fLen, i;
			fruits = ["Banana", "Orange", "Apple", "Mango"];
			fLen = fruits.length;
			text = "<ul>";
			for (i = 0; i < fLen; i++) {
				text += "<li>" + fruits[i] + "</li>";
			}
			text += "</ul>";
			document.getElementById("demo").innerHTML = text;
		</script>
	</body>
</html>
```

Associative Arrays

Many programming languages support arrays with named indexes. Arrays with named indexes are called associative arrays (or hashes). JavaScript does not support arrays with named indexes. In JavaScript, arrays always use numbered indexes.

```
<!DOCTYPE html>
<html>
	<body>
		<p id="demo"></p>
		<script>
			var person = [];
			person[0] = "John";
			person[1] = "Doe";
			person[2] = 46;
			document.getElementById("demo").innerHTML =
			person[0] + " " + person.length;
		</script>
	</body>
</html>
```

Note: If you use a named index, JavaScript will redefine the array to a standard object. After that, all array methods and properties will produce incorrect results.

```
<!DOCTYPE html>
<html>
	<body>
		<p id="demo"></p>
		<script>
			var person = [];
			person["firstName"] = "John";
			person["lastName"] = "Doe";
			person["age"] = 46;
			document.getElementById("demo").innerHTML =
			person[0] + " " + person.length;
		</script>
	</body>
</html>
```

The Difference between Arrays and Objects is; In JavaScript, arrays use numbered indexes. In JavaScript, objects use named indexes. Arrays are a special kind of objects, with numbered indexes.

When to Use Arrays? When to use Objects?

- JavaScript does not support associative arrays.
- You should use objects when you want the element names to be strings (text).
- You should use arrays when you want the element names to be numbers.

Avoid new Array(); There is no need to use the JavaScript's built-in array constructor new Array(). Use [] instead.

These two different statements both create a new empty array named points:

```
var points = new Array();          // Bad
var points = [];                   // Good
```

What Is Function?

A function is defined as a block of statements which can be invoked whenever required within a script. It can well be referred to as a subroutine which can be called with the script at any point of execution. It is the core of structured programming.

How to Declare a Function

This includes the actual list of instruction which contribute to the function. It starts with a line known as declaratory, which is followed by the function body. The function body is enclosed within braces and is a set of statement due for execution once the function evoked. To create a function you define its name, any values ("arguments"), and some statements. A arguments mean some value is to be transferred to a function for some logical or arithmetic calculation. Value can be as variable or directly constant.

How to Call a Function

The function is to be called by its name and parenthesis bracket e.g. funct() and if there is some value to be passed to the function then it to be done with the argument e.g. funct(name,age)

Return Value From Function

Functions that will return a result must use the "return" statement. This statement specifies the value which will be returned to where the function was called from.

Form Of UDF

```
function functionname()
{
    function body;
    do this;
}
```

```
function
functionname(argument1,argument2)
{
    function body;
    do this;
}
```

```
function
functionname(argument1,argument2)
{
    function body;
    do this;
   return value;
}
```

```
function functionname()
{
    function body;
    do this;
   return value;
}
```

Types Of Function

Simple function [without argument and without return]

```
<html>
      <script language="javascript">
            var name=""
            function hello(){
                  name = prompt("Enter your name")
                  alert("Welcome " + name)
            }
            function bye(){
                  alert("Good bye " + name)
            }
            hello();
            bye();
      </script>
</html>
```

Function with arguments

```
<html>
      <script language="javascript">
            var name=""
            function hello(name){
                  alert("Welcome " + name)
            }
            function bye(){
                  alert("Good bye " + name)
```

```
            }
            hello("Javascript")
            name="Language"
            bye(name)
      </script>
</html>
```

Function with arguments with return

```
<html>
      <script language="javascript">
            var name=""
            function hello(usrname){
                  return usrname+ "Javascript"
            }
            name=hello("Language")
            document.write(name)
      </script>
</html>
```

Function without arguments with return

```
<html>
      <script language="javascript">
            var name=""
            function hello()  {
                  return "Javascript"
            }
            name=hello()
            document.write(name)
      </script>
</html>
```

Javascript Built In Function

String Function

Function	Explanation
big()	Method is used to display string in bold font
small()	Method is used to display string in small font
bold()	Method is used to display string in bold
italics()	Method is used to display string in italic
strike()	String will be displayed as strikethrough
fontcolor()	Used to change the color of the fonts
fontsize()	Changes the size of the fonts
link()	String will be displayed as link
charAt()	Returns the character from the specified location
concat()	Used to join two or more string
indexOf()	Returns the position of the first occurrence of string, if not found returns -1
lastIndexOf()	Returns the position of the last occurrence of string, if not found returns -1
match()	Returns the searched string if found
replace()	Used to replace the part of string with new string
search()	Returns the position of the occurrence of string, if not found returns -1. Provides flag 'i' for case-insensitive
slice()	Extracts a part of string and returns it. [ending index position optional] [start, end]
substr()	Extracts a part of string and returns it. [number of character is optional [start, length]
substring()	Extracts a part of string and returns it. [number of character is optional [start, stop]
toLowerCase()	Converts string to lower case
toUpperCase()	Converts string to upper case
length [property]	Counts the elements or character of object
charCodeAt()	Returns the Unicode of the specified indexed char
fixed()	Returns the string as teletype text: <tt>string</tt>
split()	Replaces the specified characters with a comma.
sub()	Returns the String formatted to subscript _{string}
sup()	Returns the String formatted to superscript ^{string}

```
<html>
	<body>
		<script language="javascript">
			var txt1 = "Hello World"
			var txt2 = " How are you"
			var txt3 = " Welcome to javascript"
			//length is property
			document.write(txt1.length + '<br>')

			//big Method is used to display string in bold font
			document.write(txt1.big() + '<br>')

			//small Method is used to display string in small font
			document.write(txt1.small() + '<br>')

			//bold Method is used to display string in bold
```

```
                    document.write(txt1.bold() + '<br>')

                    //italics Method is used to display string in italic
                    document.write(txt1.italics() + '<br>')

                    //strike String will be displayed as  strikethrough
                    document.write(txt1.strike() + '<br>')

                    //fontcolor Used to change the color of the fonts
                    document.write(txt1.fontcolor('#00ccaa') + '<br>')

                    //fontcolor Used to change the color of the fonts
                    document.write(txt1.fontcolor('RGB(255,100,100)') +
'<br>')

                    //font Changes the size of the fonts
                    document.write(txt1.fontsize(3) + '<br>')

                    //link String will be displayed as link
                    document.write(txt1.link('http://www.yahoo.com') +
'<br>')
                    //charAt Returns the character from the specified
location
                    document.write(txt1.charAt(3) + '<br>')

                    //concat Used to join two or more string
(str1,str2.......)
                    document.write(txt1.concat(txt2,txt3) + '<br>')

                    //indexOf(searchvalue[,fromindex]) it's case
sensitive.returns -1 on not found
                    document.write(txt1.indexOf("Hello") + '<br>')

                    //lastIndexOf Returns the position of the last
occurrence of string, if not found returns -1
                    document.write(txt1.lastIndexOf("o") + '<br>')

                    //match will return the searched string if it is
found,else it will return 'null'.the method is case sensitive.
                    document.write(txt1.match("world") + '<br>')

                    //replace will replace searched string with second
parameter.it is case sensitive.
                    document.write(txt2.replace("are","r") + '<br>')

                    // 'i' is a flag to make replace case-insensitive.
                    document.write(txt2.replace(/ARE/i,"r") + '<br>')

                    //search is same as indexOf, additional to it 'i' flag
can also be used
                    document.write(txt3.search(/WELCOME/i) + '<br>')

                    //slice Extracts a part of string and returns it.
[ending index position optional]
                    document.write(txt3.slice(2,4) + '<br>')

                    //slice Extracts a part of string and returns it.
[minus value for reverse ending index position optional]
                    document.write(txt3.slice(10,-4) + '<br>')

                    //substr return the number of character from index
position.index starts with 0
                    document.write(txt2.substr(2,8) + '<br>')
```

```
                    //substring return the character from index postion to
index position.index starts with 0
                    document.write(txt2.substring(3,4) + '<br>')

                    //toLowerCase convert the string in lower case
                    document.write(txt2.toLowerCase() + '<br>')

                    //toUppercase convert the string in upper case
                    document.write(txt3.toUpperCase() + '<br>')

document.write(txt2.toUpperCase().fontcolor("#aabbcc").fontsize(7))
              </script>
      </body>
</html>
```

Date Functions

Methods	Explanation
Date()	Returns a new Date object
getDate()	Returns the day of the month (1-31) of object
getDay()	Returns the weekday (0-6) of object
getMonth()	Returns the month. (0-11) of object
getYear()	Returns the year in 2 or 4 [1900-1999 2 digit year] of object
getFullYear()	Returns the year. (2000) of object
getHours()	Returns the hour as a value between 0 & 23 of object
getMinutes()	Returns the minute. (0-59) of object
getSeconds()	Returns the second. (0-59) of object
getMilliseconds()	Returns milliseconds of object [2 digit less then 100,1 digit less 10]
getTime()	Returns the milliseconds since midnight January 1,1970
setDate()	Set date to object (1-31)
setMonth()	Set Month to object (0-11)
setFullYear()	Set full year (year, month, day) month and day optional
setYear()	Set year in object 1900-1999 2 digit year, else four digit
setHours()	Set Hours in object (hrs, min, sec, millisec) hours only compulsory
setMinutes()	Set minutes to object (min, sec ,milliseconds) min compulsory
setSeconds()	Set seconds to object (sec, milliseconds) sec compulsory
toGMTString()	Returns a string date value.
toLocaleString()	Returns a string date value.
toString()	Returns a string date value.

```
<html>
      <body>
            <script language="javascript">
                  var dt = new Date()
                  with (document){
                        //Date returns the date
                        write(Date() + '<br>')

                        //getDate return a day of month 1 to 31
                        write(dt.getDate() + '<br>')
                        //getDay returns a day of week 0 to 6.0=sunday
                        write(dt.getDay() + '<br>')

                        //getMonth returns a month of year 1 to 11
                        write(dt.getMonth() + '<br>')
```

```
                        //getYear return 2 digit year for 1900 to 1999,4
digit for remaining
                        write(dt.getYear() + '<br>')
                        //getFullYear return 4 digit for any year
                        write(dt.getFullYear() + '<br>')

                        //getHours returns 1 digit for 1 to 9 and 2
digit for 10 to 24
                        write(dt.getHours() + '<br>')

                        //getMinutes returns 1 digit for 1 to 9 and 2
digit for 10 to 60
                        write(dt.getMinutes() + '<br>')
                        //getSeconds returns 1 digit for 1 to 9 and 2
digit for 10 to 60
                        write(dt.getSeconds() + '<br>')
                        //getMilliseconds Returns milliseconds of object
[2 digit less then 100,1 digit less 10]
                        write(dt.getMilliseconds() + '<br>')
                        //getTime Returns the milliseconds since
midnight January 1,1970
                        write(dt.getTime() + '<br>')
                        //setDate Set's date to object (1-31)
                        dt.setDate(10)
                        write(dt.getDate() + '<br>')
                        //setMonth Set's Month to object (0-11)
                        dt.setMonth(10)
                        write(dt.getMonth() + '<br>')
                        //setYear Set's full year (year, month, day)
month and day optional
                        dt.setYear(1700)
                        write(dt.getYear() + '<br>')
                        //setFullYear Set's year in object 1900-1999 2
digit year, else four digit
                        dt.setFullYear(90)
                        write(dt.getFullYear() + '<br>')
                        //setHours's Set Hours in object (hrs, min, sec,
millisec) hours only compulsory
                        dt.setHours(10)
                        write(dt.getHours() + '<br>')
                        //setMinutes Set's minutes to object (min, sec
,milliseconds) min compulsory
                        dt.setMinutes(33)
                        write(dt.getMinutes() + '<br>')
                        //setSeconds Set's seconds to object (sec,
milliseconds) sec compulsory
                        dt.setSeconds(44)
                        write(dt.getSeconds() + '<br>')
                    }
                </script>
        </body>
</html>
```

Math functions

Methods	Explanation
max()	Returns the number with the higher value of two numbers
min()	Returns the number with the lower value of two numbers
random()	Returns a random number between 0 and 1
round()	Returns a number rounded to the nearest whole numbers
abs()	Returns absolute value of a number

ceil()	Returns number rounded larger nearest integer
floor()	Returns number rounded smaller nearest integer
exp()	Returns EX [Euler's] value
cos()	Returns cosine value
sin()	Returns sine number
tan()	Returns tangent number
pow()	Returns value of x to power of y
ceil()	Returns a the nearest whole number greater than or equal to the number
cos()	Returns the cosine of a number
log()	Returns the logarithm of a number
PI	Returns PI
sqrt()	Returns the square root of a number
SQRT1_2	Returns the square root of 0.5
SQRT2	Returns the square root of 2

```
<html>
	<body>
		<script language="javascript">
			with (document){
				//abs display's absolute values only
					write(Math.abs(7.325) + '<br>')
				//ceil display's next greater integer value
					write(Math.ceil(4.25)+ '<br>')
				//floor gives previous integer value
					write(Math.floor(10.22)+ '<br>')
//exp returns Euler constant base of natural logarithms
					write(Math.exp(1)+ '<br>')
				//cos returns consine of number
					write(Math.cos(2)+ '<br>')
				//sin returns sine of number
					write(Math.sin(2)+ '<br>')
				//tan returns tangent of angle
					write(Math.tan(4)+ '<br>')
				//log returns log,if argument is negative NaN is
returned
					write(Math.log(2)+ '<br>')
			//pow returns the value of x to the power of y
					write(Math.pow(2,2)+ '<br>')
			//random returns random value between 0 to 1
					write(Math.random()+ '<br>')
			//max gives maximum value
					write(Math.max(24,5,6)+ '<br>')
			//min returns minimum value
					write(Math.min(2,2)+ '<br>')
			//round gives nearest integer value
					write(Math.round(5.22222)+ '<br>')
				write(Math.E  + '<br>')
				write(Math.PI + '<br>')
			}
		</script>
	</body>
</html>
```

Array Functions

Methods	Explanation
concat()	Merge two or more array, not affect to original array
join()	Change separator for elements [default comma]

pop()	Remove the last element of array, returns the last element , affect to original array
push()	Adds a element at the end of array, returns the length of array , affect to original array
reverse()	Make the array in reverse order , affect to original array, returns new array
shift()	Remove the first element of array , return last element , affect to original array
sort()	Sort the elements alphabetically , ASCII on character
length [property]	Returns the length of array

```
<html>
	<script language="javascript">
		myarr1 = new Array(5,4,9,3,2.5)
		myarr2 = new Array("aa","er","er",1,5,false)
		//concat
			document.write(myarr1.concat(myarr2));
		//join
			document.write(myarr1.join("<br>"))
		//pop
			document.write(myarr1.pop())
		//length
			document.write(myarr1.length)
		//push
			document.write(myarr1.push(1,2,3))
		//reverse
			document.write(myarr2.reverse())
		//shift
			document.write(myarr2.shift())
		//sort
			document.write(myarr1.sort())
	</script>
</html>
```

Events

- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
- When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.
- Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.
- Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

Mouse Events

Event	Description	DOM
onclick	The event occurs when the user clicks on an element	2
ondblclick	The event occurs when the user double-clicks on an element	2
onmousedown	The event occurs when the user presses a mouse button over an element	2
onmouseleave	The event occurs when the pointer is moved out of an element	2
onmousemove	The event occurs when the pointer is moving while it is over an element	2
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children	2
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	2
onmouseup	The event occurs when a user releases a mouse button over an element The order of events related to the onmouseup event (for the left/middle mouse button): 1. onmousedown 2. onmouseup 3. onclick The order of events related to the onmouseup event (for the right mouse button): 1. onmousedown 2. onmouseup 3. oncontextmenu	2

Keyboard Events

Event	Description	DOM
onkeydown	The event occurs when the user is pressing a key	2
onkeypress	The event occurs when the user presses a key	2
onkeyup	The event occurs when the user releases a key. The order of events 1. onkeydown 2. onkeypress 3. onkeyup	2

Frame/Object Events

Event	Description	DOM
onabort	The event occurs when the loading of a resource has been aborted	2
onbeforeunload	The event occurs before the document is about to be unloaded	2

onerror	The event occurs when an error occurs while loading an external file	2
onhashchange	The event occurs when there has been changes to the anchor part of a URL	3
onload	The event occurs when an object has loaded	2
onpageshow	The event occurs when the user navigates to a webpage	3
onpagehide	The event occurs when the user navigates away from a webpage	3
onresize	The event occurs when the document view is resized	2
onscroll	The event occurs when an element's scrollbar is being scrolled	2

Form Events

Event	Description	DOM
onblur	The event occurs when an element loses focus	2
onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <keygen>, <select>, and <textarea>)	2
onfocus	The event occurs when an element gets focus	2
onfocusin	The event occurs when an element is about to get focus	2
onfocusout	The event occurs when an element is about to lose focus	2
oninput	The event occurs when an element gets user input	3
oninvalid	The event occurs when an element is invalid	3
onreset	The event occurs when a form is reset	2
onsearch	The event occurs when the user writes something in a search field (for <input="search">)	3
onselect	The event occurs after the user selects some text (for <input> and <textarea>)	2
onsubmit	The event occurs when a form is submitted	2

MouseEvent Object

Property	Description	DOM
altKey	Returns whether the "ALT" key was pressed when the mouse event was triggered	2
button	Returns which mouse button was pressed when the mouse event was triggered	2
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered	2
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered	2
ctrlKey	Returns whether the "CTRL" key was pressed when the mouse event was triggered	2
detail	Returns a number that indicates how many times the mouse was clicked	2
pageX	Returns the horizontal coordinate of the mouse pointer, relative to the document, when the mouse event was triggered	
pageY	Returns the vertical coordinate of the mouse pointer, relative to the document, when the mouse event was triggered	
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered	2
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered	2
shiftKey	Returns whether the "SHIFT" key was pressed when an event was triggered	2

which	Returns which mouse button was pressed when the mouse event was triggered	2

KeyboardEvent Object

Property	Description	DOM
altKey	Returns whether the "ALT" key was pressed when the key event was triggered	2
ctrlKey	Returns whether the "CTRL" key was pressed when the key event was triggered	2
charCode	Returns the Unicode character code of the key that triggered the onkeypress event	2
key	Returns the key value of the key represented by the event	3
keyCode	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event	2
location	Returns the location of a key on the keyboard or device	3
shiftKey	Returns whether the "SHIFT" key was pressed when the key event was triggered	2
which	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event	2

Clipboard Events

Event	Description
oncopy	The event occurs when the user copies the content of an element
oncut	The event occurs when the user cuts the content of an element
onpaste	The event occurs when the user pastes some content in an element

```
<!DOCTYPE html>
<html>
	<body>
		<p>Click the button to trigger a function that will output
"Hello World" in a p element with id="demo".</p>
		<button onclick="myFunction()">Click me</button>
		<p id="demo"></p>
		<script>
			function myFunction() {
				document.getElementById("demo").innerHTML =
"Hello World";
			}
		</script>
	</body>
</html>
```

```
<!DOCTYPE html>
<html>
	<body>
		<img onmouseover="bigImg(this)" onmouseout="normalImg(this)"
border="0" src="smiley.gif" alt="Smiley" width="32" height="32">
		<p>The function bigImg() is triggered when the user moves
the mouse pointer over the image.</p>
		<p>The function normalImg() is triggered when the mouse
pointer is moved out of the image.</p>
		<script>
			function bigImg(x) {
				x.style.height = "64px";
```

```
                        x.style.width = "64px";
                  }

                  function normalImg(x) {
                        x.style.height = "32px";
                        x.style.width = "32px";
                  }
            </script>
      </body>
</html>
```

```
<!DOCTYPE html>
<html>
      <body>

            <p>A function is triggered when the user releases a key in
the input field. The function transforms the character to upper
case.</p>
            Enter your name: <input type="text" id="fname"
onkeyup="myFunction()">
            <script>
            function myFunction() {
                  var x = document.getElementById("fname");
                  x.value = x.value.toUpperCase();
            }
            </script>
      </body>
</html>
```

```
<!DOCTYPE html>
<html>
      <body onload="myFunction()">
            <h1>Hello World!</h1>
            <script>
            function myFunction() {
                  alert("Page is loaded");
            }
            </script>
      </body>
</html>
```

```
<!DOCTYPE html>
<html>
      <body onresize="myFunction()">
            <p>Try to resize the browser window to display the windows
height and width.</p>
            <p id="demo"></p>
            <p><strong>Note:</strong> this example will not work
properly in IE8 and earlier. IE8 and earlier do not support the
outerWidth/outerHeight propery of the window object.</p>
            <script>
            function myFunction() {
                  var w = window.outerWidth;
                  var h = window.outerHeight;
                  var txt = "Window size: width=" + w + ", height=" + h;
                  document.getElementById("demo").innerHTML = txt;
            }
            </script>
      </body>
</html>
```

```
<!DOCTYPE html>
<html>
	<body>
		<p>Select a new car from the list.</p>
		<select id="mySelect" onchange="myFunction()">
		  <option value="Audi">Audi
		  <option value="BMW">BMW
		  <option value="Mercedes">Mercedes
		  <option value="Volvo">Volvo
		</select>
		<p>When you select a new car, a function is triggered which
outputs the value of the selected car.</p>
		<p id="demo"></p>
		<script>
		function myFunction() {
			var x = document.getElementById("mySelect").value;
			document.getElementById("demo").innerHTML = "You
selected: " + x;
		}
		</script>
	</body>
</html>
```

```
<!DOCTYPE html>
<html>
	<body>
		Select some of the text: <input type="text" value="Hello
world!" onselect="myFunction()">
		<script>
		function myFunction() {
			alert("You selected some text!");
		}
		</script>
	</body>
</html>
```

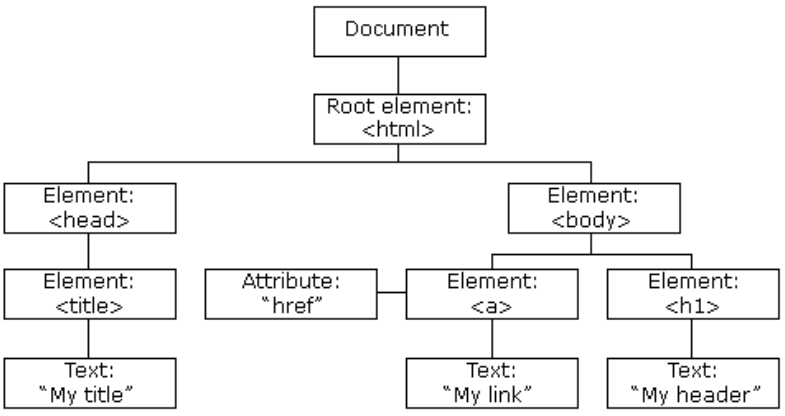
```
<!DOCTYPE html>
<html>
	<body>
		<input type="text" oncopy="myFunction()" value="Try to copy
this text">
		<p id="demo"></p>
		<script>
		function myFunction() {
			document.getElementById("demo").innerHTML = "You
copied text!"
		}
		</script>
	</body>
</html>
```

```
<!DOCTYPE html>
<html>
	<body>
		<h2 onclick="showCoords(event)">Click this heading to get
the x (horizontal) and y (vertical) coordinates of the mouse pointer
when it was clicked.</h2>

		<p><strong>Tip:</strong> Try to click different places in
the heading.</p>
```

```
            <p id="demo"></p>

            <script>
            function showCoords(event) {
                  var x = event.clientX;
                  var y = event.clientY;
                  var coords = "X coords: " + x + ", Y coords: " + y;
                  document.getElementById("demo").innerHTML = coords;
            }
            </script>
      </body>
</html>
```

```
<!DOCTYPE html>
<html>
      <body>
            <p>Press a key on the keyboard in the input field to find
out if the SHIFT key was pressed or not.</p>
            <input type="text" onkeydown="isKeyPressed(event)">
            <p id="demo"></p>
            <script>
            function isKeyPressed(event) {
                  var x = document.getElementById("demo");
                  if (event.shiftKey) {
                        x.innerHTML = "The SHIFT key was pressed!";
                  } else {
                        x.innerHTML = "The SHIFT key was NOT pressed!";
                  }
            }
            </script>
      </body>
</html>
```

DOM - Document Object Model



The DOM is a W3C (World Wide Web Consortium) standard. The DOM defines a standard for accessing documents: "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of Objects:

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

- HTML DOM methods are actions you can perform (on HTML Elements).
- HTML DOM properties are values (of HTML Elements) that you can set or change.
- A property is a value that you can get or set (like changing the content of an HTML element).
- A method is an action you can do (like add or deleting an HTML element).

```
<html>
	<body>
		<p id="demo"></p>
		<script>
			document.getElementById("demo").innerHTML = "Hello
World!";
		</script>
	</body>
</html>
:  getElementById is a method, while innerHTML is a property.
```

Finding HTML Elements

Method	Description
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name

Changing HTML Elements

Method	Description
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute(attribute, value)	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element

Adding Events Handlers

Method		Description
document.getElementById(id).onclick = function(){code}	=	Adding event handler code to an onclick event

Finding HTML Objects

The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5. Later, in HTML DOM Level 3, more objects, collections, and properties were added.

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.body	Returns the <body> element	1
document.domain	Returns the domain name of the document server	1
document.forms	Returns all <form> elements	1
document.images	Returns all <img> elements	1
document.lastModified	Returns the date and time the document was updated	3
document.links	Returns all <area> and <a> elements that have a href attribute	1
document.readyState	Returns the (loading) status of the document	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

```
<!DOCTYPE html>
<html>
	<style>
		#container {
		  width: 400px;
		  height: 400px;
		  position: relative;
		  background: yellow;
		}
		#animate {
		  width: 50px;
		  height: 50px;
		  position: absolute;
		  background-color: red;
		}
	</style>
	<body>
		<p>   <button onclick="myMove()">Click Me</button></p>
		<div id ="container">   <div id ="animate"></div>     </div>
		<script>
			function myMove() {
			  var elem = document.getElementById("animate");
			  var pos = 0;
```

```
                    var id = setInterval(frame, 5);
                    function frame() {
                        if (pos == 350) {
                          clearInterval(id);
                        } else {
                          pos++;
                          elem.style.top = pos + 'px';
                          elem.style.left = pos + 'px';
                        }
                    }
                </script>
        </body>
</html>
```

```
<!DOCTYPE html>
<html>
        <body>
                <p>This example uses the addEventListener() method to attach
a click event to a button.</p>
                <button id="myBtn">Try it</button>
                <p id="demo"></p>
                <script>

        document.getElementById("myBtn").addEventListener("click",
displayDate);

                    function displayDate() {
                        document.getElementById("demo").innerHTML =
Date();
                    }
                </script>
        </body>
</html>
```

History Object

The window.history object can be written without the window prefix. To protect the privacy of the users, there are limitations to how JavaScript can access this object.

Window History Back

The history.back() method loads the previous URL in the history list. This is the same as clicking the Back button in the browser.

```
<html>
        <head>
                <script>
                    function goBack() {
                        window.history.back()
                    }
                </script>
        </head>
        <body>
                <input type="button" value="Back" onclick="goBack()">
        </body>
</html>
```

Window History Forward

The history forward() method loads the next URL in the history list. This is the same as clicking the Forward button in the browser.

```
<html>
	<head>
	<script>
		function goForward() {
			window.history.forward()
		}
	</script>
	</head>
	<body>
		<input type="button" value="Forward" onclick="goForward()">
	</body>
</html>
```

JavaScript Form Validation

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

- **Basic Validation** – First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
- **Data Format Validation** – Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

String Validation

If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

```
<!DOCTYPE html>
<html>
	<head>
		<script>
			function validateForm() {
				var x = document.forms["myForm"]["fname"].value;
				if (x == null || x == "") {
					alert("Name must be filled out");
					return false;
				}
			}
		</script>
	</head>
	<body>
		<form name="myForm" onsubmit="return validateForm()"
method="post">
			Name: <input type="text" name="fname">
			<input type="submit" value="Submit">
		</form>
	</body>
</html>
```

Numeric Validation

```
<!DOCTYPE html>
<html>
	<body>
	<h1>JavaScript Can Validate Input</h1>
	<p>Please input a number between 1 and 10:</p>
	<input id="numb">
	<button type="button" onclick="myFunction()">Submit</button>
	<p id="demo"></p>
	<script>
		function myFunction() {
			var x, text;

			// Get the value of the input field with id="numb"
			x = document.getElementById("numb").value;

			// If x is Not a Number or less than one or greater
than 10
			if (isNaN(x) || x < 1 || x > 10) {
				text = "Input not valid";
			} else {
				text = "Input OK";
			}
			document.getElementById("demo").innerHTML = text;
		}
	</script>
	</body>
</html>
```

JavaScript Regular Expressions

A regular expression is a sequence of characters that forms a search pattern. The search pattern can be used for text search and text replace operations.

What Is a Regular Expression?

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for. A regular expression can be a single character, or a more complicated pattern. Regular expressions can be used to perform all types of text search and text replace operations.

Using String search()

```
<!DOCTYPE html>
<html>
	<body>
	<p>Search a string for "Kundalia", and display the position of the
match:</p>
	<button onclick="myFunction()">Try it</button>
	<p id="demo"></p>
	<script>
		function myFunction() {
			var str = "Visit Kundalia!";
			var n = str.search(/Kundalia/i);
			document.getElementById("demo").innerHTML = n;
		}
	</script>
	</body>
</html>
```

Use String replace()

```
<!DOCTYPE html>
<html>
	<body>
		<p>Replace "microsoft" with "Linux" in the paragraph
below:</p>
		<button onclick="myFunction()">Try it</button>
		<p id="demo">Please visit Microsoft!</p>
		<script>
			function myFunction() {
				var str =
document.getElementById("demo").innerHTML;
				var txt = str.replace(/microsoft/i,"Linux");
				document.getElementById("demo").innerHTML = txt;
			}
		</script>
	</body>
</html>
```

Regular Expression Modifiers

Modifiers can be used to perform case-insensitive more global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Metacharacters are characters with a special meaning:

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*

```
<!DOCTYPE html>
<html>
	<body>
		<p>Search for an "e" in the next paragraph:</p>
		<p id="p01">The best things in life are free!</p>
		<button onclick="myFunction()">Try it</button>
		<p id="demo"></p>
		<script>
```

```
                  function myFunction() {
                        text = document.getElementById("p01").innerHTML;
                        document.getElementById("demo").innerHTML =
/e/.test(text);
                  }
            </script>
      </body>
</html>
```

```
<!DOCTYPE html>
<html>
      <body>
      <p>Search for an "e" in the next paragraph:</p>
      <p id="p01">The best things in life are free!</p>
      <button onclick="myFunction()">Try it</button>
      <p id="demo"></p>
      <script>
            function myFunction() {
                  text = document.getElementById("p01").innerHTML;
                  document.getElementById("demo").innerHTML =
/e/.exec(text);
            }
      </script>
      </body>
</html>
```

```
<html>
      <head>
            <script type="text/javascript">
                  var ck_name = /^[A-Za-z0-9 ]{3,20}$/;
                  var ck_email = /^([\w-]+(?:\.[\w-]+)*)@((?:[\w-
]+\.)*\w[\w-]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$/i
                  var ck_username = /^[A-Za-z0-9_]{1,20}$/;
                  var ck_password =  /^[A-Za-z0-9!@#$%^&*()_]{6,20}$/;
                  function validate(form){
                        var name = form.name.value;
                        var email = form.email.value;
                        var username = form.username.value;
                        var password = form.password.value;
                        var gender = form.gender.value;
                        var errors = [];

                        if (!ck_name.test(name)) {
                        errors[errors.length] = "You valid Name .";
                        }

                        if (!ck_email.test(email)) {
                              errors[errors.length] = "You must enter a
valid email address.";
                        }
                        if (!ck_username.test(username)) {
                              errors[errors.length] = "You valid
UserName no special char .";
                        }

                        if (!ck_password.test(password)) {
                              errors[errors.length] = "You must enter a
valid Password min 6 char.";
                        }
                        if (gender==0) {
                              errors[errors.length] = "Select Gender";
                        }
```

```
                        if (errors.length > 0) {
                              reportErrors(errors);
                              return false;
                        }

                        return true;
                  }

                  function reportErrors(errors){
                        var msg = "Please Enter Valide Data...\n";
                        for (var i = 0; i<errors.length; i++) {
                              var numError = i + 1;
                              msg += "\n" + numError + ". " + errors[i];
                        }
                        alert(msg);
                  }
            </script>
      </head>
      <body id="public">
            <form onSubmit="return validate(this);" name="form">
                  <h2>Registration Form</h2>
                  Full Name <input id="name" name="name" type="text"
maxlength="50" /> <br /> <br />
                  Email <input id="email" name="email" type="text"
maxlength="30" /> <br /> <br />
                  User-ID     <input id="username" name="username"
type="text" maxlength="10" /> <br /> <br />
                  Password <input id="password" name="password"
type="password" maxlength="14" /> <br /> <br />
                  Gender
                        <select id="gender" name="gender">
                              <option value="0">Gender</option><option
value="1">Male</option><option value="2">Female</option>
                        </select> <br /> <br />
                  <input  type="submit" value="Submit" style="
background:#0060a1; color:#FFFFFF; font-size:14px; border:1px solid
#0060a1"/>
            </form>
      </body>
</html>
```