U. P. RAJARSHI TANDON
OPEN UNIVERSITY

BCA-E9

Bachelor in Computer Applications

Block

1

History of Internet and www

Printed and Published by Pro. (Dr.) Arun Kumar Gupta, Registrar, **Uttar Pradesh Rajarshi Tandon Open University, 2019.**

Printed By : Chandrakal Universal Pvt Ltd. 42/7 Jawahar Lal Neharu Road Allahabad

BLOCK INTRODUCTION

The objective of this course is to explain the history and hence evolution of Internet and basic concepts of Web technology. Basically; Web technology deals with several other technologies like Java, JSP, PHP, ASP etc. It is also important to know the difference between the client side programming and server side programming. Hypertext Markup Language (HTML) is the basic and important language which is used for web development. HTML came into picture or rather it was incorporated with the Netscape Navigator, the first Internet browser. Later; it was embedded in other browsers too like Internet Explorer, Mozilla, Opera etc. Usually they are the main platform for the web page and used for client side programs. But with the inception of DHTML, It became more robust and made server side programming possible. DHTML stands for dynamic HTML. This basically consists three other technologies altogether i.e. JavaScript, XML, CSS. The combination of HTML and XML is known as XHTML.

In this course you will also learn the difference between static and dynamic web programming. You will know that how small programs using HTML can be done. After this you will also learn the server side technologies ie ASP, Servlet, and JSP. ASP and JSP are two popular technologies which are widely used in Web development. By learning the HTML, ASP and JSP you can develop Internet programs. This course aims to provide an extensive variety of topics on Web based programs with appropriate applications and examples. The course is organized into following ten units in all. These are briefly described below:

Unit 1 introduces the notion of Internet and fundamental principles of HTML.

Unit 2 covers the basic features of event models and image functions which are basically responsible for better interface in web designing.

Unit 3 describes the role and functionality of dynamic model and collections.

Unit 4 revolves around the concept of event driven programming or event model.

Unit 5 explains about Transitions in DHTML which basically changes from one static/dynamic Web page to another Web page in terms of its special effects for example conversion of colour image to gray scale image.

Unit 6 is used for the development of a bridge between front-end and back-end. It provides a detail description of various controls like ListContrls, HtmlControls, Web Controls.

Unit 7 aims to describe the basics of Database and DBMS. It explains about the database and data models in brief. It also aims to describe the SQL basic commands.

Unit 8 deals with the topics about Web pages and documents on the Internet that provides useful information are the web resources.

Unit 9 explains about the Servlet which is an advance Java technology and widely used for many applications like session handling, applying security at different levels etc.

Unit 10 aims to describe the JSP (Java Server pages), a server side advance Java technology acts as a template for HTML, Java, & JSP itself. In many design patterns it is widely used.

UNIT 1 : INTERNET AND WWW

Structure

1.0 INTRODUCTION

Network is very popular and important term with a great meaning in it. The Internet evolved from the ARPANET, to which other networks were added to form an internetwork. The present Internet is actually a collection of many thousands of networks, rather than a single network. What characterizes it is the use of the TCP/IP protocol stack throughout. Basically when two or more than two computers are connected in such a way by means of wire or wireless medium that they are able to exchange the information then they form a computer network. Internet is the abbreviation for the International network or a network of networks, bringing together people, information, hardware and software around the World. You can connect to the Internet by dialing out to an Internet Service Provider (ISP) using SLIP (Serial Line Internet Protocol) or PPP (Point to Point Protocol) or directly through Cable Modem, DSL (Digital Subscriber Line), dedicated ISP connection, using TCP/IP (Transmission Control Protocol/ Internet Protocol).

Roughly speaking, networks can be divided up into LANs, MANs, WANs, and internetworks, with their own characteristics, technologies, speeds, and niches. LANs cover a building and operate at high speeds. MANs cover a city, for example, the cable television system, which is now used by many people to access the Internet. WANs cover a country or continent. LANs and MANs are unswitched (i.e., do not have routers); WANs are switched. Wireless networks are becoming extremely popular, especially wireless LANs. Networks can be interconnected to form internetworks.

Today you cannot live without the Internet. Many areas in which it is useful and necessary are: Communications, Information search, File manipulation, Remote control of other computers, Net through hypermedia, E-Commerce (Electronic-Commerce). M-Commerce (Mobile-Commerce) is another important area which has come into the picture due to the huge increment of mobile users today. The day has come when people are unable to live without Internet. You have seen that the use of Internet is increasing gradually now-a-days due to the inception of smart phones. They are interested only in what they can get benefit from. Even though, Internet is now modern technology which provides a lot of advantages for society. Moreover, it also has disadvantage if we use it without meaningful information and hence it will lead problems to the whole society.

1.1 OBJECTIVES

After the end of this unit, you will be able to:

- o History of Internet
- o How to get into Internet
- o Basic protocols and technologies used in Internet
- o Role of HTML in creating Internet programs
- o Advantage of Internet
- o Future of Internet

1.2 HISTORY OF THE INTERNET

The Internet is at once a world-wide broadcasting capability, a mechanism for information distribution, and a medium for collaboration and interaction between individuals and their computers without regard for geographic location. The Internet is a product of the Cold War. It was originally developed by the Government of the United States during the 1970s as a means of sharing information and protecting communications in the event of a nuclear attack. During the 1980s it developed quickly, first into an academic exchange network, then as a means of mass electronic communication available in principle to anyone having access to a personal computer and a telephone line.

Moreover, It is a global network of computers (also known as the World-Wide Web) which allows instantaneous access to an expanding number of individual Web sites offering information about practically anything and everything—including the contents of daily newspapers, the price of goods in local shopping malls, library holdings, commodity prices, sports news and gossip, eroticism, and so-called chat-rooms (by means of which people can communicate with each other on-line about their interests, hobbies, and opinions).

The Internet had its roots during the 1960's as a project of the United States government's Department of Defense, to create a non-centralized network. This project was called ARPANET (Advanced Research Projects Agency Network), created by the Pentagon's Advanced Research Projects Agency established in 1969 to provide a secure and survivable communications network for organizations engaged in defense-related research. In order to make the network more global a new sophisticated and standard protocol was needed. They developed IP (Internet Protocol) technology which defined how electronic messages were packaged, addressed, and sent over the network. The standard protocol was invented in 1977 and was called TCP/IP (Transmission Control Protocol/Internet Protocol). TCP/IP allowed users to link various branches of other complex networks directly to the ARPANET, which soon came to be called the Internet. Furthermore, the different domains/application areas of Internet are shown in figure-1.1 below.

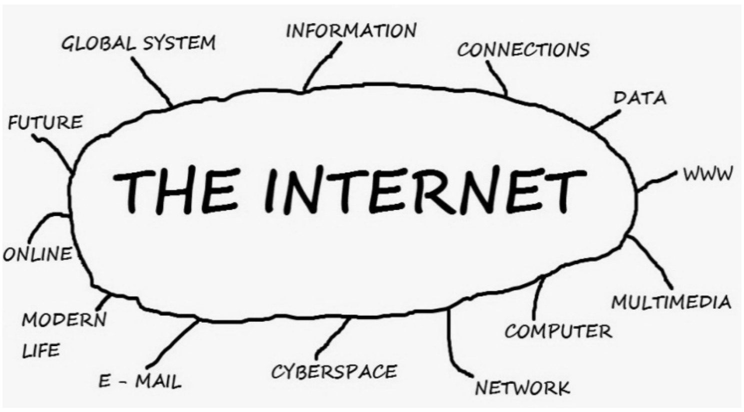


Figure 1.1: Domain of Internet

Internet was based on the idea that there would be multiple independent networks of rather arbitrary design, beginning with the ARPANET as the pioneering packet switching network, but soon to include packet satellite networks, ground-based packet radio networks and other networks. The Internet as we now know, embodies a key underlying technical idea, namely that of open architecture networking. In this approach, the choice of any individual network technology was not dictated by particular network architecture but rather could be selected freely by a provider and made to interwork with the other networks through a meta-level "Internetworking Architecture". Recall that Kleinrock had shown in 1961 that packet switching was a more efficient switching method. Along with packet switching, special purpose interconnection arrangements between networks were another possibility. While there were other limited ways to interconnect different networks, they required that one be used as a component of the other, rather than acting as a peer of the other in offering end-to-end service.

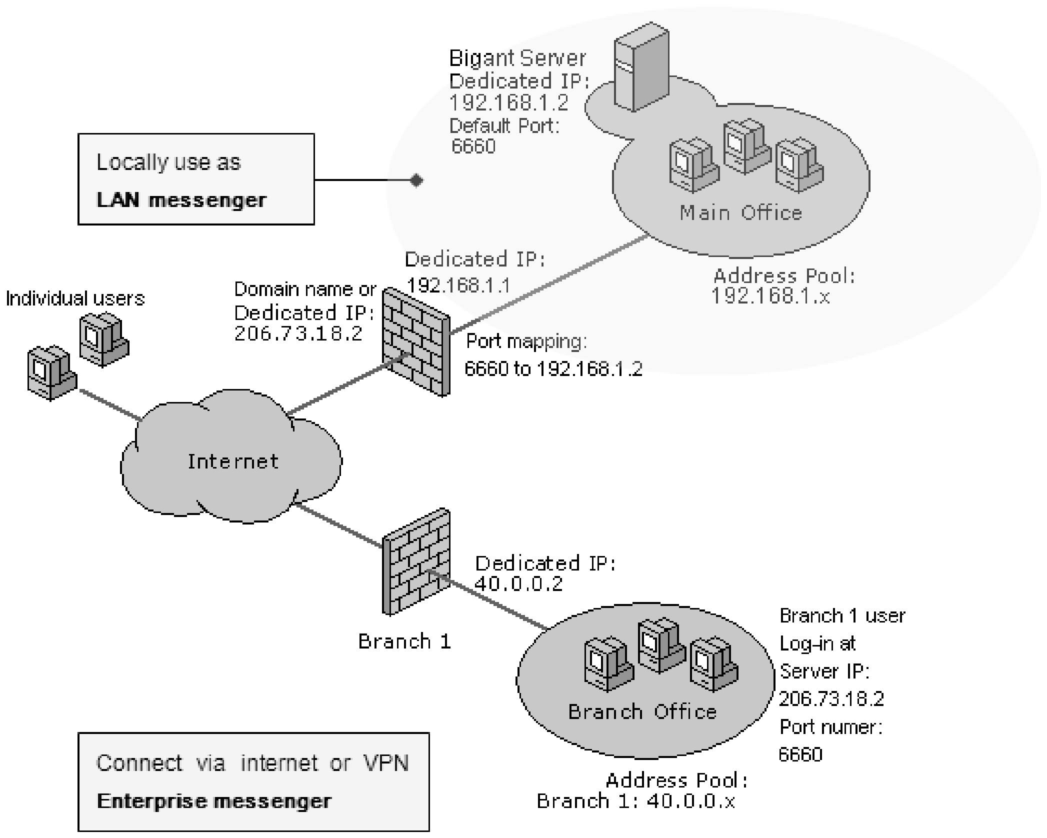


Figure 1.2: Internet and Its Connections

The World Wide Web (WWW) allows computer users to position and view multimedia-based documents (i.e., documents with text, graphics, animations, audios and/or videos) on almost any subject. Even though the Internet was developed more than three decades ago, the introduction of the WWW was a relatively recent event. In 1990, Tim Berners-Lee of CERN (the European Laboratory for Particle Physics) developed the World Wide Web and several communication protocols that form the backbone of the WWW. The Internet and the World Wide Web will surely be listed among the most significant and profound creations of humankind. In the past, most computer applications ran on standalone computers. (i.e., computers that were not connected to one another) Today's applications can be written to communicate among the world's hundreds of millions of computers. The Internet makes our work easier by mixing computing and communications technologies. It makes information immediately and conveniently accessible worldwide. It makes it possible for individuals and small businesses to get worldwide contact. In the last decade, the Internet and World Wide Web have altered the way people communicate, conduct business and manage their daily lives. They are changing the nature of the way business is done.

1.3 WORLD WIDE WEB (WWW)

WWW, basically, is everything today. Without WWW, we cannot get even a single service of Internet or other things which are available on Web. The Web, or World Wide Web (WWW), is conceptually a system of Internet servers that support specially formatted documents like HTML, XML, JavaScript, Java etc. In other words, it is a system of interlinked internet documents. The documents are formatted in a markup language called HTML (Hyper Text Markup Language) or XML (extensible markup language) that supports links to other available web documents, as well as graphics, audio, and video files. This means you can jump from one document to another simply by clicking on hot spots. Not all Internet servers are part of the WWW. Earlier, with the inception of SGML/HTML in Netscape Navigator, web pages were static but today they are dynamic with very advance features. Moreover, dynamic web pages need robust technologies like ASP, JSP, Java, PHP etc. Furthermore, WWW has some important components (Figure-1.3) like client, server, HTML, HTTP, URI (Uniform Resource Identifier).

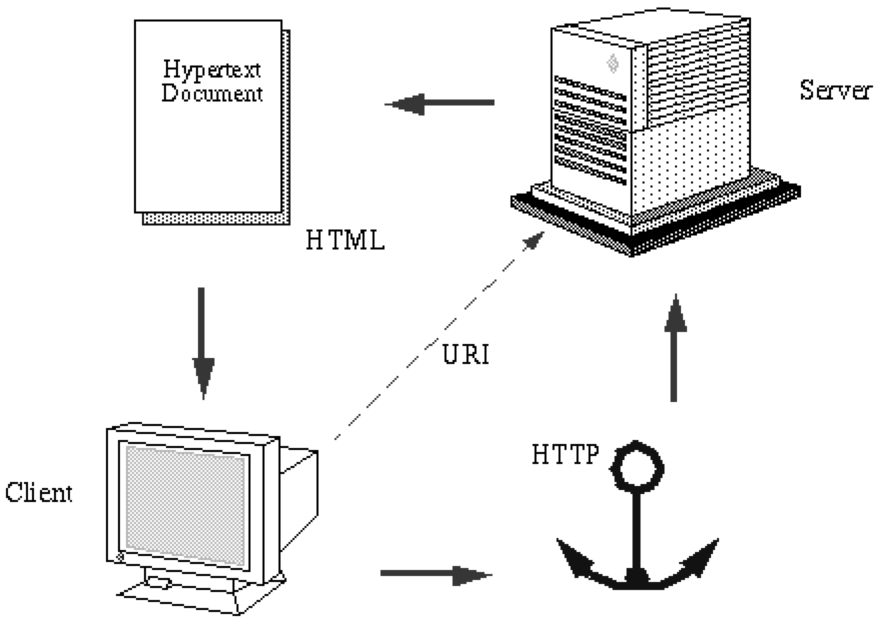


Figure 1.3: Components of WWW

Also, WWW is the largest network in the world that connects hundreds of thousands of individual networks all over the world and therefore the most popular term used for the Internet is the "Information Highway". Rather than moving through geographical space, it moves your ideas and information through cyberspace – the space of electronic movement of ideas and information. The Web consists of information organized into web pages containing text and multimedia files. It contains hypertext links or highlighted keywords and images that lead to related information. A collection of linked web pages that has a common theme or focus is called a Website. The main page that all of the pages on a particular web site are organized around is called the site's "Home Page".

1.4 DIFFERENCE BETWEEN WEB AND INTERNET

The Internet is a worldwide network of computers, linked mostly by telephone lines; the Web is just one of many things (called applications) that can run on the Internet. When you send an email, you're using the Internet: the Net sends the words you write over telephone lines to your friends. When you chat to someone online, you're most likely using the Internet too - because it's the Net that swaps your messages back and forth. But when you update a blog or Google for information to help you write a report, you're using the Web over the Net.

Web is the worldwide collection of text pages, digital photographs, music files, videos, and animations that you can access over the Internet. What makes the Web so special (and, indeed, gives it its name) is the way all this information is connected together. The basic building blocks of the Web are pages of text, like this one - Web pages as we call them. A collection of Web pages on the same computer is called a web site. Every web page (including this one) has highlighted phrases called links (or hypertext links) all over it. Clicking one of these links takes you to another page on a web site or another web site entirely.

1.5 INTRODUCTION TO WEBSITE

A web site is a related collection of WWW files that includes a beginning file called a home page. A company or an individual tells you how to get to their web site by giving you the address of their home page. From the home page, you can get to all the other pages on their site. For example, the Web site for IBM has the home page address of http://www.ibm.com. (The home page address actually includes a specific file name like *index.html* but, as in IBM's case, when a standard default name is set up, users don't have to enter the file name.) IBM's home page address leads to thousands of pages.

The basic idea of the Web is that you can read information that anyone else has stored on a publicly accessible space called their web site. If you're familiar with using computers for word processing, you'll know that when you create a document such as a letter or a resume, it exists on your computer as a file, which you store in a place called a folder (or directory). A web site is simply a collection of interlinked documents, usually stored in the same directory on a publicly accessible computer known as a server. Apart from the main documents (text pages), a web site also contains images or graphic files (photographs, typically stored as JPG files, and artworks, usually stored as GIF or PNG files). So the basic idea of creating a web site involves writing all these text pages and assembling the various graphic files you need. Consequently putting them all together in a folder where other people on Internet can access them.

1.5.1 Static Website

There are many static web sites on the Internet, you may not be able to tell immediately if it is a static or dynamic website, but the chances are, if the site looks basic and is for a smaller company, and simply delivers information without any bells and whistles, it could be a static website. Static websites can only really be updated by someone with knowledge of web site development. Static web sites are the cheapest to develop and host, and many smaller companies still use these to get a web presence. These can be developed solely by using html.

1.5.2 Advantages of Static Websites

- Quick to develop
- Cheap to develop
- Cheap to host

1.5.3 Disadvantages of static websites

- Requires web development expertise to update site
- Site not as useful for the user

- Content can get stagnant

1.5.4 Dynamic Website

Dynamic sites on the other hand can be more expensive to develop initially, but the advantages are numerous. At a basic level, a dynamic web site can give the web site owner the ability to simply update and add new content to the site. For example, news and events could be posted to the site through a simple browser interface. Dynamic features of a site are only limited by imagination. Some examples of dynamic website features could be: content management system, e-commerce system, bulletin/discussion boards, intranet or extranet facilities, ability for clients or users to upload documents, ability for administrators or users to create content or add information to a site (dynamic publishing).

1.5.5 Advantages of dynamic web sites

- Much more functional website
- Much easier to update
- New content brings people back to the site and helps in the search engines
- Can work as a system to allow staff or users to collaborate

1.5.6 Disadvantages of dynamic web sites

- Slower / more expensive to develop
- Hosting costs a little more

CHECK YOUR PROGRESS

- Define the term Internet.
- Write the name of computer networks which was developed initially prior to Internet.
- What is a web site?
- Compare dynamic and static web site.

1.6 HYPER TEXT MARKUP LANGUAGE (HTML)

HTML stands for Hyper Text Markup Language, which is the most widely used language on Web to develop web pages. HTML is a markup language for describing web documents (web pages). HTML was created by Berners-Lee in late 1991. HTML 2.0 was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but the current version is HTML-5 which is an extension to HTML 4.01, and this version was published in 2012. HTML has following characteristics:

- HTML stands for Hyper Text Markup Language
- A markup language is a set of markup tags
- HTML documents are described by HTML tags
- Each HTML tag describes different document content

1.4.1 Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Now the different elements of the HTM can be explained as below:

- The **<!DOCTYPE html>** declaration defines this document to be HTML5
- The text between **<html>** and **</html>** describes an HTML document
- The text between **<head>** and **</head>** provides information about the document
- The text between **<title>** and **</title>** provides a title for the document
- The text between **<body>** and **</body>** describes the visible page content
- The text between **<h1>** and **</h1>** describes a heading
- The text between **<p>** and **</p>** describes a paragraph

Moreover, HTTP is short for Hyper Text Transfer Protocol which is underlying protocol used by the World Wide Web (WWW) and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. It is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

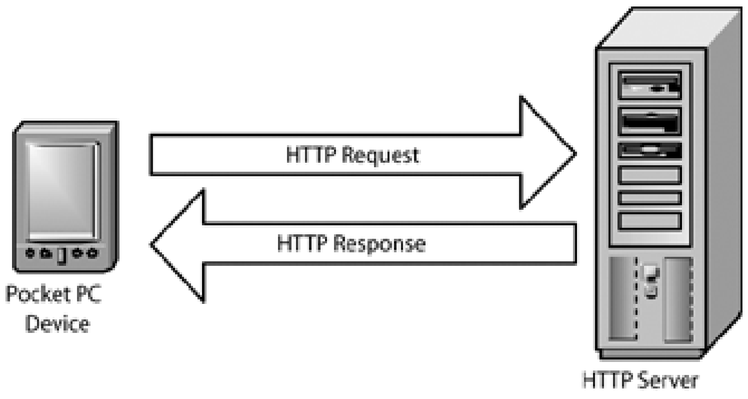


Figure 1.4: Basic Diagram of HTTP

For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server as shown in figure 1.4, directing it to fetch and transmit the requested Web page. The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed.

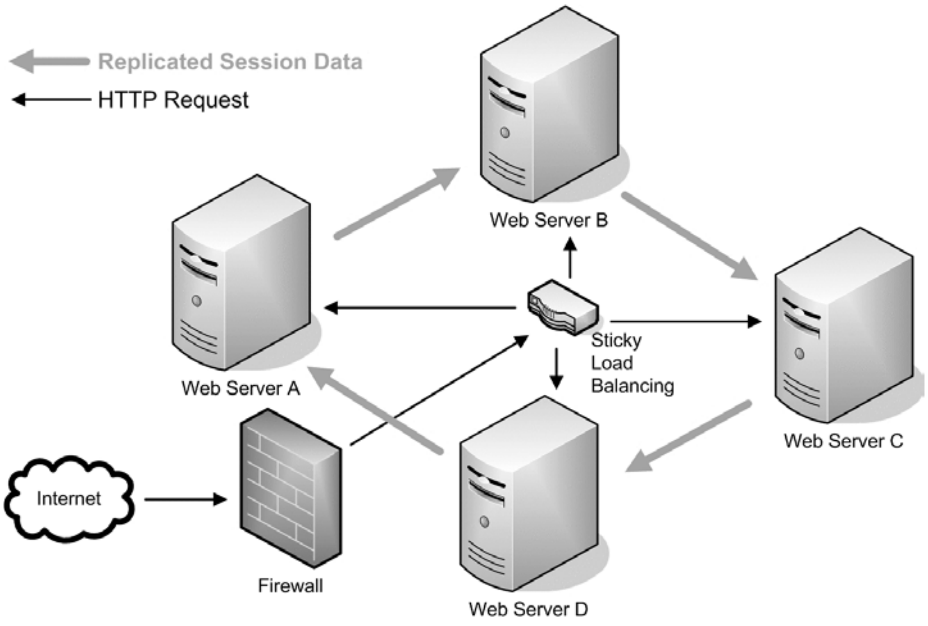


Figure 1.5: Internet & HTTP

1.7 SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

Simple Mail Transfer Protocol or simply SMTP is part of the application layer of the TCP/IP protocol. Using a process called "store and forward," SMTP moves your email on and across networks. It works closely with something called the Mail Transfer Agent (MTA) to send your communication to the right computer and email inbox. SMTP spells out and directs how your email moves from your computer's MTA to an MTA on another computer, and even several computers. Using that "store and forward" feature mentioned before, the message can move in steps from your computer to its destination. At each step, Simple Mail Transfer Protocol is does its job. Lucky for us, this all takes place behind the scenes, and we don't need to understand or operate SMTP.

SMTP is an Internet standard for electronic mail transmission. First defined by RFC 821 in 1982, it was last updated in 2008 with the Extended SMTP additions by RFC 5321—which is the protocol in widespread use today. SMTP by default uses TCP port 25.

SMTP is able to transfer only text—it isn't able to handle fonts, graphics, attachments, etc.—maybe that's why it's called simple. Fortunately, Multipurpose Internet Mail Extensions (MIME) was created to lend a hand. MIME encodes all the non-text content into plain text. In that transformed format, SMTP is coaxed into transferring the data. SMTP sometimes stands for "stop." Most of us don't know this, but our Internet Service Providers typically have a limit to the number of emails we can send out over a certain amount of time. Most of the time, it's limited to a set number per hour or per day.

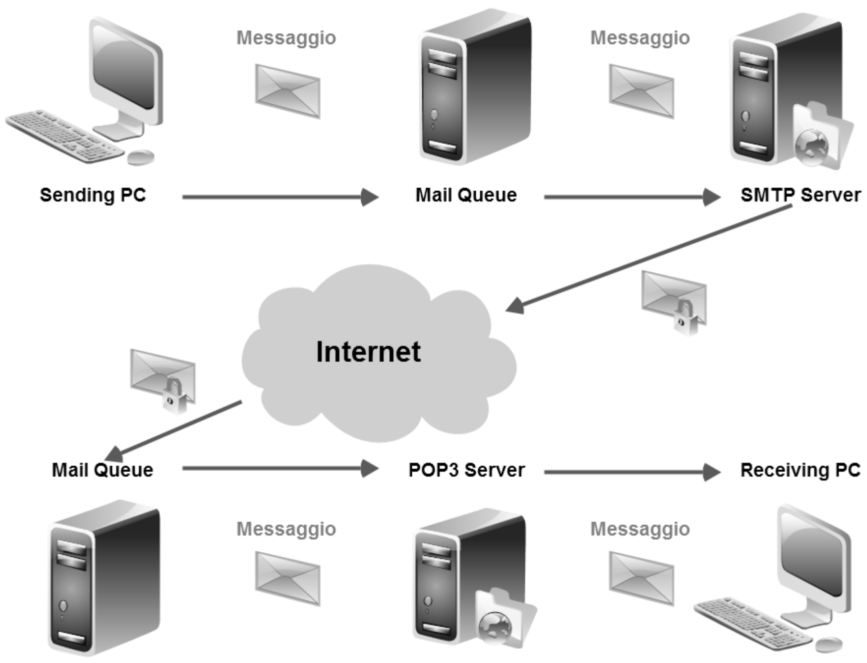


Figure1.6: SMTP server

Each ISP relies on its SMTP to determine (and govern) the email that can be sent out by one connection. (It is a protocol, after all.) For some people who work at home or manage large mailing lists, that could be a problem. After they hit their limit, the ISP will simply stop sending emails. If they think you're a spammer, they might even shut down your account. That email limit varies by ISP. For example, the typical Comcast Cable Internet customer is limited to 1,000 emails per day. (Their business customers have a limit of 24,000 emails daily.) Verizon and AT&T do it differently. They put a limit of 100 on the number of recipients you can have on one sent email.

SMTP is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, send mail is the most widely-used SMTP server for e-mail. A commercial package, Send mail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

1.8 POP3 (POST OFFICE PROTOCOL 3)

POP3 (Post Office Protocol 3) is the most recent version of a standard protocol for receiving e-mail. POP3 is a client/server protocol in which e-mail is received and held for you by your Internet server. POP3, which is an abbreviation for Post Office Protocol 3, is the third version of a widespread method of receiving email. Much like the physical version of a post office clerk, POP3 receives and holds email for an individual until they pick it up. And, much as the post office does not make copies of the mail it receives, in previous versions of POP3, when an individual downloaded email from the server into their email program, there were no more copies of the email on the server; POP automatically deleted them. POP3 makes it easy for anyone to check their email from any computer in the world, provided they have configured their email program properly to work with the protocol.

POP3 has become increasingly sophisticated so that some administrators can configure the protocol to "store" email on the server for a certain period of time, which would allow an individual to download it as many times as they wished within that given time frame. However, this method is not practical for the vast majority of email recipients.

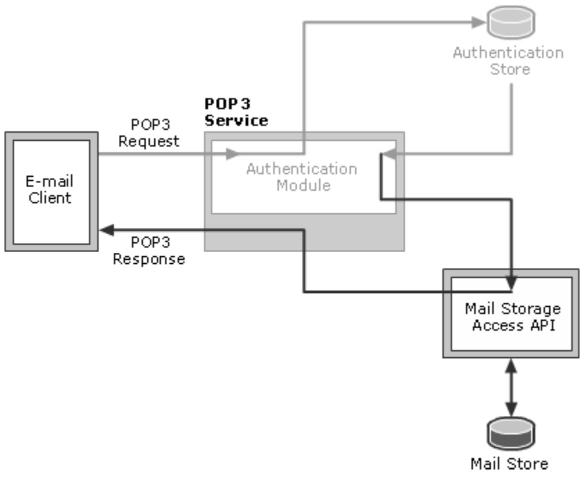


Figure 1.7: Basic Diagram of POP3

While mail servers can use alternate protocol retrieval programs, such as IMAP, POP3 is extremely common among most mail servers because of its simplicity and high rate of success. Although the newer version of POP offers more "features," at its basic level, POP3 is preferred because it does the job with a minimum of errors.

1.6.1 Working with Email Applications

Because POP3 is a basic method of storing and retrieving email, it can work with virtually any email program, as long as the email program is configured to host the protocol. Many popular email programs, including Eudora and Microsoft Outlook, are automatically designed to work with POP3. Each POP3 mail server has a different address, which is usually provided to an individual by their web hosting company. This address must be entered into the email program in order for the program to connect effectively with the protocol. Generally, most email applications use the 110 port to connect to POP3. Those individuals who are configuring their email program to receive POP3 email will also need to input their username and password in order to successfully receive email.

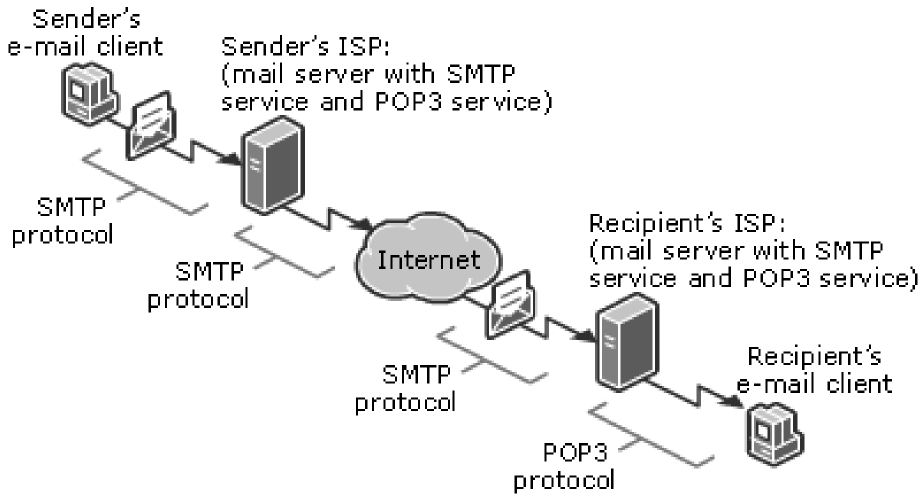

Sender's e-mail client
Sender's ISP: (mail server with SMTP service and POP3 service)
SMTP protocol
SMTP protocol
Internet
Recipient's ISP: (mail server with SMTP service and POP3 service)
SMTP protocol
Recipient's e-mail client
POP3 protocol

Figure 1.8: Working of POP3

1.9 MULTI-PURPOSE INTERNET MAIL EXTENSIONS (MIME)

MIME (Multi-Purpose Internet Mail Extensions) is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol. In 1991, Nathan Borenstein of Bellcore proposed to the IETF that SMTP be extended so that Internet (but mainly Web) clients and servers could recognize and handle other kinds of data than ASCII text. As a result, new file types were added to "mail" as a supported Internet Protocol file type. Servers insert the MIME header at the beginning of any Web transmission. Clients use this header to select an appropriate "player" application for the type of data the header indicates. Some of these players are built into the Web client or browser (for example, all browsers come with GIF and JPEG image players as well as the ability to handle HTML files); other players may need to be downloaded.

Multipurpose Internet Mail Extensions, a specification for formatting non-ASCII messages so that they can be sent over the Internet. Many e-mail clients now support MIME, which enables them to send and receive graphics, audio, and video files via the Internet mail system. In addition, MIME supports messages in character sets other than ASCII.

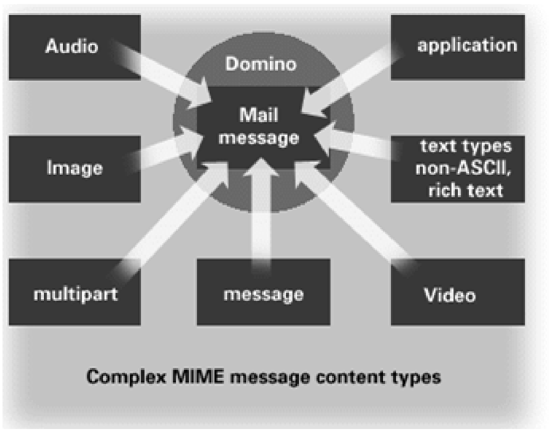

Audio
Domino
application
Mail message
Image
text types non-ASCII, rich text
multipart
message
Video
Complex MIME message content types

Figure 1.9 : Content Categories of MIME

There are many predefined MIME types, such as GIF graphics files and PostScript files. It is also possible to define your own MIME types. In addition to e-mail applications, Web browsers also support various MIME types. This enables the browser to display or output files that are not in HTML format. Multipurpose Internet Mail Extensions (MIME) is one of the open Internet standards that is integrated into Domino 4.6. With the enhancement of MIME, Domino moves toward native MIME and the full support of complex MIME messages. MIME support means more than 7-bit plain ASCII text, which was the only type handled by the original specification (RFC 822) for Internet mail messages. As technology evolved, the original specification became inadequate so MIME was designed to enable mail to handle more complex messages with 8-bit character sets and non-ASCII text, multimedia, images, and application-specific formats. MIME is a draft standard and is described in RFCs 2045, 2046, 2047, 2048, and 2049.

The MIME RFCs describe a group of content-types that were anticipated as elements of mail messages. MIME was also designed as an extensible mechanism that would allow new content-types and other character sets to be added over time without breaking the current structures. With the addition of subtypes and parameters, the content-types declare the general types of data, specify a format for the data type, and indicate whether to display the data in a raw form or in a specific form for each mail message. You will recognize the subtypes and parameters as the familiar browser "helpers" that map the data type to an application for viewing (for example, .avi in Media Player or .pdf in Acrobat Reader).

1.10 INTERNET MESSAGE ACCESS PROTOCOL (IMAP)

In computing, the Internet Message Access Protocol (IMAP) is an Internet standard protocol used by e-mail clients to retrieve e-mail messages from a mail server over a TCP/IP connection. IMAP is defined by RFC 3501. IMAP, a protocol for retrieving e-mail messages. The latest version, IMAP4, is similar to POP3 but supports some additional features. For example, with IMAP4, you can search through your e-mail messages for keywords while the messages are still on mail server. You can then choose which messages to download to your machine. IMAP was developed at Stanford University in 1986.

IMAP (Internet Message Access Protocol) is a standard email protocol that stores email messages on a mail server, but allows the end user to view and manipulate the messages as though they were stored locally on the end user's computing device(s). This allows users to organize messages into folders, have multiple client applications know which messages have been read, flag messages for urgency or follow-up and save draft messages on the server.

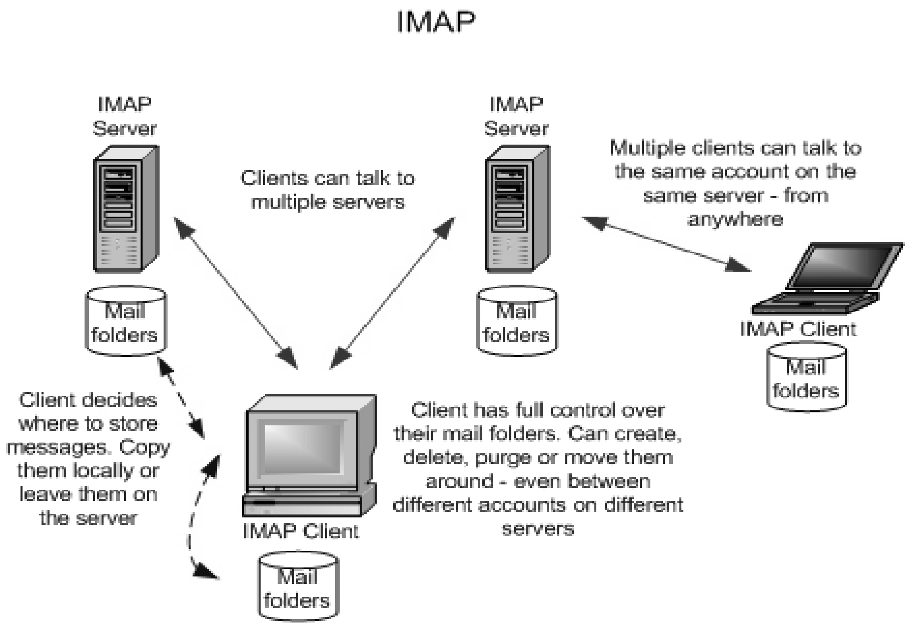

IMAP
IMAP Server
Mail folders
Clients can talk to multiple servers
IMAP Server
Mail folders
Multiple clients can talk to the same account on the same server - from anywhere
IMAP Client
Mail folders
Client decides where to store messages. Copy them locally or leave them on the server
IMAP Client
Mail folders
Client has full control over their mail folders. Can create, delete, purge or move them around - even between different accounts on different servers

Figure 1.10: Basic Working of IMAP

IMAP can be contrasted with another client/server email protocol, Post Office Protocol 3 (POP3). With POP3, mail is saved for the end user in a single mailbox on the server and moved to the end user's device when the mail client opens. While POP3 can be thought of as a "store-and-forward" service, IMAP can be thought of as a remote file server. Most implementations of IMAP support multiple logins; this allows the end user to simultaneously connect to the email server with different devices. For example, the end user could connect to the mail server with his Outlook iPhone app and his Outlook desktop client at the same time. The details for how to handle multiple connections are not specified by the protocol but are instead left to the developers of the mail client. Even though IMAP has an authentication mechanism, the authentication process can easily be circumvented by anyone who knows how to steal a password by using a protocol analyzer because the client's username and password are transmitted as clear text. In an Exchange Server environment, administrators can work around this security flaw by using Secure Sockets Layer (SSL) encryption for IMAP.

CHECK YOUR PROGRESS

- What do you mean by HTML?
- Write a program in HTML to print "Hello World".
- Give the name of different Internet browsers in which you can run HTML program
- How we can run a HTML program?
- Define the role of SMTP.

1.11 SUMMARY

Internet is the network of computer networks which form and act as a single huge network for transport across distances which can be anywhere from the same office to anywhere in the world. In 1970 ARPA

(Advanced Research Project Agency) researched a lot of projects faced with dilemma, which focus on researches involved with communication and you can see the picture of network via Protocol TCP/IP. In 1995 wide network "world wide web" replaced how to transfer document while using Internet.

The Web, or World Wide Web, is basically a system of Internet servers that support specially formatted documents. The documents are formatted in a markup language called HTML (*HyperText Markup Language*) that supports links to other documents, as well as graphics, audio, and video files. This means you can jump from one document to another simply by clicking on hot spots. Not all Internet servers are part of the World Wide Web.

The basic requirements for connecting to the Internet are a computer device, a working Internet line and the right modem for that Internet line. In addition, software programs such as Internet browsers, email clients, Usenet clients and other special applications are needed in order to access the Internet.

HTTP is a communication protocol. It defines mechanism for communication between browser and the web server. It is also called request and response protocol because the communication between browser and server takes place in request and response pairs.

Email is a service which allows us to send the message in electronic mode over the internet. It offers an efficient, inexpensive and real time mean of distributing information among people. E-mail Protocols are set of rules that help the client to properly transmit the information to or from the mail server. Protocols such as SMTP, POP, and IMAP plays a very important role in Internet services like sending data, receiving data etc.

Late 2000 web's mobility has controlled Internet more than 80% of all Internets mobility. Companies which use Internet for trade is called (E-commerce) include advertising of selling, buying, and providing service to customers.- Businessmen and other institutions use Internet with sound and video in conference and forms of any communication that people can get it. Use of E-mail is to communicate between colleagues and other staffs in companies including live broadcast of Television and Radio program. Scientists, Teachers and students use Internet to communicate with each other and doing research or finding necessary documents in time.

1.12 TERMINAL QUESTIONS

1. What do you understand by Internet?
2. Explain the history of Internet.
3. Write the role of HTTP in Internet.
4. Is HTML a necessary language for the Internet?
5. What do you mean by Email? Explain its components.
6. What do you mean by a protocol? Explain different protocols in brief.
7. Write a program in HTML to print the “This is the world of Internet”.
8. Define the meaning of website. How we can develop it?

9. Compare static and dynamic website.
10. Create a web page illustrating text formatting tags.
11. Create a web page to demonstrate font variations.
12. Prepare a sample code to illustrate three types of lists in HTML.
13. Using Nested tables create your Curriculum Vitae.
14. Write the usefulness of IMAP.

Unit 2 : Introduction to JavaScript

Structure

2.0 INTRODUCTION

Nowadays Internet has become the most important requirement of every individual. Without it you cannot live even a single minute. Because every information is available on Internet today is JavaScript is a scripting language for client and server applications. ***JavaScript is not Java.*** JavaScript was developed jointly by Sun Microsystems and Netscape Communications and introduced with Netscape Navigator v2.0. (At the time it was called LiveScript - the name change came later as a marketing ploy to cash in a Java's popularity!). JavaScript is similar in appearance to C++ or Java. That is because JavaScript is *object oriented* and *event driven* (more on those later). The characteristics of JavaScript are: It is the programming language of HTML and the Web, programming makes computers do what you want them to do, easy to learn.

JavaScript commands may be embedded in HTML documents and are interpreted by the client. Microsoft's Internet Explorer (V3.0 and later) also interprets JavaScript; however IE JavaScript and Netscape JavaScript don't always agree. Some web browsers do not support JavaScript at all. It can be checked by running the same webpage in different browsers. Client-side JavaScript is simpler than Java. JavaScript is appealing as a language for giving students a quick introduction to programming concepts and techniques. It is a very popular language to put the client side validations. Java script functions make it very robust and powerful for the web developers.

2.1 OBJECTIVES

After the end of this unit, you will be able to:

- o Know the origin of JavaScript.
- o Distinguish JavaScript from Java.
- o Be able to list features of JavaScript that differentiate it from HTML.
- o Understand the advantages of teaching JavaScript as an introductory programming language.
- o Distinguish between client side and server side applications.
- o Be able to embed JavaScript in a web page using the **<script>...</script>** tags.
- o Know that JavaScript is not supported by all web browsers. Be able to hide JavaScript from nonstandard browsers.

2.2 WHAT IS JAVASCRIPT

JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers. Java Script is the most commonly used as a client side scripting language. This means that Java Script code is written into an HTML page. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it.

The fact that the script is in the HTML page means that your scripts can be seen and copied by whoever views your page. Nonetheless, to your mind this openness is a great advantage, because the flip side is that you can view, study and use any JavaScript you encounter on the WWW. Java Script can be used in other contexts than a Web browser. Netscape created server-side Java Script as a CGI-language that can do roughly the same as Perl or ASP. There is no reason why Java Script couldn’t be used to write real, complex programs. It is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. Java Script has come a long way from its humble origins as a simple interpreted object-oriented language for browser-side scripting of web pages. Its many inadequacies, poor debugging and testing, and its design weaknesses, have now been circumvented by frameworks and libraries.

2.3 JAVA SCRIPT VS JAVA

Basically Java Script is *not* the same as Java. Although the names are much alike, Java Script is primarily a scripting language for use within HTML pages, while Java is a real programming language that does quite different things from JavaScript. In addition Java is much harder to learn. It was developed by Sun for use in pretty much anything that needs some computing power. JavaScript was developed by Brendan Eich, then working at Netscape, as a client side scripting language (even though there's no fundamental reason why it can't be used in a server side environment).

Originally the language was called Live Script, but when it was about to be released Java had become immensely popular. At the last possible moment Netscape changed the name of its scripting language to “JavaScript”. This was done purely for marketing reasons. Worse, Eich was ordered to "make it look like Java". This has given rise to the idea that JavaScript is a "dumbed-down" version of Java. Unfortunately there's not the slightest shred of truth in this story.

Java and JavaScript both descend from C and C++, but the languages (or rather, their ancestors) have gone in quite different directions. You can see them as distantly related cousins. Both are object oriented (though this is less important in JavaScript than in many other languages) and they share some syntax, but the differences are more important than the similarities.

2.4 THE JAVASCRIPT LANGUAGE

JavaScript is not a programming language in strict sense. Instead, it is a scripting language because it uses the browser to do the dirty work. If you command an image to be replaced by another one, JavaScript tells the browser to go do it. Because the browser actually does the work, you only need to pull some strings by writing some relatively easy lines of code. That’s what makes JavaScript an easy language to start with.

But don’t be fooled by some beginner’s luck: JavaScript can be pretty difficult, too. First of all, despite its simple appearance it is a fully fledged programming language: it is possible to write quite complex programs in JavaScript. This is rarely necessary when dealing with web pages, but it is possible. This means that there are some complex programming structures that you’ll only understand after protracted studies. Secondly, and more importantly, there are the browser differences. Though modern web browsers all support JavaScript, there is no sacred law that says they should support exactly the same JavaScript. So basic JavaScript is easy to learn, but when you start writing advanced scripts browser differences (and occasionally syntactic problems) will creep up. If you talk about the Java script approaches then It may be on the top and sometimes It can be used as a platform too (figure-2.1).

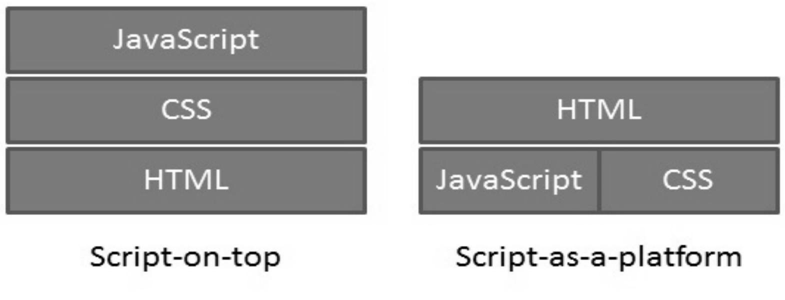


Figure 2.1: Approaches of JavaScript

JavaScript is a programming language that can be included on web pages to make them more interactive. You can use it to check or modify the contents of forms, change images, open new windows and write dynamic page content. You can even use it with CSS to make DHTML (Dynamic HyperText Markup Language). This allows you to make parts of your web pages appear or disappear or move around on the page. JavaScripts only execute on the page(s) that are on your browser window at any set time. When the user stops viewing that page, any scripts that were running on it are immediately stopped. The only exceptions are cookies or various client side storage APIs, which can be used by many pages to store and pass information between them, even after the pages have been closed. The main work of Java script is to control and handle the client requests (figure-2.2).

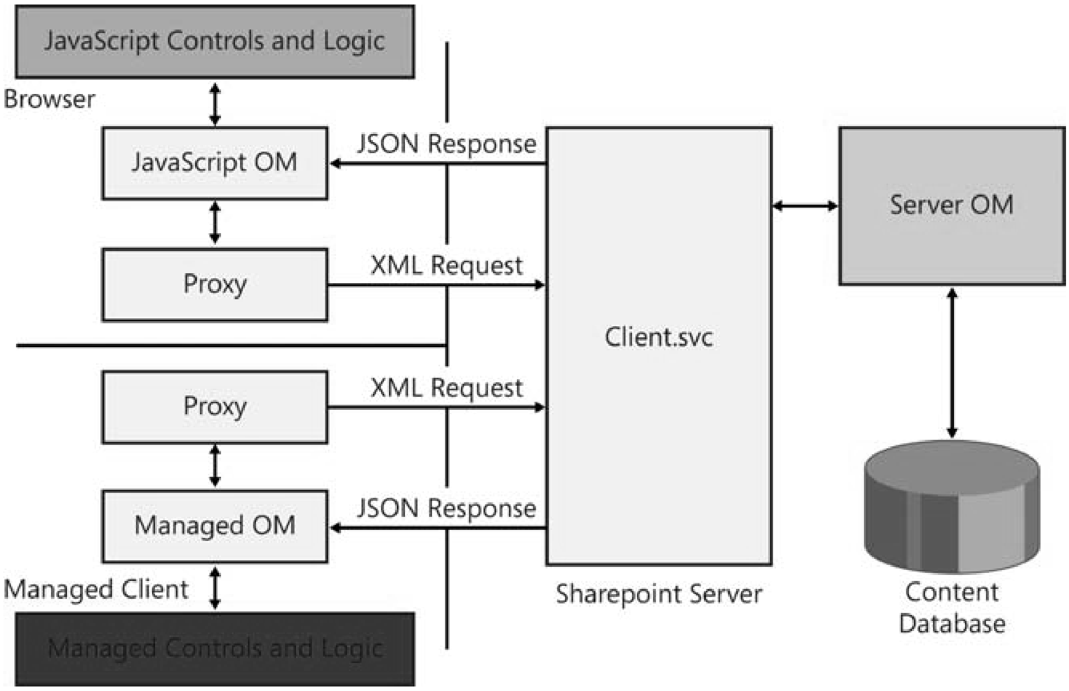


Figure 2.2: Architecture of Java script

Before we go any further, JavaScript has nothing to do with Java. If we are honest, JavaScript, originally nicknamed LiveWire and then LiveScript when it was created by Netscape, should in fact be called ECMAscript as it was renamed when Netscape passed it to the ECMA for standardization.

2.5 Client-Side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content. The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server. JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

CHECK YOUR PROGRESS

- What is JavaScript?
- Discuss the role of JavaScript in website development.
- Give the meaning of script.
- Tell the name of scripting languages other than JavaScript?

2.6 CHARACTERISTICS OF JAVASCRIPT

The main characteristics of JavaScript are listed as follows:

- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is a scripting language - a scripting language is a lightweight programming language.
- A JavaScript is lines of executable computer code.
- A JavaScript is usually embedded directly in HTML pages. JavaScript can put dynamic text into an HTML page - A JavaScript statement like : document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page.
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- Everyone can use JavaScript without purchasing a license.
- JavaScript is supported by all major browsers, like Netscape and Internet Explorer.
- JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.
- JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element.
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server, this will save the server from extra processing.

2.7 ADVANTAGES OF JAVASCRIPT

There are so many advantages of JavaScript. The merits of using JavaScript are:

Less server interaction: You can validate user input before sending the page off to the server. This saves server traffic, which means fewer loads on your server.

Immediate feedback to the visitors: They don't have to wait for a page reload to see if they have forgotten to enter something.

Increased interactivity: You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

Richer interfaces: You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

2.8 LIMITATIONS OF JAVASCRIPT

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features:

- o Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- o JavaScript cannot be used for networking applications because there is no such support available.
- o JavaScript doesn't have any multithreading or multiprocessor capabilities. Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

2.9 PROGRAMMING FUNDAMENTAL OF JAVA SCRIPT

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page. You can place the **<script>** tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the **<head>** tags. The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows:

```
<script ...>
JavaScript code
</script>
```

The script tag takes two important attributes:

Language: This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

Type: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript". So your JavaScript syntax will look as follows:

```
<script language="javascript" type="text/javascript">
Write JavaScript code Here
</script>
```

2.9.1 First JavaScript Code

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a "//-->". Here "//" signifies a comment in JavaScript, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JavaScript code. Next, we call a function document.write which writes a string into our HTML document. This function can be used to write text, HTML, or both. Take a look at the following code:

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

Output:

```
Hello World!
```

2.9.2 Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters. So the identifiers Time and TIME will convey different meanings in JavaScript. It is to be noted here that care must be taken while writing variable and function names in JavaScript.

2.9.3 Comments in JavaScript

JavaScript supports both C-style and C++-style comments. These are explained below:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

2.9.4 How to use comments in JavaScript

```
<script language="javascript" type="text/javascript">
<!--
// This is a comment. It is similar to comments in C++
/*
* This is a multiline comment in JavaScript
* It is very similar to comments in C Programming
*/
//-->
</script>
```

All the modern browsers come with built-in support for JavaScript. Frequently, you may need to enable or disable this support manually. This chapter explains the procedure of enabling and disabling JavaScript support in your browsers: Internet Explorer, Firefox, chrome, and Opera.

2.9.5 JavaScript in Internet Explorer

Here are the steps to turn on or turn off JavaScript in Internet Explorer:

- Follow **Tools -> Internet Options** from the menu.
- Select **Security** tab from the dialog box.
- Click the **Custom Level** button.
- Scroll down till you find the **Scripting** option.
- Select *Enable* radio button under **Active scripting**.
- Finally click OK and come out.

To disable JavaScript support in your Internet Explorer, you need to select **Disable** radio button under **Active scripting**.

2.9.6 JavaScript in Firefox

- Here are the steps to turn on or turn off JavaScript in Firefox:
- Open a new tab -> type **about: config** in the address bar.
- Then you will find the warning dialog. Select **I'll be careful, I promise!**
- Then you will find the list of **configure options** in the browser.
- In the search bar, type **javascript.enabled**.
- There you will find the option to enable or disable javascript by right-clicking on the value of that option -> **select toggle**.

If javascript.enabled is true; it converts to false upon clicking toggle. If javascript is disabled; it gets enabled upon clicking toggle.

2.10 HOW IS JAVASCRIPT CONSTRUCTED?

In many ways JavaScript is like other languages i.e. it also contains variables, data types, literals, pre-defined and user-defined functions, arrays, objects etc. Moreover these are described below:

- The basic part of a script is a variable, literal or object. A variable is a word that represents a piece of text, a number, a boolean true or false value or an object. A literal is the actual number or piece of text or boolean value that the variable represents. An object is a collection of variables held together by a parent variable, or a document component.
- The next most important part of a script is an operator. Operators assign literal values to variables or say what type of tests to perform.
- The next most important part of a script is a control structure. Control structures say what scripts should be run if a test is satisfied.
- Functions collect control structures, actions and assignments together and can be instructed to run those pieces of script as and when necessary.
- The most obvious parts of a script are the actions it performs. Some of these are done with operators but most are done using methods. Methods are a special kind of function and may do things like submitting forms, writing pages or displaying messages.
- Events can be used to detect actions, usually created by the user, such as moving or clicking the mouse, pressing a key or resetting a form. When triggered, events can be used to run functions.
- Lastly and not quite so obvious is referencing. This is about working out what to write to access the contents of objects or even the objects themselves.

2.11 OBJECT BASED SCRIPTING FOR THE WEB

JavaScript is an excellent language to write object oriented web applications. It can support OOP because it supports inheritance through prototyping as well as properties and methods. Many developers cast off JS as a suitable OOP language because they are so used to the class style of C# and Java. Many people don't realize that JavaScript supports inheritance. When you write object-oriented code it instantly gives you power; you can write code that can be re-used and that is encapsulated.

Objects work so well because they act just like real life objects- objects have properties and methods. So if we were talking about a lamp, a property of it may be its height or width, say 12cm. A method of it may be to shine (an action). And when it's shining, its brightness property would be of a greater value than when it wasn't. JavaScript gives you the ability to make your own objects for your own applications. With your objects you can code in events that fire when you want them to, and the code is encapsulated. It can be initialized any amount of times.

2.12 OBJECTS

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers:

- **Encapsulation:** the capability to store related information, whether data or methods, together in an object.
- **Aggregation:** the capability to store one object inside another object.

- **Inheritance:** the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
- **Polymorphism:** the capability to write one function or method that works in a variety of different ways.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object otherwise the attribute is considered a property.

2.12.1 Object Properties

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page. The syntax for adding a property to an object is:

```
objectName.objectProperty = propertyValue;
```

For example:- The following code gets the document title using the "**title**" property of the **document** object.

```
var str = document.title;
```

2.12.2 Object Methods

Methods are the functions that let the object do something or let something be done to it. There is a small difference between a function and a method – at a function is a standalone unit of statements and a method is attached to an object and can be referenced by the **this** keyword. Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

For example: Following is a simple example to show how to use the **write()** method of document object to write any content on the document.

```
document.write ("This is test");
```

2.12.4 User-Defined Objects

All user-defined objects and built-in objects are descendants of an object called **Object**.

The new Operator

The **new** operator is used to create an instance of an object. To create an object, the **new** operator is followed by the constructor method. In the following example, the constructor methods are Object(), Array(), and Date(). These constructors are built-in JavaScript functions as:

```
var employee = new Object();

var books = new Array("C++", "Perl", "Java");

var day = new Date("August 15, 1947");
```

The Object() Constructor

A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called **Object()** to build the object. The return value of the **Object()** constructor is assigned to a variable. The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the **var** keyword.

Example: How to create an Object.

```
<html>
<head>
<title>User-defined objects</title>
<script type="text/javascript">
var book = new Object(); // Create the object
book.subject = "Web Technology"; // Assign properties to the object
book.author = "Krishan Kumar";
</script>
</head>
<body>
<script type="text/javascript">
document.write("Book name is : " + book.subject + "<br>");
document.write("Book author is : " + book.author + "<br>");
</script>
</body>
</html>
```

Output

```
Book name is: Web Technology
Book author is: Krishan Kumar
```

2.13 FUNCTIONS

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. You must have seen functions like **alert()** and **write()** in the earlier chapters. We were using these functions again and again, but they had been written in core JavaScript only once.

JavaScript allows us to write our own functions as well. This section explains how to write your own functions in JavaScript.

2.13.1 Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces. The basic syntax is shown below:

```
<script type="text/javascript">
<!--
function functionname(parameter-list)
{
statements
}
//-->
</script>
```

Example: A function called sayHello that takes no parameters:

```
<script type="text/javascript">
<!--
function sayHello()
{
alert("Hello there");
}
//-->
</script>
```

2.13.2 Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
<head>
<script type="text/javascript">
function sayHello()
{
document.write ("Hello there!");
}
</script>
```

```
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="sayHello()" value="Say Hello">
</form>
<p>Use different text in write method and then try...</p>
</body>
</html>
```

2.14 ARRAY

The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type. To create an Array Object we may have the following syntax:

```
var fruits = new Array( "apple", "orange", "mango" );
```

The **Array** parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295. You can create array by simply assigning values as follows:

```
var fruits = [ "apple", "orange", "mango" ];
```

You will use ordinal numbers to access and to set values inside an array as follows:

fruits[0] is the first element

fruits[1] is the second element

fruits[2] is the third element

2.14.1 Array Properties

Here is a list of the properties of the Array object along with their description

Property	**Description**
constructor	Returns a reference to the array function that created the object.
index	The property represents the zero-based index of the match in the string
input	This property is only present in arrays created by regular expression matches.
length	Reflects the number of elements in an array.
prototype	The prototype property allows you to add properties and methods to an object.

Constructor

Javascript array **constructor** property returns a reference to the array function that created the instance's prototype. Its syntax is as follows:

```
array.constructor
```

length

Javascript array **length** property returns an unsigned, 32-bit integer that specifies the number of elements in an array. Its syntax is as follows:

```
array.length
```

Prototype

The prototype property allows you to add properties and methods to any object (Number, Boolean, String, Date, etc.). Prototype is a global property which is available with almost all the objects. Its syntax is as follows:

```
object.prototype.name = value
```

2.14.2 Array Methods

Here is a list of the methods of the Array object along with their description.

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
map()	Creates a new array with the results of calling a provided function on every element in this array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new

	length of the array.
reduce()	Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
sort()	Sorts the elements of an array.
toString()	Returns a string representing the array and its elements.

CHECK YOUR PROGRESS

- What is Array?
- Write a small program in JavaScript using array to print the table of 7.

2.15 SUMMARY

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

Javascript is a dynamic computer programming language and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the **<head>** tags.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

2.16 TERMINAL QUESTIONS

1. What do you understand by JavaScript?
2. Compare client side and server side JavaScript.
3. Why we call JavaScript as object oriented language?
4. Give the characteristics of JavaScript.
5. Write a program for function in JavaScript to put a validation at client side for login and password.
6. Write a program using JavaScript to find the factorial of given number.
7. Create a java script program to accept the first, middle, last names of user and print them.
8. Write a java script program to add two numbers.
9. Write a java script program to find the factorial of given number.
10. Write a java Script program to print all prime numbers.
11. Write a java script program to sort the array (Use Bubble Sort).
12. Write a java script program to evaluate the following mathematical Expression
    1+2/2!+3/3!+…………+n/n!

U. P. RAJARSHI TANDON OPEN UNIVERSITY

BCA-E9

Bachelor in Computer Applications

Block

2

DYNAMIC HTML

Course Design Committee

Course Preparation Committee

BLOCK INTRODUCTION

Block-2 basically contains four units in all which mainly explain the concepts of dynamic HTML. This block incorporates Unit-3, Unit-4, Unit-4, and Unit-5 and intended with DHTML and its related concepts. DHTML basically is the abbreviation for dynamic Hypertext Markup Language. It is the combination of three main technologies: JavaScript, CSS, and XML. These three techniques have remarkably changed the dynamic features of Web sites. It has also eased web programming approach upto a great extent. Moreover DHTML is the advancement of static HTML and more powerful than HTML. The first thing that we need to clear about DHTML is that it is neither a language like HTML, JavaScript etc. nor a Web standard. It is just a combination of HTML, JavaScript and CSS.

Unit-3 describes the role and functionality of object model and collections. Object models are based on different objects available. DHTML is the advancement of static HTML and more powerful than HTML. Basically the major elements of DHTML include HTML code, scripting languages such as JavaScript, the Document Object Model, Cascading Style Sheets, multimedia filters, and the browser itself.

On the other hand, Unit-4 revolves around the concept of event driven programming or event model. This event basically initiates a script whenever the page is loaded for the client upon his request. It is used in the body tag and is fired when an element finishes its loading. In order to make your sites interactive—to react to user input—you will need to work with such events. These events are directly or indirectly related with the different control like buttons, text box, label, combo box, radio button, list, check box etc. Moreover, the language that supports event is called event driven programming language.

Unit-5 explains about Transitions in DHTML which basically changes from one static/dynamic Web page to another Web page in terms of its special effects for example conversion of colour image to gray scale image. Transitions provide temporary changes. It allows transfer from one page to another with pleasant visual effect for example, random dissolve. Both transitions and filters are programmable and can be programmed to make changes for user requests and hence receiving response after the service from the server. Filters may of many types like flip Filters, mask Filters, image Filters, chroma filters etc. Furthermore these also depend on the browser, as Mozilla Firefox runs such code very smoothly and Internet Explorer might have some problems sometime.

Unit-6 is used for the development of a bridge between front-end and back-end. It provides a detail description of various controls like List Controls, Html Controls, Web Controls. In DHTML, It can be done using two properties of data handling i.e. Data Source Objects (DSOs) and Tabular Data Control (TDC). In DHTML interaction with the table can be done using recordset.

Unit 3 : Object Model and Collections

Structure

3.0 INTRODUCTION

Today Web based applications have become the necessity of the several organizations so that their information could be made global for the betterment of business. Web page is a very important part of any Website. A Web page of any Website should be attractive as well as informative. Web page may include forms, texts, buttons, tables, frames etc. Various technologies have come into the picture for creating the interactive Web pages. DHTML or Dynamic Hypertext Markup Language is the best choice for the most of the software developers irrespective of working on different technologies. It also allows Web developers to control the presentation logic and providing access to the elements of Web pages.

DHTML is the advancement of static HTML and more powerful than HTML. Basically the major elements of DHTML include HTML code, scripting languages such as JavaScript, the Document Object Model, Cascading Style Sheets, multimedia filters, and the browser itself.

DHTML is not a scripting language like JavaScript. It is a browser feature that allows your browser (Netscape Navigator 4.x or higher, or Microsoft Internet Explorer 4.x or higher) to be dynamic. A “dynamic” browser can alter a Web page’s look *after* the document has loaded. Also, DHTML is *not* one particular, technology or set of features. It includes several technologies and describes how these technologies interact.

“Dynamic HTML” is a marketing term coined by both Netscape and Microsoft to describe a series of technologies introduced in the 4.0 versions of their browsers, to enhance the “dynamic” capabilities of those browsers.” DHTML use scripting to retrieve and modify the properties and attributes.

A new concept XHTML has also become very popular in Web development. It is necessary good to understand the basic with DHTML. the W3C created XHTML, which combines the tags and attributes of HTML with the structure of a language called XML.

3.1 OBJECTIVES

- To use the Dynamic HTML Object Model and scripting to create dynamic Web pages.
- To understand the Dynamic HTML object hierarchy.
- To use the *all* and *children* collections to enumerate all of the XHTML elements of a Web page.
- To use dynamic styles and dynamic positioning.
- To use the frames collection to access objects in a separate frame on your Web page.
- To use the navigator object to determine which browser is being used to access your page.

3.2 INTRODUCTION TO DHTML

Dynamic Hyper Text Markup (DHTML) Language cannot really be considered as a markup language for the Internet and also there's no standard fully defining it. DHTML refers to a set of related Internet technology to provide more interactive HTML pages, i.e. the content can be hanged through different mouse movement after the loading the page.

The first thing that we need to clear about DHTML is that it is neither a language like HTML, JavaScript etc. nor a Web standard. It is just a combination of HTML, JavaScript and CSS. It just uses these languages features to build dynamic Web pages. DHTML is a feature of Netscape Communicator 4.0, and Microsoft Internet Explorer 4.0 and 5.0 and is entirely a client-side technology. The implementations of DHTML are:

- The HTML used to present documents.
- Style sheets (CSS) to define a style for multiple objects and their position on the page.
- The Document Object Model (DOM), suggesting a hierarchy of objects to facilitate their manipulation.
- JavaScript (and possibly the VBScript).

3.2.1 Features of DHTML

Once a Web server processes a Web page and sends it to the computer requesting it (called the 'Client' computer) it cannot get any more data from the server unless a new request is made. So to move around this drawback we use Dynamic HTML (DHTML) which is combining HTML and a scripting language that runs on the Client's browser to bring special effects to otherwise static pages. The scripting language that we will be using is JavaScript as most browsers support it. The scripting language can be used to alter HTML data shown (or present but hidden) on the current page by manipulating the properties for the HTML tags involved. Basically some script function is called to execute the required effect when events like 'MouseOver', 'MouseOut', 'Click', etc. occur.

- Simplest feature is making the page dynamic.

- Can be used to create animations, games, applications, provide new ways of navigating through Websites.
- DHTML use low-bandwidth effect which enhance Web page functionality.
- Dynamic building of Web pages is simple as no plug-in is required.
- Facilitates the usage of events, methods and properties and code reuse.

3.2.2 Examples of DHTML

Using Layers

All the data to ever be displayed is loaded into the existing pages in HTML format but is hidden by using the visibility property of HTML layer elements like LAYER, DIV, etc. Now when an event like Mouse Over occurs a JavaScript function is called to show the layer. The layer is then hidden on Mouse Out event.

Using Image Rollovers and Swaps

If you have some knowledge of Web designing then you should know that when creating Image rollovers or image Swaps you usually take some script and include it to the content of your HTML pages. If you look carefully at the HTML code for the images you will find that they call functions on MouseOver and MouseOut events.

Using Dynamic Forms

While surfing you must have noticed how some forms seem to have special functions like: (a) validation of fields (b) text box character limits (c) dynamic displayed lists depending on your selection (d) redirection to other pages (e) manipulating your keyboard input or disabling some keys or browser buttons, etc.

Using Cascading Style Sheets (CSS)

You might be surprised to know that Cascading Style Sheets (CSS) is part of DHTML. A CSS file is just a JavaScript file with the extension .css which is necessary for it to be recognized by the server and tools like Macromedia's Dreamweaver to contain styles.

3.3 OBJECT REFRENCING

In object oriented programming object oriented programming plays a very important role. Any object reference basically works as the alias for an existing variable. Further it is to be noted here that this type of referencing is different from referencing in machine dependent languages like C/C++. Elements id is the key thing in object referencing. The simplest way to reference an element is by using the element’s id attribute. The element is represented as an object. XHTML attributes become properties that can be manipulated by scripting. Basically XHTML is the combination of HTML and XML.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Object Model Introduction -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Object Model Example</title>
<script type = "text/javascript">
<!--
function start()
{
alert( pText.innerText );
pText.innerText = "Thanks for coming.";
}
// -->
</script>
</head>
<body onload = "start()">
<p id = "pText">Welcome to our Web page!</p>
</body>
</HTML>
```

CHECK YOUR PROGRESS

- Define the term DHTML.
- Give the name of elements of DHTML.

3.4 COLLECTIONS ALL AND CHILDREN

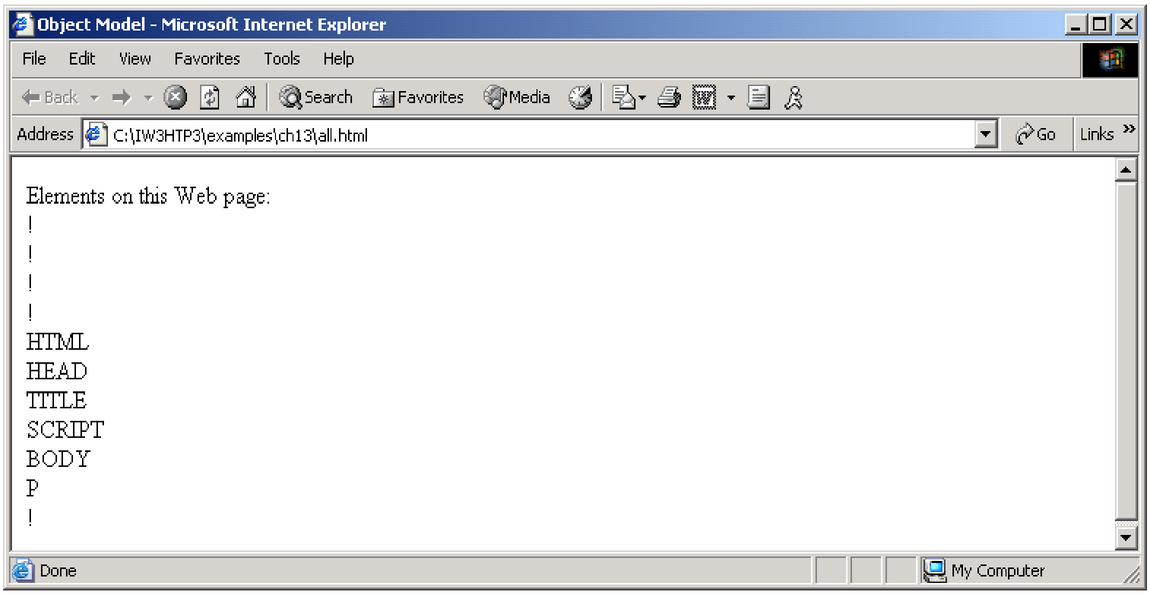
Collections are nothing but the collection of similar type of items i.e. array. Array is well known important feature of almost all programming languages. In DHTML it is known as collections. These are of two types:

- **all:** all the XHTML elements in a document
- **children:** Specific element contains that element's child elements

Example: Use of all

```
<?xml version = "1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <!-- Using the all collection -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Object Model </title>
<script type = "text/javascript">
<!--
function start()
{
for ( var loop = 0; loop < document.all.length; ++loop )
elements += "<br />" + document.all[ loop ].tagName;
pText.innerHTML += elements;
alert( elements );
}
// -->
</script>
</head>
<body onload = "start()">
<p id = "pText">Elements on this page:</p>
</body>
</HTML>
```

Output


Object Model - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch13\all.html
Go Links
Done
My Computer

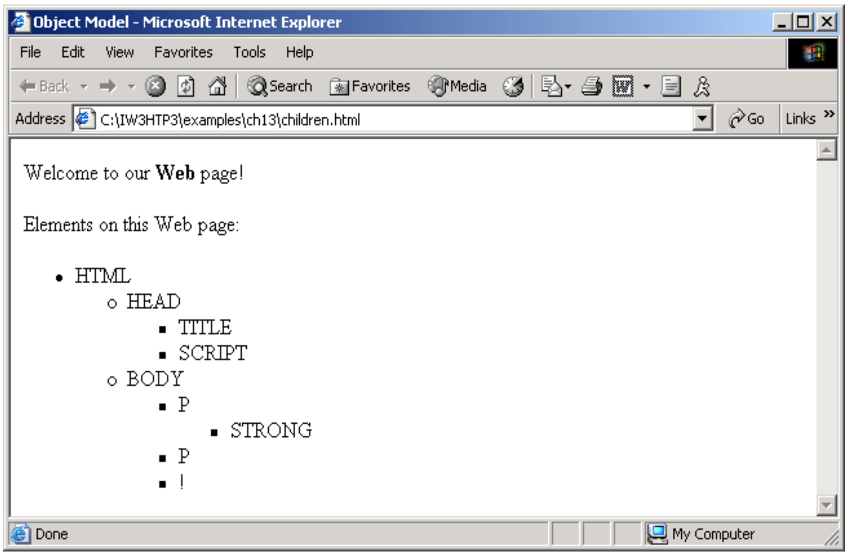
```
Elements on this Web page:
!
!
!
!
HTML
HEAD
TITLE
SCRIPT
BODY
P
!
```

Example: Use of children

```
<?xml version = "1.0"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!—The children collection -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Object Model </title>
<script type = "text/javascript">
<!--
var elements = "<ul>";
function child( object )
{
var loop = 0;
elements += "<li>" + object.tagName + "<ul>";
for ( loop = 0; loop < object.children.length; loop++ )
{
if (object.children[loop].children.length )
child( object.children[ loop ] );
else
elements += "<li>" + object.children[ loop ].tagName + "</li>";
}
elements += "</ul>" + "</li>";
}
// -->
</script>
</head>
<body onload = "child( document.all[4]);
myDisplay.outerHTML += elements;
myDisplay.outerHTML += '</ul>';">
<p>Welcome to our <strong>Web</strong> page!</p>
<p id = "myDisplay">Elements on this page:</p>
</body>
</HTML>
```

Output


Object Model - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address C:\IW3HTP3\examples\ch13\children.html
Done
My Computer

```
Welcome to our Web page!

Elements on this Web page:

  • HTML
      o HEAD
          ▪ TITLE
          ▪ SCRIPT
      o BODY
          ▪ P
              ▪ STRONG
          ▪ P
          ▪ !
```

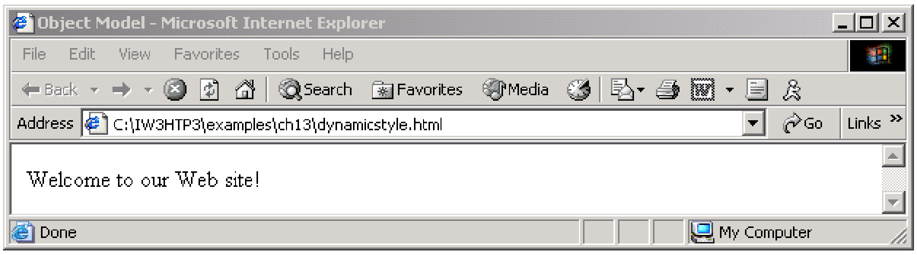
3.5 DYNAMIC STYLE

Sometime we need to change the style of the elements. Element's style can be changed by changing the class's attribute dynamically using DHTML Object Model.

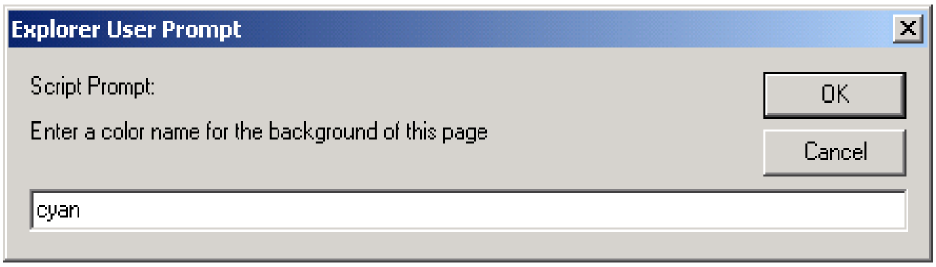
Example

```
<?xml version = "1.0"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!— dynamic styles -->
<html xmlns = "http://www.w3.org/1999/xhtml">

<head>
<title>Object Model </title>
<script type = "text/javascript">
<!--
function start( )
{
var inputColor = prompt("Enter a color name for the " +"background of this page", "" );
document.body.style.backgroundColor = inputColor;
}
// -->
</script>
</head>
<body onload = "start( )";
<p>Welcome to our website!</p>
</body>
</HTML>
```

Output


Object Model - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch13\dynamicstyle.html
Go Links
Done
My Computer

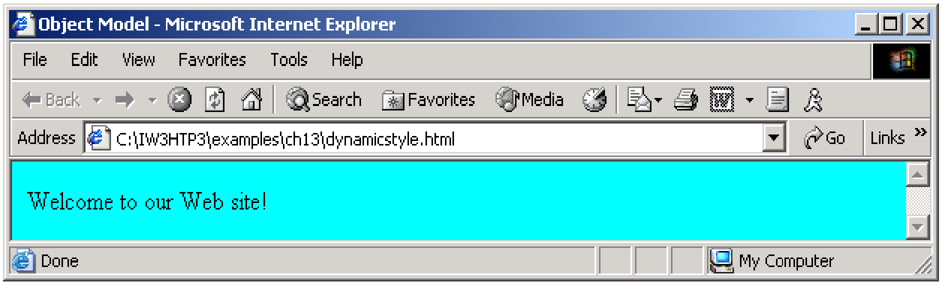
Welcome to our Web site!



Script Prompt:

Enter a color name for the background of this page

cyan


Object Model - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch13\dynamicstyle.html
Go Links
Done
My Computer

Welcome to our Web site!

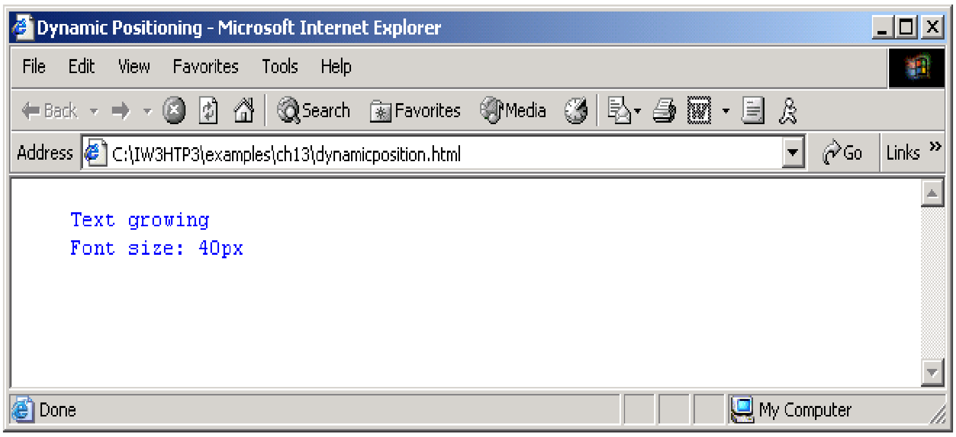
3.6 DYNAMIC POSITION

So far we have not discussed about the position of the elements of Web page. It can be done easily using scripting. For this you need declare an element's CSS position property to be either *absolute* or *relative*. And then Move the element by manipulating any of the *top*, *left*, *right* or *bottom* CSS properties anywhere.

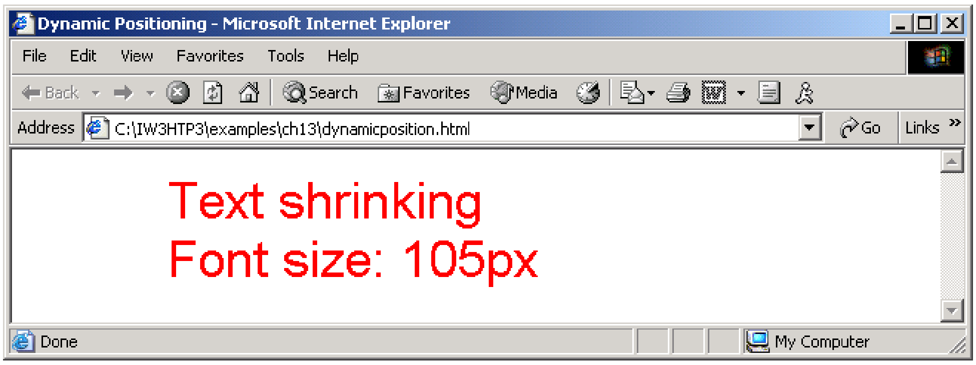
Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!—Dynamic Positioning -->
<html xmlns = "http://www.w3.org/1999/xhtml">

<head>
<title> Dynamic Positioning </title>
```

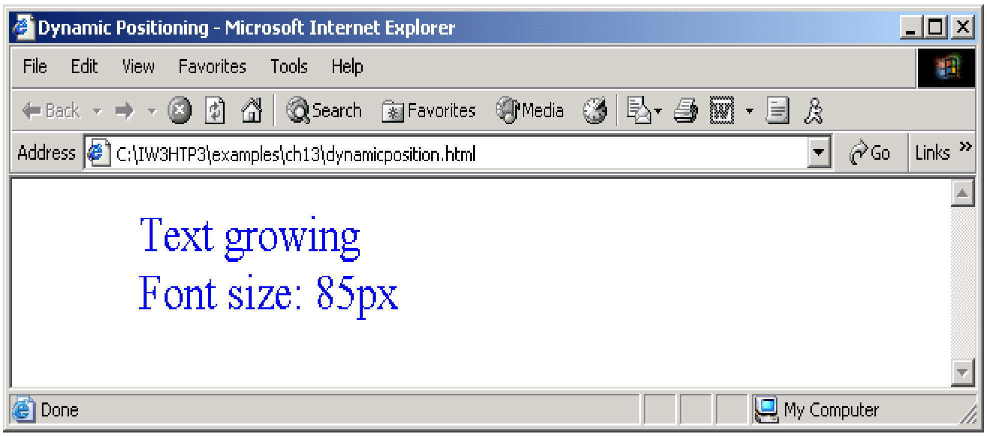
```
<script type = "text/javascript">
<!--
var speed =5;
var count = 10;
var direction = 1;
var firstLine = "Text growing";
var fontStyle = [ "serif", "sans-serif", "monospace" ];
var fontStylecount = 0;

function start()
{
window.setInterval( "run()", 100 );
}
function run()
{
count += speed;
if ( ( count % 200 ) == 0 )
{
speed *= -1;
direction = !direction;
pText.style.color = ( speed < 0 ) ? "red" : "blue" ;
firstLine = ( speed < 0 ) ? "Text shrinking" : "Text growing";
pText.style.fontFamily = fontStyle[ ++fontStylecount % 3 ];
}
pText.style.fontSize = count / 3;
pText.style.left = count;
pText.innerHTML = firstLine + "<br /> Font size: " + count + "px";
}
// -->
</script>
</head>
<body onload = "start()">
<p id = "pText" style = "position: absolute; left: 0; font-family: serif; color: blue"> Welcome!</p>
</body>
</HTML>
```


Dynamic Positioning - Microsoft Internet Explorer
Address C:\IW3HTP3\examples\ch13\dynamicposition.html

Text growing
Font size: 40px


Dynamic Positioning - Microsoft Internet Explorer
Address C:\IW3HTP3\examples\ch13\dynamicposition.html

Text shrinking
Font size: 105px


Dynamic Positioning - Microsoft Internet Explorer
Address C:\IW3HTP3\examples\ch13\dynamicposition.html

Text growing
Font size: 85px

CHECK YOUR PROGRESS

- What do you understand by all and children?
- Compare all and children.
- Define dynamic position. How we can do it?

3.7 USING THE FRAMES COLLECTION

Earlier in HTML frames can be used to partition the html Web page. Referencing elements and objects in different frames by using the frames collection. The main attributes of frameset are: src, name, rows. To use frameset you can use <frameste> ……</frameset> tag.

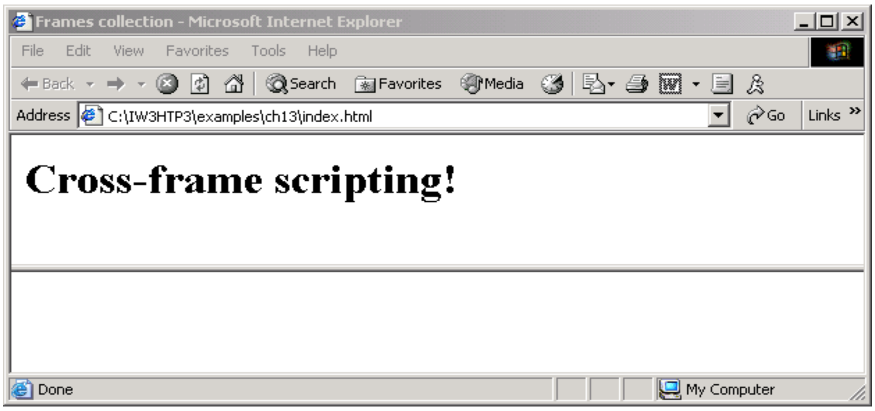
Example: Following gives the code for *Index.html* in which another file *top.html* is being called.

```
<?xml version = "1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <!—Using the Frames Collection -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title> Frames collection </title>
</head>
<frameset rows = "100, *">
<frame src = "top.html" name = "upper" />
<frame src = "" name = "lower" />
</frameset>
</html>
```

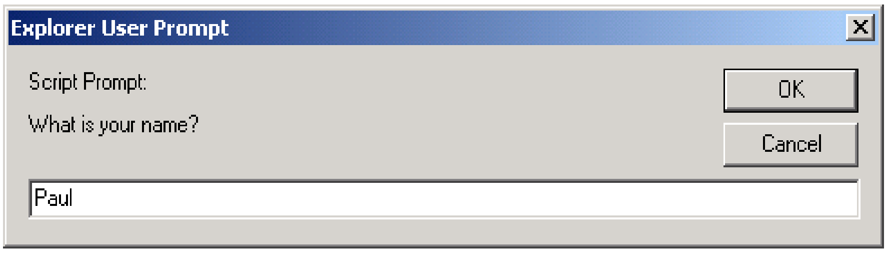
Example : Following program gives the code for *top.html*

```
<?xml version = "1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <!— Cross-frame scripting -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title> The Frames collection </title>
<script type = "text/javascript">
<!—
function start()
{
var text = prompt( "What is your name?", "" );
parent.frames( "lower" ).document.write("<h1>Hello, " +  text + "</
h1>" ); }
// -->
</script>
</head>
<body onload = "start()">
<h1>Cross-frame scripting!</h1>
</body>
</html>
```

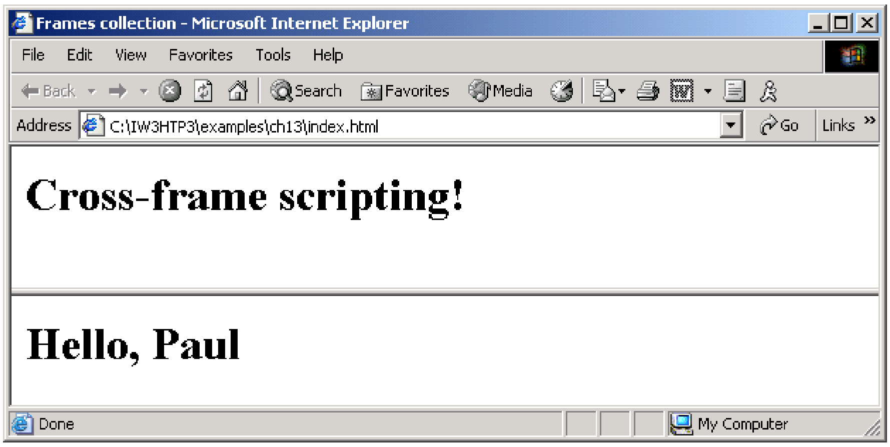
Output


Frames collection - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address C:\IW3HTP3\examples\ch13\index.html
Done
My Computer

Cross-frame scripting!



Script Prompt:
What is your name?
Paul


Frames collection - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address C:\IW3HTP3\examples\ch13\index.html
Done
My Computer

Cross-frame scripting!

Hello, Paul

3.8 NAVIGATOR OBJECT

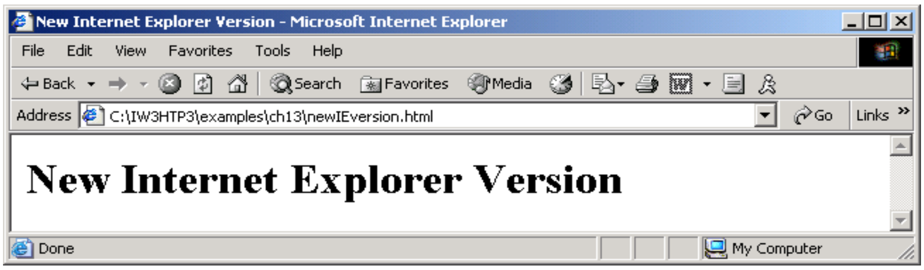
Sometimes you need to know about the Web browser hence you need some code/syntax. If you want to know about the Web browser then you can use navigator object. It also allows Web developers to know what a browser is doing for the user. It is a great help for the developers. Some keywords are location, navigator etc.

Example

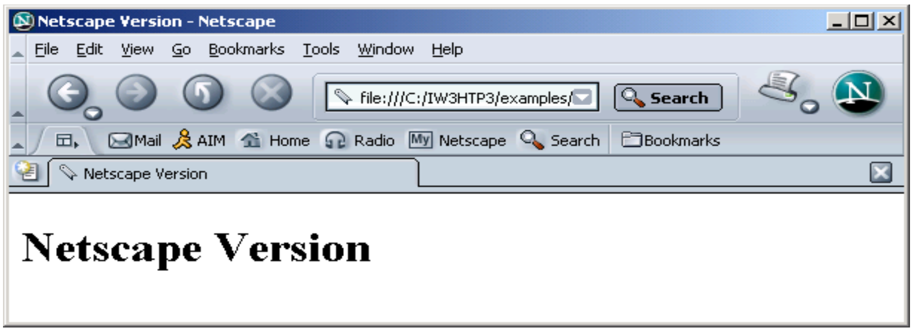
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<!— Using the navigator object -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title> Using the navigator object </title>
<script type = "text/javascript">
<!—
function start()
{
if (navigator.appName=="Microsoft Internet Explorer")
{
if ( navigator.appVersion.substring( 1, 0 ) >= "4" )
document.location = "newIEversion.html";
else
document.location = "oldIEversion.html";
}
else
document.location = "NSversion.html";
}
// -->
function start()
{
if (navigator.appName=="Microsoft Internet Explorer")
{
if ( navigator.appVersion.substring(1,0) >= "4" )
document.location = "newIEversion.html";
else
document.location = "oldIEversion.html";
}
else
document.location = "NSversion.html";
}
</script>
</head>
<body onload = "start()">
<p> Redirecting your browser to the appropriate page, please wait...</p>
</body>
</html>
```

Output


New Internet Explorer Version - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch13\newIEversion.html
Go Links
Done
My Computer

New Internet Explorer Version


Netscape Version - Netscape
File Edit View Go Bookmarks Tools Window Help
file:///C:/IW3HTP3/examples/
Search
Mail AIM Home Radio Netscape Search Bookmarks
Netscape Version

Netscape Version

CHECK YOUR PROGRESS

- What do you understand by frame tag? Write the syntax for it.
- Give the syntax for navigator object.

3.9 SUMMARY

- The *document* object provides access to every element in the XHTML document and allows dynamic modification of the XHTML document.
- The process by which a Web browser assembles and formats an HTML document is called parsing or rendering.
- *The body* tag provides access to the *body* element of an XHTML document.
- *Navigator* contains information about the Web browser, such as the name of the browser, the version of the browser, the operating system on which the browser is running and other information that can help a script writer customize the user's browsing experience.
- Location contains the URL of the rendered document. When this object is set to a new URL, the browser immediately switches (navigates) to the new location.
- Event can be used in an event handler to obtain information about the event that occurred (e.g., the mouse *x-y* coordinates during a mouse event).
- Many objects have an *all* collection that provides access to every element contained in the object. For example, the *body* object's *all* collection provides access to every element in the body element of an XHTML document.
- The *form* element in the XHTML document appears in the collection in the order they were defined in the XHTML document.

- The *frames* tag contains *window* objects that represent each frame in the browser window. Each frame is treated as its own sub-window.

3.10 TERMINAL QUESTIONS

1. Which browsers support DHTML?
2. How will non-supporting browsers handle DHTML?
3. Describe how DHTML work with JavaScript.
4. Distinguish between Static vs. Dynamic HTML.
5. Give the meaning of XHTML. Why it was developed?
6. How do XHTML documents maintain compatibility with HTML documents?
7. Explain how to use the <!DOCTYPE> declaration?
8. What elements are required for all XHTML documents?

Unit 4 : DHTML : Event Model

Structure

4.0 INTRODUCTION

Basically, event model is based on the events that occur during the request of the client on Web. Whenever you click on any control of Web page an event occurs. Event may be of different types like mouse single click, mouse double click, mouse over, mouse up, mouse down, key up, key down etc. Events occur when the user does something with your HTML document: clicks a link, loads a page, or moves the mouse.

In order to make your sites interactive—to react to user input—you will need to work with such events. These events are directly or indirectly related with the different control like buttons, text box, label, combo box, radio button, list, check box etc. Moreover, the language that supports event is called event driven programming language.

Many languages are totally based on events like Visual Basic, Visual C++, ASP, JSP etc. Java script is also very popular for the handling of events. In Java awt (abstract window toolkit) provides various controls which can also help in client side development. Theses event in JavaScript are handled or controlled using several functions or scripts which respond to the requests of users. Consequently the contents of the Website are becoming more and more dynamic. And theses all things are very simple and easy to understand for Web authors. Dynamic HTML along with its contents plays a very crucial role among all the technologies we have been seeing in Web development so far. Some popular events are: onclick, onblurr, onerror, onload, onsubmit, onreset, onmousemove etc.

4.1 OBJECTIVES

In this lesson, at the end of this chapter you will be able to learn the following:

- Firstly to understand the concept of event.
- To understand the event handlers and event bubbling.
- To be able to create event handlers that respond to mouse and keyboard events.
- To be able to use the event object to be made aware of and, ultimately, respond to user actions.
- To understand how to recognize and respond to the most popular events.

4.2 EVENT onclick

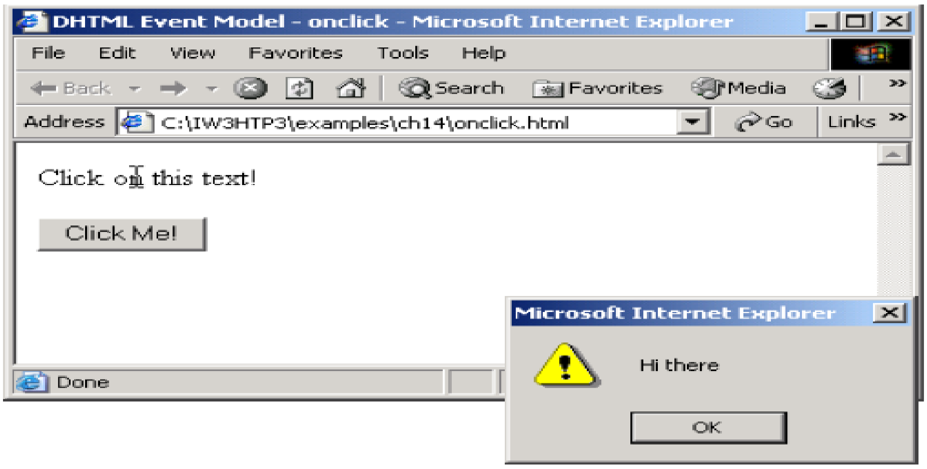
This is the event which is invoked when a user click on a specific control or item. It is very popular traditional event and most often used with buttons.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Demonstrating the onclick event -->

 <html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
<title>DHTML Event Model - onclick</title>
<!-- The for attribute declares the script for a certain element,-->
<!-- and the event for a certain event.-->
<script type = "text/javascript" for = "para"  event = "onclick">
<!--
alert( "Hi there" );
// -->
</script>
</head>
<body>
<!-- The id attribute gives a unique identifier -->
<p id = "para">Click on this text!</p>
```

```
<!-- You can specify event handlers inline -->

<input type = button" value = Click Me!" onclick = "alert( 'Hi again' )" />

</body>

</html>
```

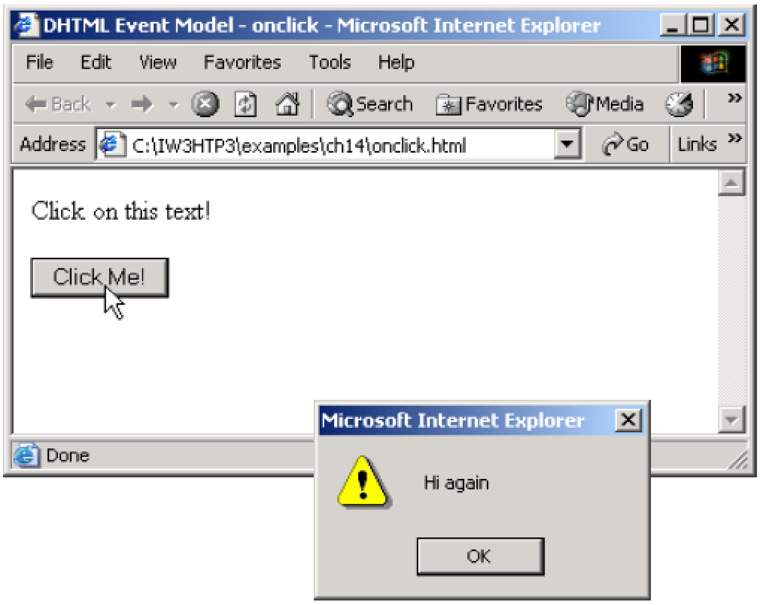
Output


DHTML Event Model - onclick - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch14\onclick.html Go Links
Done
Microsoft Internet Explorer
OK

Click on this text!

Click Me!

Hi there


DHTML Event Model - onclick - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch14\onclick.html Go Links
Done
Microsoft Internet Explorer
OK

Click on this text!

Click Me!

Hi again

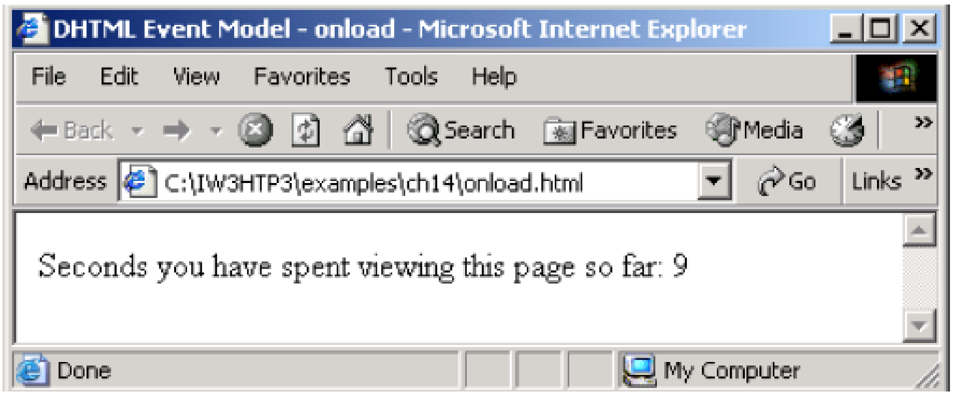
4.3 Event onload

This event basically initiates a script whenever the page is loaded for the client upon his request. It is used in the body tag and is fired when an element finishes its loading.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <!—Showing the onload event -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
 <title>DHTML Event Model - onload</title>
 <script type = "text/javascript">
 <!--
 var seconds = 0;
 function startTimer( )
{
 // 1000 milliseconds = 1 second
 window.setInterval( "updateTime()", 1000 );
 }
 function updateTime( )
{
 seconds++;
 soFar.innerText = seconds;
 }
 // -->
 </script>
</head>
<body onload = "startTimer()">
<p>Seconds you have spent viewing this page so far:
<strong id = “soFar”>0</strong></p>
</body>
</html>
```

Output


DHTML Event Model - onload - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address C:\IW3HTP3\examples\ch14\onload.html
Done
My Computer

Seconds you have spent viewing this page so far: 9

CHECK YOUR PROGRESS

- What do you understand by event model?
- Give the names of controls which are directly responsible for the happening of events.
- Write the syntax for creating a simple submit button.

4.4 EVENT onerror

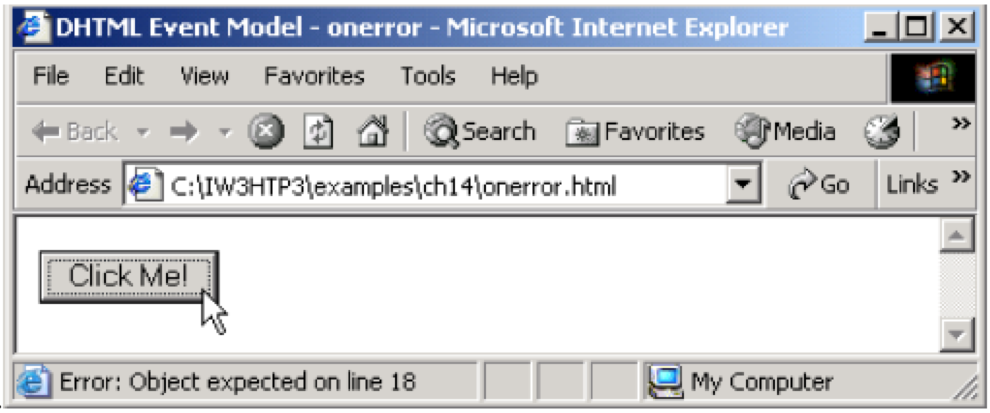
Exception or error is the condition when some problem occurs during the time of compilation or run time. In most of the languages it is known as exception handling and handled using different type of syntax. In DHTML It is done using *onerror* syntax. This event executes the special error-handling code in a script.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Demonstrating the onerror event -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>DHTML Event Model - onerror</title>
<script type = "text/javascript">
<!--
```

```
// Specify that if an onerror event is triggered
// in the window function handleError should execute
window.onerror = handleError;
function doThis()
{
 alert( "hi" ); // alert misspelled, creates an error
}
// The ONERROR event passes three values to the function: the name of
// the error, the url of the file, and the line number.
function handleError( errType, errURL, errLineNum )
{
// Writes to the status bar at the
// bottom of the window.
window.status = "Error: " + errType +
" on line " + errLineNum;
 // Returning a value of true cancels the browser’s reaction.
return true;
 }
 // -->
</script>
</head>
<body>
<input id = "mybutton" type = "button"
value = "Click Me!"  onclick = "doThis()" />
</body>
</html>
```

Output


DHTML Event Model - onerror - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch14\onerror.html Go Links
Click Me!
My Computer

Error: Object expected on line 18

4.5 TRACKING THE MOUSE WITH EVENT onmousemove

Basically this event gives the position of the mouse. It fires repeatedly when the user moves the mouse over the Web page.

Example

```
 <?xml version = "1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Demonstrating the onmousemove event -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>DHTML Event Model - onmousemove event</title>
<script type = "text/javascript">
<!--
function updateMouseCoordinates()
{
coordinates.innerText = event.srcElement.tagName +
"(" + event.offsetX + "," + event.offsetY + ")";
}
// -->
</script>
</head>
<body style = "back-groundcolor: wheat"
```

```
onmousemove = "updateMouseCoordinates()">
<span id = "coordinates">(0, 0)</span><br />
<img src = "Computer.gif" style = "position: absolute;
top: 100; left: 100" alt = "Krishan" />
</body>
</html>
```

4.6 FORM PROCESSING WITH onfocus and onblur

The event *onfocus* fires when element gains focus; and *onblur* event fires when element loses its focus.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Demonstrating the onfocus and onblur events -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
<title>DHTML Event Model - onfocus and onblur</title>
<script type = "text/javascript">
<!--
var helpArray =
[ "Enter your name in this input box.",
"Enter your email address in this input box, " +
"in the format user@domain.",
"Check this box if you liked our site.",
"In this box, enter any comments you would " +
"like us to read.",
"This button submits the form to the " +
"server-side script",
"This button clears the form",
"This textarea provides context-sensitive " +
"help. Click on any input field or use the TAB " +
"key to get more information about the " +
"input field." ];
 function helpText( messageNum )
{
myForm.helpBox.value = helpArray[ messageNum ];
}
 // -->
</script>
</head>
<body>
<form id = "myForm" action = "">
Name:
<input type = "text" name = "name"
onfocus = "helpText(0)" onblur = "helpText(6)" /><br />
Email:
<input type = "text" name = "email"
onfocus = "helpText(1)" onblur = "helpText(6)" /><br />
Click here if you like this site
<input type = "checkbox" name = "like" onfocus =
"helpText(2)" onblur = "helpText(6)" /><br /><hr />
Any comments?<br />
<textarea name = "comments" rows = "5" cols = "45"
onfocus = "helpText(3)" onblur = "helpText(6)">
</textarea><br />
<input type = "submit" value = "Submit" onfocus =
"helpText(4)" onblur = "helpText(6)" />
<input type = "reset" value = "Reset" onfocus =
"helpText(5)" onblur = "helpText(6)" />
<textarea name = "helpBox" style = "position: absolute;
right: 0; top: 0" readonly = “true” rows = "4" cols = "45">
This textarea provides context-sensitive help. Click on any input field or use the Tab key
```

```
to get more information about the input field.
</textarea>
</form>
</body>
</html>
```

CHECK YOUR PROGRESS

- Define the term error. How we can handle it?
- Write the difference between onfocus and onblur.

4.7 FORM PROCESSING WITH onsubmit and onreset

Form is a tag which is very necessary to submit the information from client side to server side. Various JavaScript functions are used with form tag. Normally regular contents are kept within the form tag i.e. <form>. Some attributes of the form tag are: accept, action, charset, class, dir, enctype, id, lang, method, name, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onReset, onSubmit, style, target, title.

You must define at least two special form attributes, which provide the name of the form's processing server and the method by which the parameters are to be sent to the server. A third, optional attribute lets you change how the parameters get encoded for secure transmission over the network. Further onsubmit and onreset are used frequently whenever the information is submitted through the form tag.

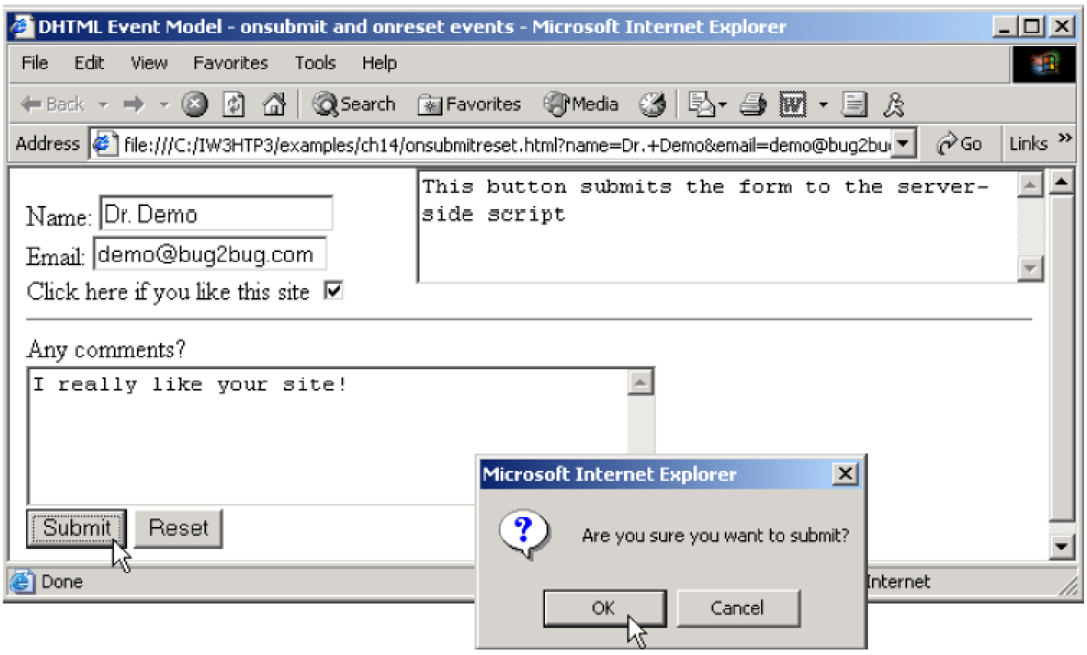
Example

```
 <?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Demonstrating the onsubmit and onreset events -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
 <title>
 DHTML Event Model - onsubmit and onreset events
 </title>
 <script type = "text/javascript">
 <!--
 var helpArray =
 [ "Enter your name in this input box.",
 "Enter your email address in this input box, " +
 "in the format user@domain.",
 "Check this box if you liked our site.",
 "In this box, enter any comments you would " +
 "like us to read.",
 "This button submits the form to the " +
 "server-side script",
 "This button clears the form",
 "This textarea provides context-sensitive " +
 "help. Click on any input field or use the Tab " +
 "key to get more information about " +
 "the input field." ];
function helpText(messageNum)
{
 myForm.helpBox.value = helpArray[ messageNum ];
}
function formSubmit()
{
window.event.returnValue = false;
if ( confirm ( "Are you sure you want to submit?" ) )
window.event.returnValue = true;
}
function formReset()
{
window.event.returnValue = false;
if ( confirm( "Are you sure you want to reset?" ) )
```

```
 window.event.returnValue = true;
 }
 // -->
</script>
</head>
<body>
 <form id = "myForm" onsubmit = "formSubmit()"
onreset = formReset()" action = "">
 Name: <input type = "text" name = "name"
onfocus = "helpText(0)" onblur = "helpText(6)" /><br />
Email: <input type = "text" name = "email"
onfocus = "helpText(1)" onblur = "helpText(6)" /><br />
Click here if you like this site
<input type = "checkbox" name = "like" onfocus =
 "helpText(2)" onblur = "helpText(6)" /><hr />
Any comments?<br />
<textarea name = "comments" rows = "5" cols = "45"
onfocus = "helpText(3)" onblur = "helpText(6)">
</textarea><br />
<input type = "submit" value = "Submit" onfocus =
"helpText(4)" onblur = "helpText(6)" />
<input type = "reset" value = "Reset" onfocus =
 "helpText(5)" onblur = "helpText(6)" />
 <textarea name = "helpBox" style = "position: absolute;
 right:0; top: 0" rows = "4" cols = "45">
This textarea provides context-sensitive help.
Click on any input field or use the Tab key to
get more information about the input field.
```

```
</textarea>

</form>

</body>

</html>
```

Output



DHTML Event Model - onsubmit and onreset events - Microsoft Internet Explorer

Address file:///C:/IW3HTP3/examples/ch14/onsubmitreset.html?name=Dr.+Demo&email=demo@bug2bu

Name: Dr. Demo

Email: demo@bug2bug.com

Click here if you like this site ☑

This button submits the form to the server-side script

Any comments?

I really like your site!

Microsoft Internet Explorer

Are you sure you want to submit?

4.8 EVENT BUBBLING

This is crucial part of the event model. It process whereby events fired in child elements. Normally It bubbles up to their parent elements. DOM elements can be nested inside each other. And somehow, the handler of the parent works even if you click on it’s child due to event bubbling.

Example

```
<?xml version = "1.0"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Disabling event bubbling -->

<html xmlns = "http://www.w3.org/1999/xhtml">

<head>

<title>DHTML Event Model - Event Bubbling</title>
```

```
<script type = "text/javascript">

<!--

function documentClick()

{

alert( "You clicked in the document" );

}

function paragraphClick(value)

{

alert( "You clicked the text" );

if(value)

event.cancelBubble = true;

}

document.onclick = documentClick;

 // -->

</script>

</head>

<body>

<p onclick = "paragraphClick( false )">Click here!</p>

<p onclick = "paragraphClick( true )">Click here, too!</p>

</body>

</html>
```

4.9 SUMMARY

- Event and event model are the two important terms of DHTML. Now a days most of the new technologies revolve around this concept. Importantly without event model a Web author cannot do anything. Today every client request deal with a different kind of request. Some popular events are mouse up, mouse down, mousemove, mouseover, keyup, keydown etc.
- The enents onclick occurs when any of the controls of Web page is clicked and directed to respective task,

- In the same way onload is responsible for the event when it loads a task after clicking a control e.g. a submit button.
- The event *onfocus* fires when element gains focus; and *onblur* event fires when element loses its focus.
- Event bubbling directs an event to its intended target, it works like this: A button is clicked and the event is directed to the button. If an event handler is set for that object, the event is triggered. If no event handler is set for that object, the event bubbles up to the objects parent or <html>.
- Events first are *captured* down to deepest target, then *bubble* up.
- Form tag is a very important and the success of any event also depends on the right use of this tag. You must define at least two special form attributes, which provide the name of the form's processing server and the method by which the parameters are to be sent to the server. A third, optional attribute lets you change how the parameters get encoded for secure transmission over the network.

4.10 TERMINAL QUESTIONS

1. Briefly explain the meaning of event and event model.
2. Write the names of crucial events which commonly occur during surfing or accessing a Website.
3. How can we load the Web page?
4. Compare onfucus and onblur.
5. Give the meaning and significance of event bubbling.
6. What is the role of onerror? Give its syntax and explain.
7. Write the name of some alternative technologies which can be used instead of DHTML which can be used for event handling using various controls.
8. Define the term <form> tag. Explain the important attributes associated with this tag.

Unit - 5 : DHTML: Filters and Transitions

Structure

5.0 INTRODUCTION

Transitions in DHTML are basically changes from one static/dynamic Web page to another Web page in terms of its special effects for example conversion of colour image to gray scale image. Some DHTML visual effects which have transitions are: page transitions (random dissolves, vertical blinds), converting colour images to gray scale images, making letters for different emphasis, adding shadows for 3D effects etc. On the other hand Filters, specified with the cascade style sheet (CSS) property, causes changes that are persistent.

Moreover, Transitions provide temporary changes. It allows transfer from one page to another with pleasant visual effect for example, random dissolve. Both transitions and filters are programmable and can be programmed to make changes for user requests and hence receiving response after the service from the server. Filters may of many types like flip Filters, mask Filters, image Filters, chroma filters etc. Furthermore these also depend on the browser, as Mozilla Firefox runs such code very smoothly and Internet Explorer might have some problems sometime.

5.1 OBJECTIVES

In this lesson, you will learn:

- To use filters to achieve special effects.
- To combine filters to achieve an even greater variety of special effects.
- To be able to create animated visual transitions between Web pages.

o To be able to modify filters dynamically, using DHTML.

5.2 Flip Filters

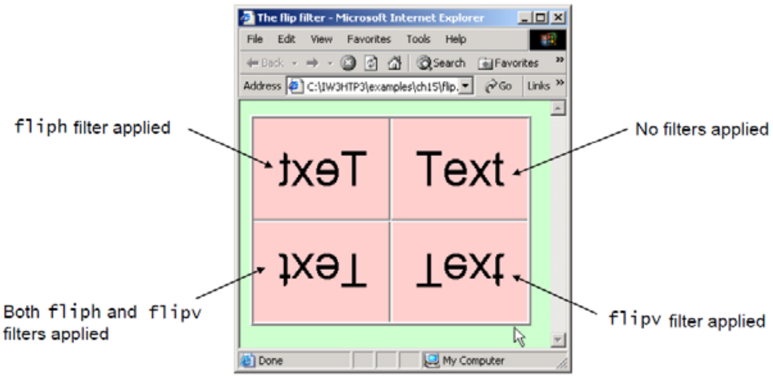
Flip filters are basically of two types i.e. flip horizontal (fliph) and flip vertical (flipv). These flipv and fliph filters mirror text or images vertically and horizontally, respectively.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <!-- Using the flip filters -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
<title>The flip filter</title>
<style type = "text/css">
 body { background-color: #CCFFCC }
table { font-size: 3em; font-family: Arial, sans-serif;
background-color: #FFCCCC; border-style: ridge ;
border-collapse: collapse }
td { border-style: groove; padding: 1ex }
</style>
</head>
<body>
<table>
<tr>
<!-- Filters are applied in style declarations -->
<td style = "filter: fliph">Text</td>
<td>Text</td>
</tr>
<tr>
<!-- More than one filter can be applied at once -->
<td style = "filter: flipv fliph">Text</td>
```

```
<td style = "filter: flipv">Text</td>
</tr>
</table>
</body>
</html>
```

Output


The flip filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\flip.
Go Links
fliph filter applied
No filters applied
Both fliph and flipv filters applied
flipv filter applied
Text
Text
Text
Text
Done
My Computer

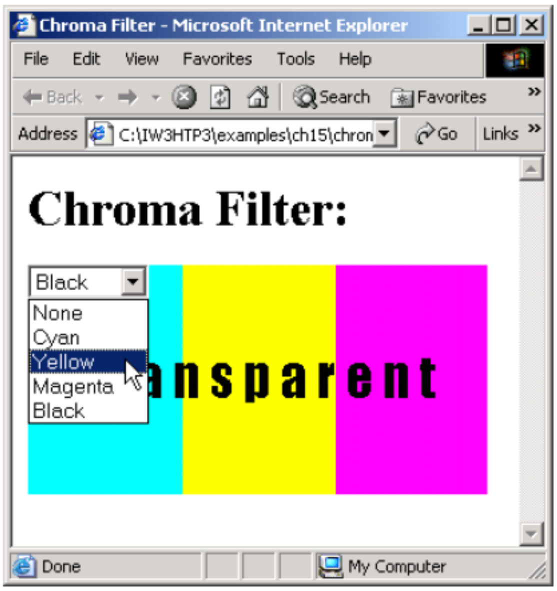
5.3 TRANSPARENCY WITH chroma filter

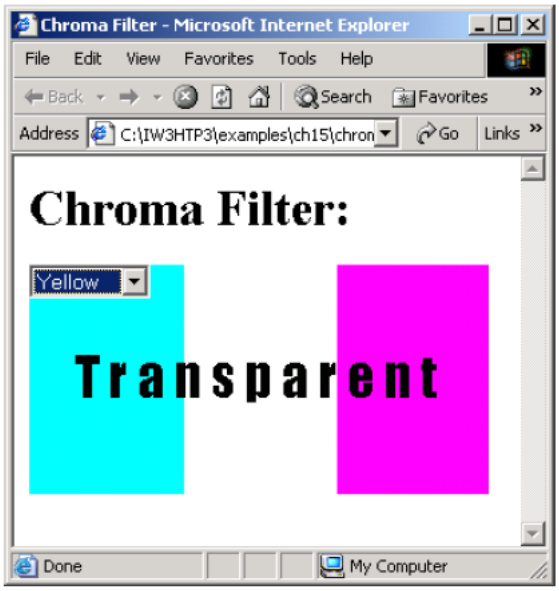
Transparency effects can be added using chroma filters dynamically. For doing these no graphics editor is required for making changes into an image.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Applying transparency using the chroma filter -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Chroma Filter</title>
<script type = "text/javascript">
<!--
 function changecolor( theColor )
 {
 if (theColor) {
 // if the user selected a color, parse the
 // value to hex and set the filter color.
 chromaImg.filters( "chroma" ).color = theColor;
 chromaImg.filters( "chroma" ).enabled = true;
 }
 else // if the user selected "None",
 // disable the filter.
 chromaImg.filters( "chroma" ).enabled = false;
 }
 // -->
 </script>
</head>
<body>
<h1>Chroma Filter:</h1>

 <img id = "chromaImg" src = "trans.gif" style ="position: absolute;
   filter: chroma" alt = "Transparent Image" />
<form action = "">
<!-- The onchange event fires when -->
<!-- a selection is changed -->
<select onchange = "changecolor( this.value )">
<option value = "">None</option>
<option value = "#00FFFF">Cyan</option>
<option value = "#FFFF00">Yellow</option>
<option value = "#FF00FF">Magenta</option>
<option value = "#000000" selected = "selected">Black</option>
</select>
</form>
</body>
</html>
```

Output





5.4 CREATING IMAGE MASKS

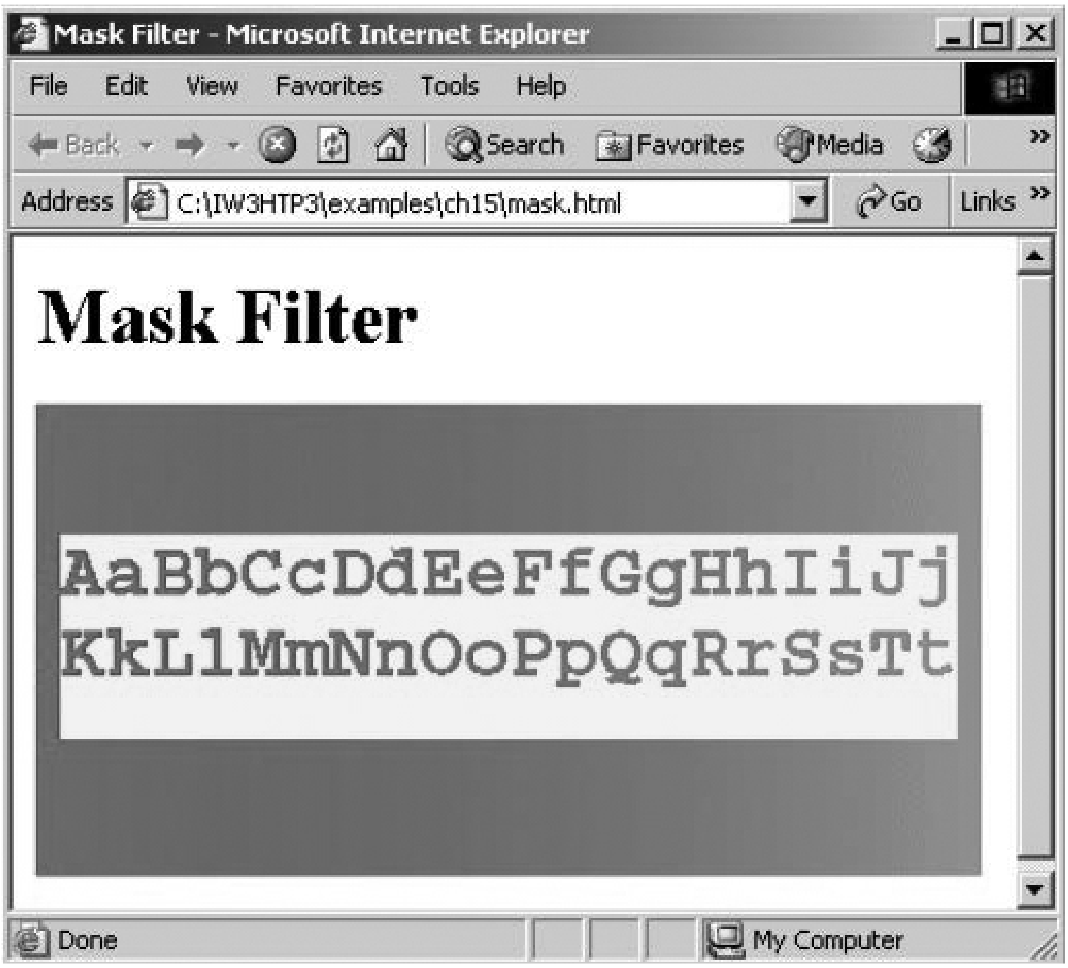
Like other functions in DHTML you can also process images. You can change the background, foreground, etc. Normally background color is a solid color and foreground is transparent to the image or color behind it.

Example

```
<?xml version = "1.0"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Placing a mask over an image -->

<html xmlns = "http://www.w3.org/1999/xhtml">

<head> <title>Mask Filter</title></head>

<body>

<h1>Mask Filter</h1>

<!-- Filter parameters are specified in parentheses, -->

<!-- in the form param1 = value1, param2 = value2, -->

<!-- etc. -->

<div style = "position: absolute; top: 125; left: 20;
```

```
filter: mask( color = #CCFFFF )">

<h1 style = "font-family: Courier, monospace">

AaBbCcDdEeFfGgHhIiJj<br />

KkLlMmNnOoPpQqRrSsTt

</h1>

</div>

<img src = "gradient.gif" width = "400" height = "200" alt = "Image with Gradient Effect" />

</body>

</html>
```

Output


Mask Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\mask.html
Go Links
Done
My Computer

Mask Filter

AaBbCcDdEeFfGgHhIiJj
KkLlMmNnOoPpQqRrSsTt

CHECK YOUR PROGRESS

- o What do you understand by filter in DHTML?
- o Give the syntax for chroma and flip filters.

5.5 ADDING SHADOWS TO TEXT

Like other word processing softwares e.g. MS word, MS Excel etc. shadows or special effects can also be provided in DHTML. For this filters are used to show effects like three-dimensional appearance. It is done using directions for the texts.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Applying the shadow filter -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Shadow Filter</title>
<script type = "text/javascript">
<!--
var shadowDirection = 0;
function start()
{
window.setInterval( "runDemo()", 500 );
}
 function runDemo()
{
 shadowText.innerText = "Shadow Direction: " + shadowDirection % 360;
 shadowText.filters( "shadow" ).direction = ( shadowDirection % 360 );
 shadowDirection += 45;
 }
-->
</script>
</head>
<body onload = "start()">
<h1 id = "shadowText" style = "position: absolute; top: 25;
  left: 25; padding: 10; filter: shadow( direction = 0, color = red )">
Shadow Direction: 0</h1>
</body>
</html>
```

5.6 CREATING GRADIENTS WITH alpha

The slope or gradient is measured in terms of some angle or direction provided by Web author in Web development. This gradual change can be given using alpha filter. Basically gradient in DHTL is gradual progress from starting color to target color. The gradient is divided in four categories according to their styles: Uniform opacity (value 0), Linear gradient (value 1), Circular gradient (value 2), Rectangular gradient (value 3).

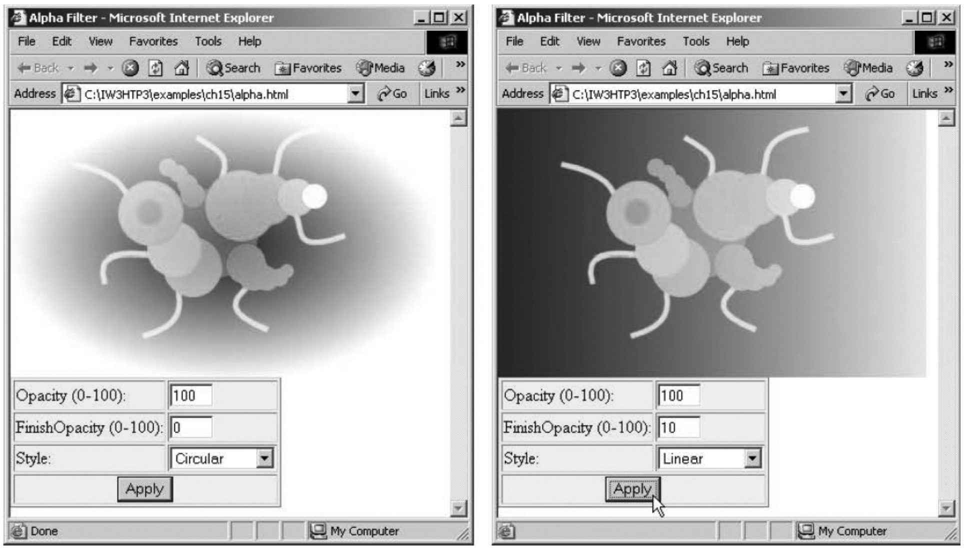
Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Applying the alpha filter to an image -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Alpha Filter</title>
<script type = "text/javascript">
<!--
function run()
 {
```

```
pic.filters( "alpha" ).opacity = opacityButton.value;

pic.filters( "alpha" ).finishopacity =

opacityButton2.value;

pic.filters( "alpha" ).style = styleSelect.value;

}

// -->

</script>

</head>

<body>

<div id = "pic"

style = "position: absolute; left:0; top: 0;

filter: alpha( style = 2, opacity = 100,

finishopacity = 0 )">

<img src = flag.gif" alt = Flag" />

</div>

<table style = "position: absolute; top: 250; left: 0;

background-color: #CCFFCC" border = "1">

<tr>

<td>Opacity (0-100):</td>

<td><input type = "text" id = "opacityButton"

size = "3" maxlength = "3" value = "100" /></td>

</tr>

<tr>

<td>FinishOpacity (0-100):</td>

<td><input type = "text" id = "opacityButton2"

size = "3" maxlength = "3" value = "0" /></td>

</tr>
```

```
 <tr>
 <td>Style:</td>
 <td><select id = "styleSelect">
<option value = "1">Linear</option>
 <option value = "2" selected = "selected"> Circular</option>
 <option value = "3">Rectangular</option>
</select></td>
</tr>
 <tr>
 <td align = "center" colspan = "2">
 <input type = "button" value = "Apply"
 onclick = "run()" />
 </td>
 </tr>
 </table>
 </body>
 </html>
```

Output


Alpha Filter - Microsoft Internet Explorer

Address C:\IW3HTP3\examples\ch15\alpha.html

Opacity (0-100):	100
FinishOpacity (0-100):	0
Style:	Circular
Apply	

Alpha Filter - Microsoft Internet Explorer

Address C:\IW3HTP3\examples\ch15\alpha.html

Opacity (0-100):	100
FinishOpacity (0-100):	10
Style:	Linear
Apply	

5.7 MAKING TEXT glow

A special kind of filter that adds a shine to the text is glow filter. It creates a colorful environment around text.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 <!-- Applying the glow filter -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Glow Filter</title>
<script type = "text/javascript">
<!--
var strengthIndex = 1;
var counter = 1;
var upDown = true;
var colorArray = [ "FF0000", "FFFF00", "00FF00",
 "00FFFF", "0000FF", "FF00FF" ];
function apply()
{
glowSpan.filters( "glow" ).color =
parseInt( glowColor.value, 16 );
glowSpan.filters( "glow" ).strength =
glowStrength.value;
}
function startdemo()
{
window.setInterval( "rundemo()", 150 );
}
function rundemo()
{
if ( upDown ) {
glowSpan.filters( "glow" ).strength =
strengthIndex++;
}
else {
glowSpan.filters( "glow" ).strength =
strengthIndex--;
}
if ( strengthIndex == 1 ) {
upDown = !upDown;
counter++;
glowSpan.filters( "glow" ).color =
parseInt( colorArray[ counter % 6 ], 16 );
}
if ( strengthIndex == 10 )
{
upDown = !upDown;
}
}
 // -->
</script>
</head>
<body style = "background-color: #00AAAA">
<h1>Glow Filter:</h1>
 <span id = "glowSpan" style = "position: absolute;
left: 200;top: 100; padding: 5; filter: glow(color = red,
strength = 5 ); font-size: 2em"> Glowing Text </span>
<table border = "1" style = "background-color: #CCFFCC">
<tr>
<td>Color (Hex)</td>
<td><input id = "glowColor" type = "text" size = "6"
maxlength = "6" value = "FF0000" /></td>
</tr>
 <tr>
 <td>Strength (1-255)</td>
<td><input id = "glowStrength" type = "text" size = "3" maxlength = "3" value = "5" />
</td>
</tr>
```

```
<tr>
<td colspan = "2">
<input type = "button" value = "Apply" onclick = "apply()" />
<input type = "button" value = "Run Demo" onclick = "startdemo()" /></td>
</tr>
</table>
</body>
</html>
```

CHECK YOUR PROGRESS

- What do you understand by shadowing?
- Give the meaning and syntax of creating gradients.

5.8 CREATING MOTION WITH blur

The blur filter creates an illusion of motion by blurring text or images in a certain direction. It has three attributes:

Add : Adds a copy of the original image over the blurred image

Direction : Determines in which direction the blur filter is applied

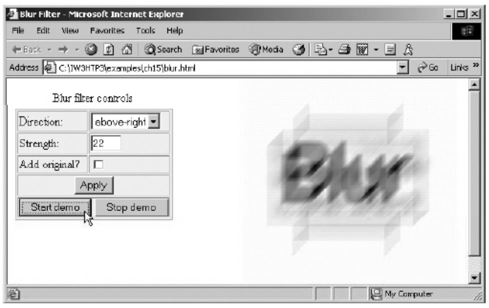
Strength : Determines how strong the blurring effect is

Example

```
 <?xml version = "1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- The blur filter -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
 <title>Blur Filter</title>
 <script type = "text/javascript">
 <!--
 var strengthIndex = 1;

 var blurDirection = 0;

 var upDown = 0;

 var timer;

function reBlur()

{

blurImage.filters( "blur" ).direction =  document.forms( "myForm" ).Direction.value;

blurImage.filters( "blur" ).strength =  document.forms( "myForm" ).Strength.value;

blurImage.filters( "blur" ).add = document.forms( "myForm" ).AddBox.checked;

}

function startDemo()

{

timer = window.setInterval( "runDemo()", 5 );

}

function runDemo( )

{

document.forms( "myForm" ).Strength.value = strengthIndex;

document.forms( "myForm" ).Direction.value = ( blurDirection % 360 );

if ( strengthIndex == 35 || strengthIndex == 0 )

upDown = !upDown;

blurImage.filters( "blur" ).strength =

( upDown ? strengthIndex++ : strengthIndex-- );

if ( strengthIndex == 0 )

blurImage.filters( "blur" ).direction =

( ( blurDirection += 45 ) % 360 );

}

 // -->
```

```
</script>
</head>
<body>
<form name = myForm" action = "">
 <table border = "1" style = "background-color: #CCFFCC">
 <caption>Blur filter controls</caption>
 <tr>
<td>Direction:</td>
<td><select name = "Direction">
<option value = "0">above</option>
 <option value = "45">above-right</option>
 <option value = "90">right</option>
 <option value = "135">below-right</option>
 <option value = "180">below</option>
 <option value = "225">below-left</option>
 <option value = "270">left</option>
 <option value = "315">above-left</option>
 </select></td>
 </tr>
 <tr>
 <td>Strength:</td>
<td><input name = "Strength" size = "3" type = "text" maxlength = "3" value = "0" /></td>
 </tr>
<tr>
 <td>Add original?</td>
 <td><input type = "checkbox" name = "AddBox" /></td>
 </tr>
 <tr>

 <td align = "center" colspan = "2">

 <input type = "button" value = "Apply"  onclick = "reBlur();" /></td>

 </tr>

 <tr>

 <td colspan = "2">

 <input type = "button" value = "Start demo"

 onclick = "startDemo();" />

 <input type = "button" value = "Stop demo"

 onclick = "window.clearInterval( timer );" /></td>

 </tr>

 </table>

 </form>

<div id = "blurImage" style = "position: absolute;

 top: 0; left: 300; padding: 0; filter: blur(

 add = 0, direction = 0, strength = 0 );

background-color: white;">

 <img align = "middle" src = "shapes.gif" alt = "Shapes" />

</div>

</body>

</html>
```

Output



Blur Filter - Microsoft Internet Explorer

Address: C:\IW3HTP3\examples\ch15\blur.html

Blur filter controls

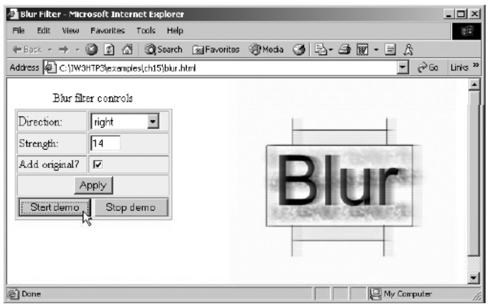
Direction: above-right

Strength: 22

Add original? ☐

Apply

Start demo Stop demo



Blur Filter - Microsoft Internet Explorer

Address: C:\IW3HTP3\examples\ch15\blur.html

Blur filter controls

Direction: right

Strength: 14

Add original? ☑

Apply

Start demo Stop demo

5.9 SUMMARY

- Flip filters are basically of two types i.e. flip horizontal (fliph) and flip vertical (flipv). These flipv and fliph filters mirror text or images vertically and horizontally, respectively.
- Transparency effects can be added using chroma filters dynamically. For doing these no graphics editor is required for making changes into an image.
- Like other functions in DHTML you can also process images. You can change the background, foreground, etc. Normally background color is a solid color and foreground is transparent to the image or color behind it.
- The gradient is divided in four categories according to their styles: Uniform opacity (value 0), Linear gradient (value 1), Circular gradient (value 2), Rectangular gradient (value 3).
- A special kind of filter that adds a shine to the text is glow filter. It creates a colorful environment around text.
- The blur filter creates an illusion of motion by blurring text or images in a certain direction.

5.10 TERMINAL QUESTIONS

1. What do you understand by filter?
2. Why we need filter and transition?
3. Give the names and respective syntax of all the filters.
4. Define the term transition.
5. Show that how you can use flip filter?
6. Write a program showing shadowing in a “Hello World”.
7. Explain gradient with the help of suitable syntax.
8. Write the meaning of blurImage.

Unit - 6 : DHTML: Data Binding with Tabular Data Control

Structure

6.0 INTRODUCTION

When you go to for the development of the Web based programs, you need to create a bridge between frontend and back end. For this, most often, you need to bind the data with database tables. A table or relation in RDBMS is the combination of row/tuple and columns/attributes. Normally data binding is the state of the data when it is present in some other form. For example at compile time it is static binding and at run time it is dynamic binding. Actually data do not reside on the server for a long time. It can be maintained on client side. Ultimately you can avoid server activity and delays caused by networks. In DHTML, It can be done using two properties of data handling i.e. Data Source Objects (DSOs) and Tabular Data Control (TDC). In DHTML interaction with the table can be done using recordset keyword.

6.1 OBJECTIVES

In this lesson, you will learn:

- To understand Dynamic HTML's notion of data binding and how to bind data to XHTML elements.
- To be able to sort and filter data directly on the client without involving the server.
- To be able to bind a table and other XHTML elements to data source objects (DSOs).
- To be able to filter data to select only records appropriate for a particular application.
- To be able to navigate backward and forward through a database with the Move methods.

6.2 SIMPLE DATA BINDING

Normally a table is the key element of any database. Data is present in the form of rows and columns. For a data file four characteristics are crucial: header row, text qualifiers, field delimiter, Recordset. The header row gives the names of the attributes below it. Text qualifiers (@) enclose data in each field. Field delimiter (|) separate each field. And Recordset is directly deal with data of a table.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Simple data binding and recordset manipulation -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Intro to Data Binding</title>
<!-- This object element inserts an ActiveX control -->
<!-- for handling and parsing our data. The PARAM -->
<!-- tags give the control starting parameters -->
<!-- such as URL. -->
<object id = "Colors"
classid = "CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value =
"HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
 <param name = "FieldDelim" value = "|" />
 </object>
<script type = "text/javascript">
 <!--
 var recordSet = Colors.recordset;
 function displayRecordNumber()
 {
```

```
if (!recordSet.EOF)
 recordNumber.innerText =  recordSet.absolutePosition;
 else
 recordNumber.innerText = " ";
 }
 function forward()
 {
 recordSet.MoveNext();
 if ( recordSet.EOF )
 recordSet.MoveFirst();
 colorSample.style.backgroundColor = colorRGB.innerText;
 displayRecordNumber();
 }
 // -->
 </script>
 </head>
 <body onload = "reNumber()" onclick = "forward()">
<h1>XHTML Color Table</h1>
<h3>Click anywhere in the browser window to move forward in the recordset.</h3>
<p>
<strong>Color Name: </strong>
<span id = "colorId" style = "font-family: monospace"
 datasrc = "#Colors" datafld = "ColorName">
</span>
<br />
<strong>Color RGB Value: </strong>
 <span id = "colorRGB" style = "font-family: monospace"
 datasrc = "#Colors" datafld = "ColorHexRGBValue">
 </span>
```

```
<br />
Currently viewing record number
<span id = "recordNumber" style = "font-weight: 900">
</span>
<br />
<span id = "colorSample" style = "background-color: aqua;
 color: 888888; font-size: 30pt">Color Sample</span></p>
</body>
</html>
```

Output

6.3 MOVING WITHIN A Recordset

If you want to move through the recordset, can move using the JavaScript. For this some functions like MoveFirst(),MovePrevious, MoveLast() are used in DHTML frequently.

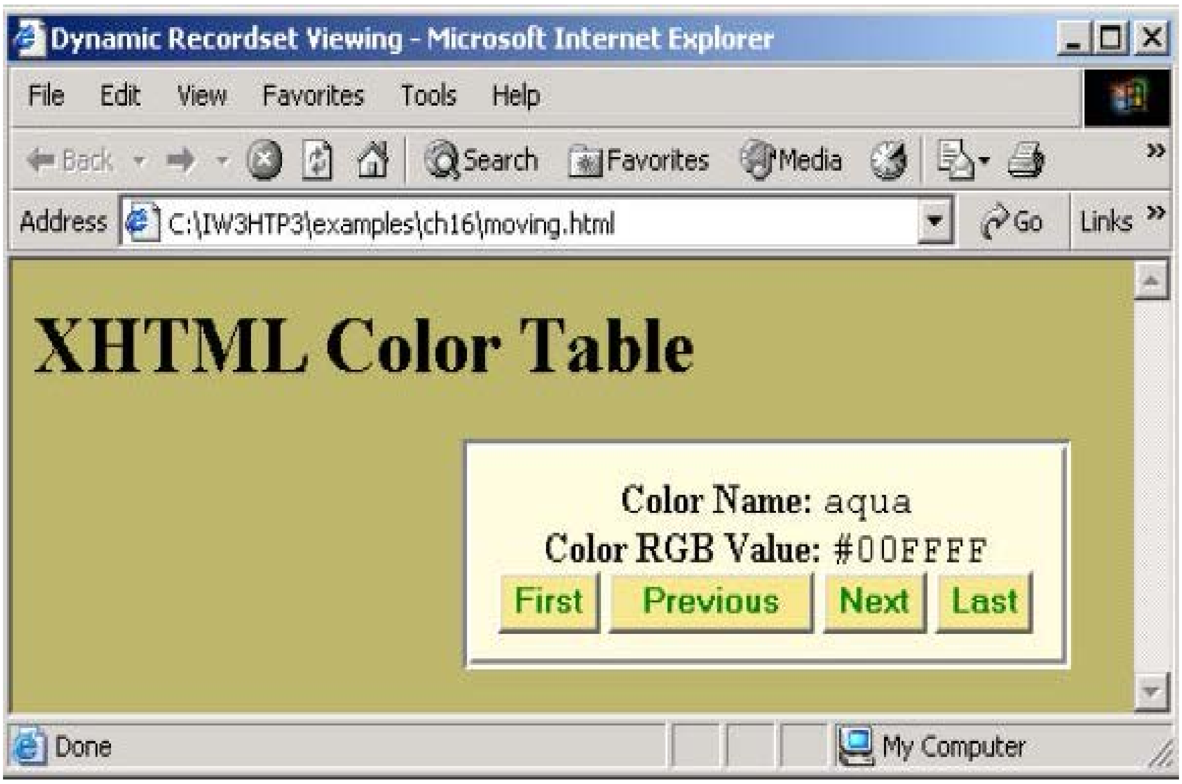
Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Moving through a recordset -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Dynamic Recordset Viewing</title>
<object id = "Colors"
classid = "CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value =
"HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
```

```
<param name = "FieldDelim" value = "|" />
</object>
<script type = "text/javascript">
<!--
var recordSet = Colors.recordset;
function update()
{
h1Title.style.color = colorRGB.innerText;
}
function move( whereto )
{
 switch ( whereTo ) {
case "first":
recordSet recordSet.MoveFirst();
update();
break;
// If recordset is at beginning, move to end.
case "previous":
recordSet.MovePrevious();
if(recordSet.BOF)
recordSet.MoveLast();
update();
 break;
// If recordset is at end, move to beginning.
case "next":
recordSet.MoveNext();
if ( recordSet.EOF )
recordSet.MoveFirst();
update();
break;
case "last":
recordSet.MoveLast();
update();
break;
}
}
// -->
</script>
<style type = "text/css">
input { background-color: khaki;
color: green;
font-weight: bold }
</style>
</head>
<body style = "background-color: darkkhaki">
<h1 style = "color: black" id = "h1Title">
XHTML Color Table</h1>
<span style = "position: absolute; left: 200; width: 270;
border-style: groove; text-align: center;
background-color: cornsilk; padding: 10">
<strong>Color Name: </strong>
<span id = "colorName" style = "font-family: monospace"
datasrc = "#Colors" datafld = "ColorName">ABC</span>
<br />
<strong>Color RGB Value: </strong>
<span id = "colorRGB" style = "font-family: monospace"
datasrc = "#Colors" datafld = "ColorHexRGBValue">ABC
```

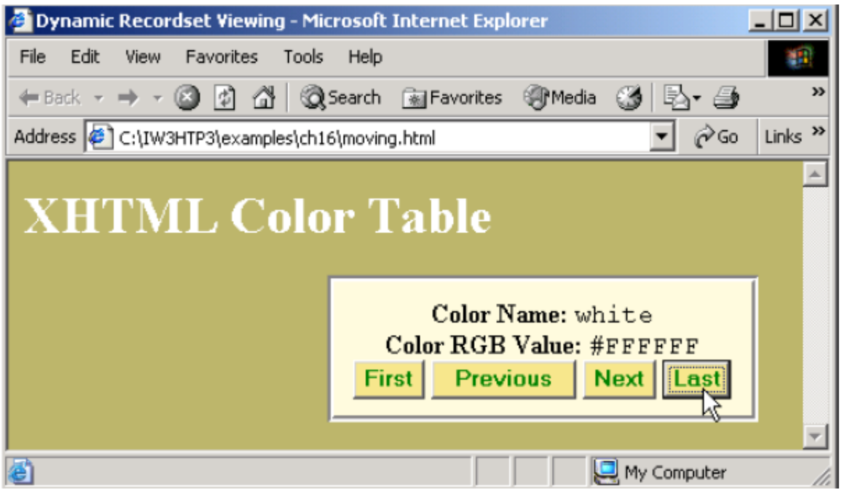
```
</span><br />
<input type = "button" value = "First"
onclick = "move( 'first' );" />
<input type = "button" value = "Previous"
onclick = "move( 'previous' );" />
<input type = "button" value = "Next"
onclick = "move( 'next' );" />
<input type = "button" value = "Last"
onclick = move( last' );" />
</span>
</body>
</html>
```

Output


Dynamic Recordset Viewing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch16\moving.html
Go Links
Done
My Computer

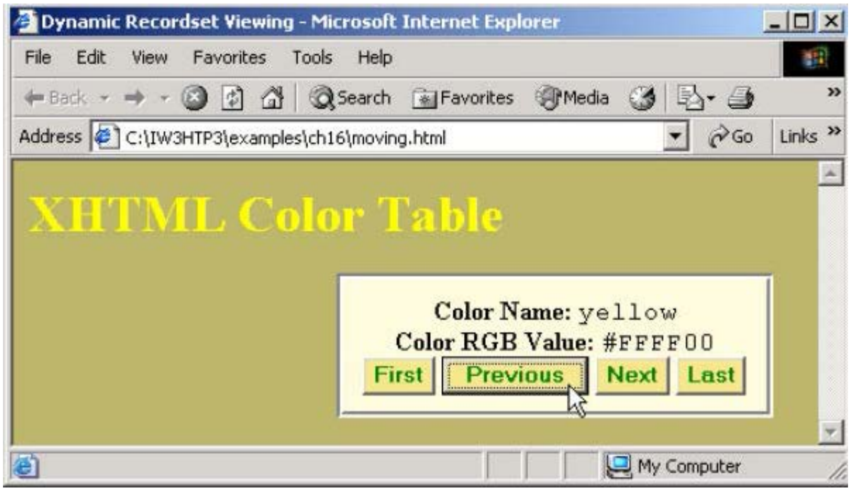
XHTML Color Table

Color Name: aqua
Color RGB Value: #00FFFF


Dynamic Recordset Viewing - Microsoft Internet Explorer
C:\IW3HTP3\examples\ch16\moving.html
First
Previous
Next
Last
My Computer

XHTML Color Table

Color Name: `white`
Color RGB Value: `#FFFFFF`


Dynamic Recordset Viewing - Microsoft Internet Explorer
C:\IW3HTP3\examples\ch16\moving.html
First
Previous
Next
Last
My Computer

XHTML Color Table

Color Name: `yellow`
Color RGB Value: `#FFFF00`

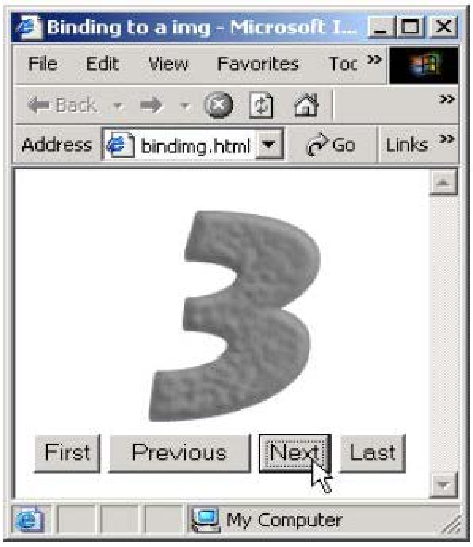
6.4 BINDING TO AN IMAGE

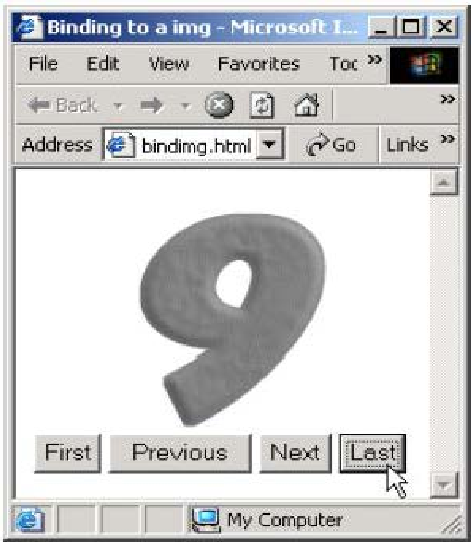
Different types of XHTML elements can be bound to data sources using img element. Changing the recordset updates the src attribute of the image.

Example

```
<?xml version = "1.0"?>

  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

  <!-- Binding data to an image -->

  <html xmlns = "http://www.w3.org/1999/xhtml">

   <head>

  <title>Binding to a img</title>

  <object id = "Images" classid =
"CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value = "images.txt" />
<param name = "UseHeader" value = "True" />
</object>
<script type = "text/javascript">
 <!--
recordSet = Images.recordset;
function move( whereTo)
{
switch( whereTo) {
case "first":
recordSet.MoveFirst();
break;
case previous":
recordSet.MovePrevious();
if ( recordSet.BOF )
recordSet.MoveLast();
break;
case "next":
recordSet.MoveNext();
if ( recordSet.EOF )
recordSet.MoveFirst();
break;
case "last":
recordSet.MoveLast();
break;
}
}
// -->
</script>
</head>
<body>
<img datasrc = "#Images" datafld = "image" alt = "Image"
style = "position: relative; left: 45px" /><br />
<input type = "button" value = "First"  onclick = "move('first');" />
<input type = "button" value = "Previous"  onclick = "move('previous');" />
<input type = "button" value = "Next"  onclick = "move('next');" />
<input type = "button" value = "Last" onclick = "move( 'last' );" />
</body>
</html>
```

Output


Binding to a img - Microsoft I...
File Edit View Favorites Toc
Back
Address bindimg.html Go Links
First Previous Next Last
My Computer


Binding to a img - Microsoft I...
File Edit View Favorites Toc
Back
Address bindimg.html Go Links
First Previous Next Last
My Computer

CHECK YOUR PROGRESS

- What do you understand by data binding?
- Give the main work of recordSet.
- Also write the name of important predefined functions which are associated with recordSet.

6.5 BINDING TO AN IMAGE

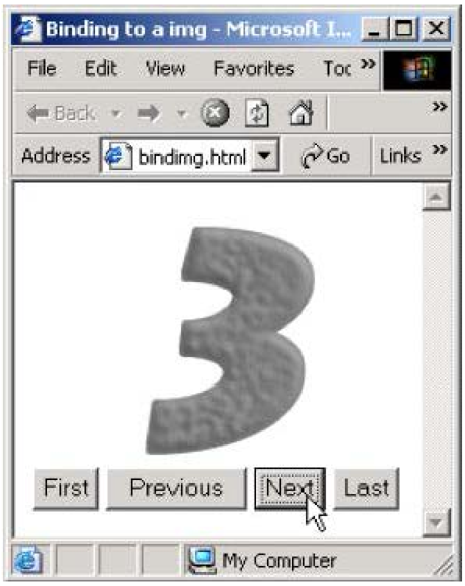
Many different types of XHTML elements can be bound to data sources. It can be done by binding to an img element. For this you need to change the src attribute of the image.

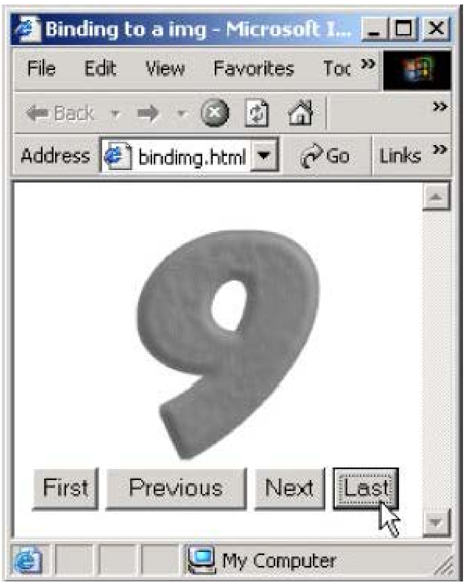
Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Binding data to an image -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Binding to a img</title>
<object id = "Images"
classid = "CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value = "images.txt" />
<param name = "UseHeader" value = "True" />
</object>
<script type = "text/javascript">
 <!--
 recordSet = Images.recordset;
function move( whereTo )
{
 switch( whereTo ) {
case "first":
recordSet.MoveFirst();
break;
case previous":
recordSet.MovePrevious();
if ( recordSet.BOF )
recordSet.MoveLast();
```

```
 break;
case "next":
 recordSet.MoveNext();
if(recordSet.EOF)
recordSet.MoveFirst();
break;
case "last":
recordSet.MoveLast();
break;
}
}
// -->
</script>
</head>
<body>
<img datasrc = "#Images" datafld = "image" alt = "Image"
style = "position: relative; left: 45px" /><br />
<input type = "button" value = "First"
 onclick = "move( 'first' );" />
<input type = "button" value = "Previous"
 onclick = "move( 'previous' );" />
<input type = "button" value = "Next"
onclick = "move( 'next' );" />
<input type = "button" value = "Last"
onclick = "move( 'last' );" />
</body>
</html>
```

Output


Binding to a img - Microsoft I...
File Edit View Favorites Toc
Back
Address bindimg.html Go Links
First Previous Next Last
My Computer


Binding to a img - Microsoft I...
File Edit View Favorites Toc
Back
Address bindimg.html Go Links
First Previous Next Last
My Computer

6.6 BINDING TO A TABLE

Binding of the data to a table is different from the normal binding.

Example

```
<?xml version = "1.0"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Using Data Binding with tables -->

<html xmlns = "http://www.w3.org/1999/xhtml">

<head>

<title>Data Binding and Tables</title>

<object id = "Colors"

classid = "CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">

<param name = "DataURL" value = "HTMLStandardColors.txt" />

<param name = "UseHeader" value = "TRUE" />

<param name = "TextQualifier" value = "@" />

<param name = "FieldDelim" value = "|" />

</object>
```

```
</head>
<body style = "background-color: darkseagreen">
<h1>Binding Data to a <code>table</code></h1>
<table datasrc = "#Colors" style = "border-style: ridge;
border-color: darkseagreen;
background-color: lightcyan">
<thead>
<tr style = background-color: mediumslateblue">
<th>Color Name</th>
<th>Color RGB Value</th>
</tr>
</thead>
<tbody>
<tr style = "background-color: lightsteelblue">
<td><span datafld = "ColorName"></span></td>
<td><span datafld = "ColorHexRGBValue"
 style = "font-family: monospace"></span></td>
</tr>
</tbody>
</table>
</body>
</html>
```

6.7 SORTING TABLE DATA

To manipulate a large data source first you need to sort the data. It can be done using sort property.

Example

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Sorting table data -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Data Binding and Tables</title>
<object id = "Colors"
classid = "CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value = "HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
<param name = "FieldDelim" value = "|" />
</object>
</head>
<body style = "background-color: darkseagreen">
<h1>Sorting Data</h1>
<table datasrc = "#Colors" style = "border-style: ridge;
border-color: darkseagreen;
background-color: lightcyan">
<caption>
Sort by:
<select onchange = "Colors.Sort = this.value;
Colors.Reset();">
<option value = "ColorName">Color Name (Ascending) </option>
<option value = "-ColorName">Color Name (Descending) </option>
<option value = "ColorHexRGBValue">Color RGB Value (Ascending)</option>
<option value = "-ColorHexRGBValue">Color RGB Value (Descending)</option>
</select>
</caption>
<thead>
```

```
<tr style = "background-color: mediumslateblue">
<th>Color Name</th>
<th>Color RGB Value</th>
</tr>
</thead>
<tbody>
<tr style = "background-color: lightsteelblue">
<td><span datafld = "ColorName"></span></td>
<td><span datafld = "ColorHexRGBValue"
style = font-family: monospace"></span></td>
</tr>
</tbody>
</table>
</body>
</html>
```

CHECK YOUR PROGRESS

- How you can do the binding in table?
- Write the name of the command to sort the data.

6.8 SUMMARY

- Transitions and filters are programmable and can be programmed to made changes for user requests. Filters may of many types like flip Filters, mask Filters, image Filters, chroma filters etc.
- Flip filters are basically of two types i.e. flip horizontal (fliph) and flip vertical (flipv). These flipv and fliph filters mirror text or images vertically and horizontally, respectively.
- Transparency effects can be added using chroma filters dynamically. For doing these no graphics editor is required for making changes into an image.
- Like other functions in DHTML you can also process images. You can change the background, foreground, etc. Normally background color is a solid color and foreground is transparent to the image or color behind it.

6.9 TERMINAL QUESTIONS

1. Define the term table.
2. Write the name of table components.
3. Give the difference between data and information.
4. Write the importance of binding.
5. How you can do the binding of the data in a table using recordSet.
6. Write a program in DHTML to sort the data of a table containing at least 10 records.
7. What do you mean by binding an image?
8. Write a program in DHTML to bind an image.

U. P. RAJARSHI TANDON OPEN UNIVERSITY

BCA-E9

Bachelor in Computer Applications

Block

3

Introduction to Database Concepts

BLOCK INTRODUCTION

Block-3 basically contains two units in all which mainly explain the concepts of database and XML. This block incorporates Unit-7 and Unit-8. As we know that database is the key component for dynamic feature of any Internet application. The term database has been derived from data and the data is the plural of datum which means facts. The inter-related collection of data is known as database.

Unit-7 describes the role and functionality of database and data models. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A database is a persistent, logically coherent collection of inherently meaningful data, relevant to some aspects of the real world or in other words "A database is a structured collection of records". A database management system stores (DBMS) data in such a way that it becomes easier to retrieve, manipulate, and produce information. DBMS is an aggregation of data, hardware, software, and users that help an enterprise manage its operational data. The main function of a DBMS is to provide efficient and reliable methods of data retrieval to many users.

Unit-8 explains about web resources and XML. XML or extensible markup language is used for the development of a bridge between front-end and back-end. Basically it is used for the data exchange. In general, Web pages and documents on the Internet that provides useful information are the web resources. While an online resource is typically data and educational in nature, any support software available online can be considered a resource.

Moreover XML is the key component of DHTML. And DHTML is the combination of: CSS, JavaScript, and XML. Furthermore XML is the extension of HTML with advance features. XML is basically used for exchanging the data over Internet. More or less, directly or indirectly, XML is being used in all the Internet technologies. You can say that a web based application cannot be built without XML. Its parsers DOM and SAX are the key components on which it is based. Other technologies that work to develop the websites are ASP, JSP, PHP etc.

Unit - 7 : Introduction to Database

Structure

7.0 INTRODUCTION

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A database is a persistent, logically coherent collection of inherently meaningful data, relevant to some aspects of the real world or in other words "A database is a structured collection of records".

A database can be summarily described as a repository for data. This makes clear that building databases is really a continuation of a human activity that has existed since writing began. The creation of a database is required by the operation of an enterprise. We use the term enterprise to designate a variety of endeavors that range from an airline, a bank, or a manufacturing company to a stamp collection or keeping track of people to whom you want to write New Year cards.

We can use a running example that deals with the database as a small college. The college keeps track of its students, its instructors, the courses taught by the college, grades received by students, and the assignment of advisors to students, as well as other aspects of the activity of the institution.

These data items constitute the operational data — that is, the data that the college needs to function. Operational data are built from various input data (application forms for students, registration forms, grade lists, schedules) and is used for generating output data (transcripts, registration records, administrative reports, etc.) Note that no computer is necessary for using such a database; a college of the 1930's would have kept the same database in paper form.

7.1 OBJECTIVES

- To know about the data and database
- Overview of database management system (DBMS)
- To know about the different data models
- Functions and advantages of the DBMS
- Relational database management system
- SQL and its languages like DDL, DML, & DCL
- Important command and their uses
- Examples based on SQL

7.2 DATABASE MANAGEMENT SYSTEMS

A database management system stores (DBMS) data in such a way that it becomes easier to retrieve, manipulate, and produce information. DBMS is an aggregation of data, hardware, software, and users that help an enterprise manage its operational data. The main function of a DBMS is to provide efficient and reliable methods of data retrieval to many users. Let's take an example of college:

If our college has 10,000 students each year and each student can have approximately 10 grade records per year, then over 10 years, the college will accumulate 1,000,000 grade records. It is not easy to extract records satisfying certain criteria from such a set, and by current standards, this set of records is quite small! Given the current concern for "grade inflation", a typical question that we may try to answer is determining the evolution of the grade averages in introductory programming courses over a 10-year period. Therefore, it is clear that efficient data retrieval is an essential function of database systems.

The basic role and activity of a DBMS is shown in figure 7.1. This is a normal process or activity that user makes a request from client side and then It is handles through a front end software. Theses request are available in the form of some queries developed using a language SQL. Hence the queries are known as SQL queries. The data is retrieved from the actual database and after retrieval it is made available for the end user in the form of reports or views.

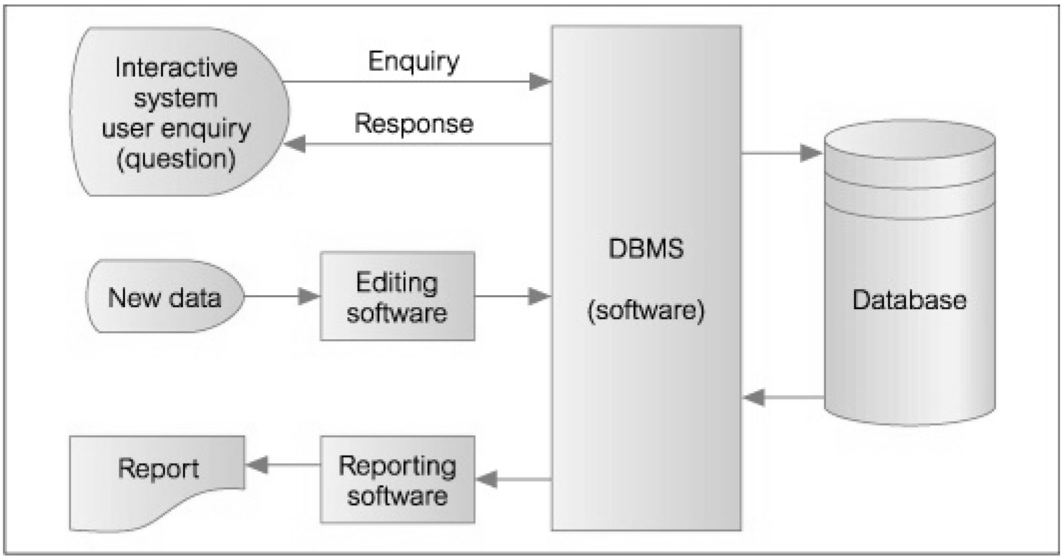


Figure 7.1 : DBMS Activity

Most DBMSs deal with several users who try simultaneously to access several data items and, frequently, the same data item. For instance, suppose that we wish to introduce an automatic registration system for students. Students may register by using terminals or workstations. Of course, we assume that the database contains information that describes the capacity of the courses and the number of seats currently available.

7.2.1 Characteristic

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics:

Real-world entity: A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

Relation-based tables: DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

Isolation of data and application: A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process. **Less redundancy:** DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

Consistency: Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

Query Language: DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

ACID Properties: DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

Multiuser and Concurrent Access: DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

Multiple views: DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

Security: Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department.

CHECK YOUR PROGRESS

- Define the term database.
- What is DBMS? Give the name of important characteristics.

7.3 DATA MODELS

As in other technologies, data models in DBMS are relatively simple representations, usually graphical, of complex real-world data structures. It facilitates interaction among the designer, the applications programmer, and the end user. Normally end-users have different views and needs for data. Data model organizes data for various users. The main building blocks of theses are: entity - anything about which data are to be collected and stored, attribute - a characteristic of an entity, relationship - describes an association among entities. Relationship may be like one-to-many (1:M) relationship, many-to-many (M:N or M:M) relationship, one-to-one (1:1) relationship.

7.3.1 The Hierarchical Model

This model was developed in the 1960s to manage large amounts of data for complex manufacturing projects. Basic logical structure is represented by an upside-down "tree" (figure). The hierarchical structure contains levels, or segments. It depicts a set of one-to-many (1:M) relationships between a parent and its children segments. Each parent can have many children and each child has only one parent. Many of the hierarchical data model's features formed the foundation for current data models

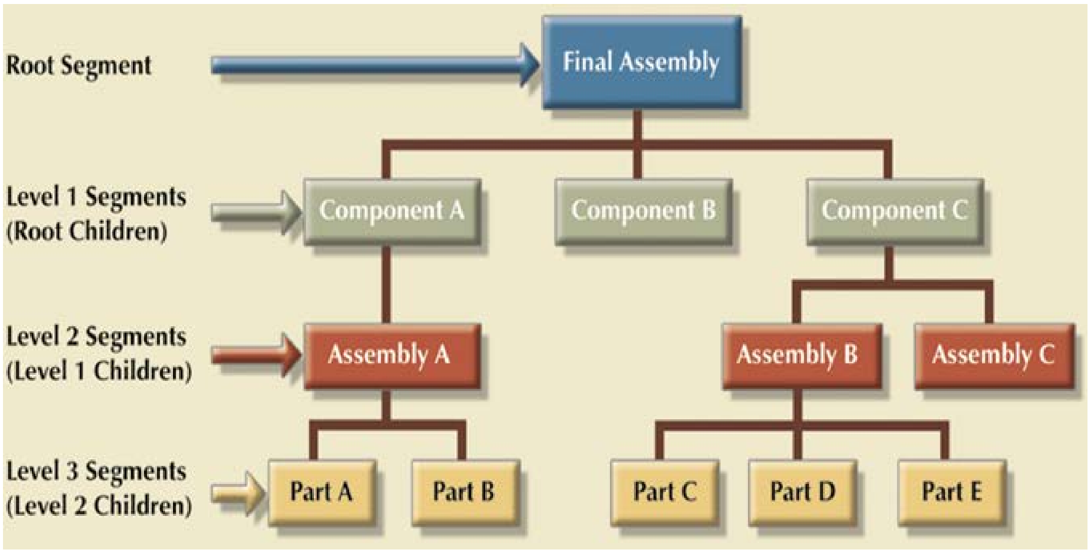


Figure 7.2: Hierarchical Model

7.3.2 The Network Model

It is created to represent complex data relationships more effectively. It improves database performance and reliability. Moreover, It impose a database standard. In terms of schema, It is a conceptual organization of entire database as viewed by the database administrator and on the other hand If you talk about subschema, It defines database portion "seen" by the application programs that actually produce the desired information from data contained within the database. It basically uses Data Management Language (DML) and hence defines the environment in which data can be managed (figure-7.3).

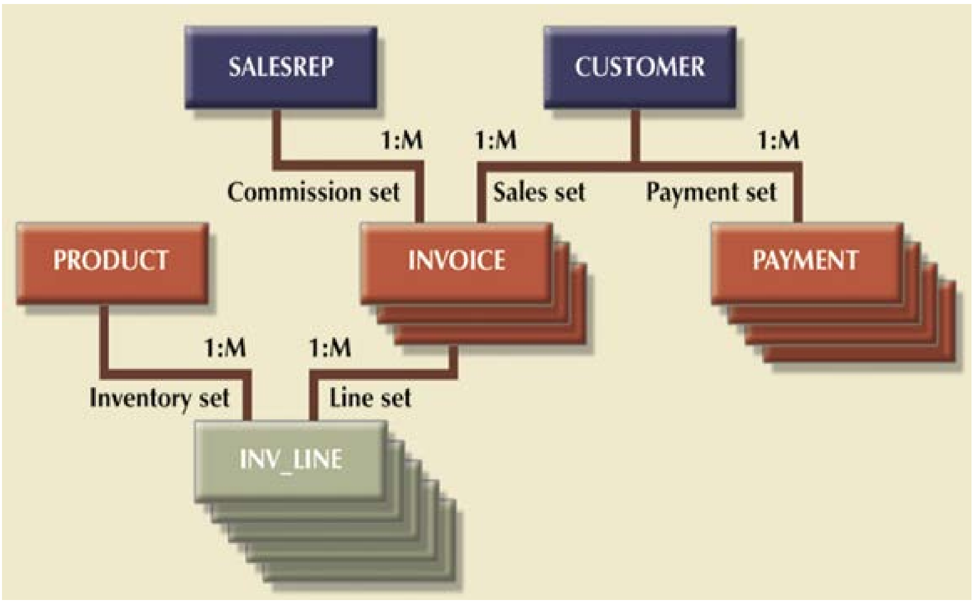


Figure 7.3 : Network Model

7.3.3 The Relational Model

It was developed by Codd (IBM) in 1970. It is considered ingenious but impractical in 1970. The concepts of relational model are very simple and based on table because of which It is called relation. Earlier computers lacked power to implement the relational model but today, microcomputers can run sophisticated relational database software. Moreover, Relational Database Management System (RDBMS) performs same basic functions provided by hierarchical and network DBMS systems, in addition to a host of other functions. Most important advantage of the RDBMS is its ability to hide the complexities of the relational model from the user. Two simple relations are shown in figure-

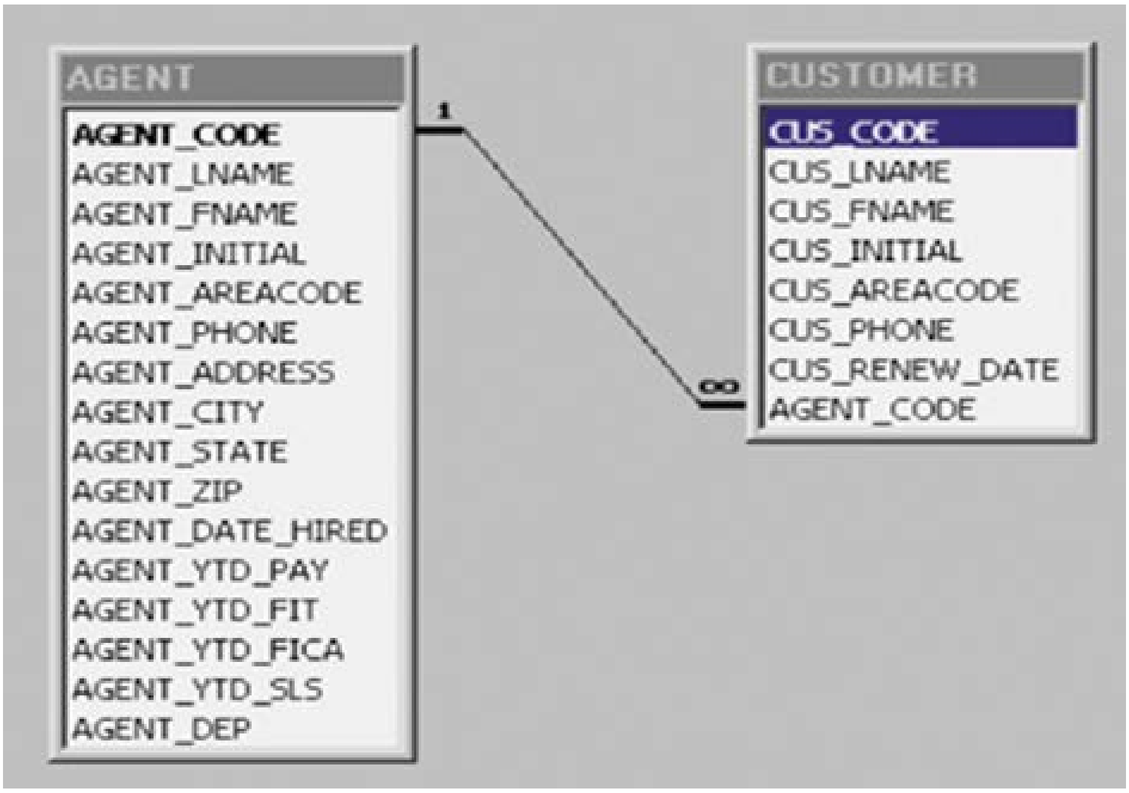


Figure 7.4: Relational Model

7.3.4 The Entity Relationship Model

This is widely accepted and adapted graphical tool for data modeling in DBMS. It was introduced by Chen in 1976. Basically It is a graphical representation of entities and their relationships in a database structure. Moreover, entity relationship diagram (ERD) uses graphic representations to model database components where entity is mapped to a relational table. Furthermore, entity instance (or occurrence) is row in table and entity set is collection of like entities. Connectivity labels types of relationships where diamond connected to related entities through a relationship line.

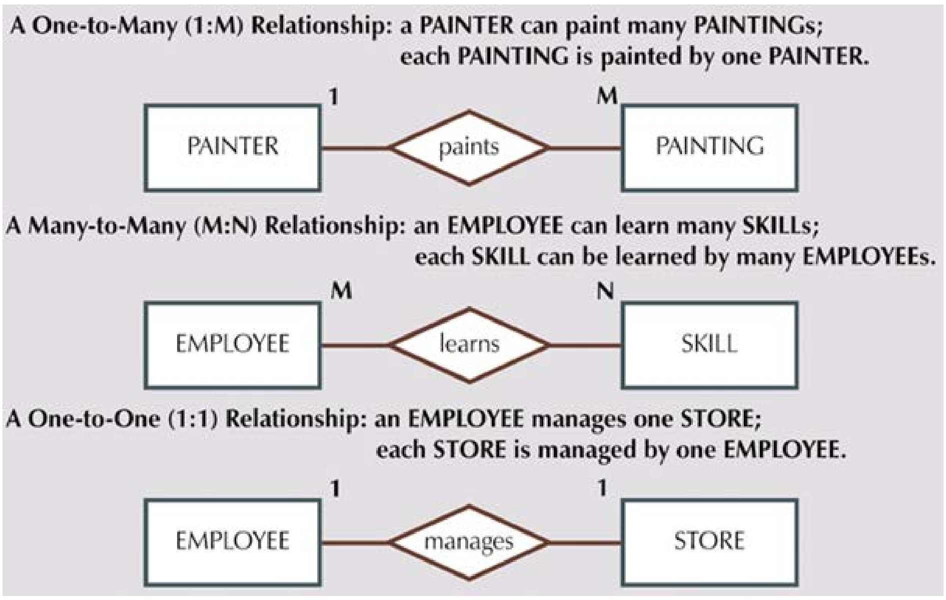
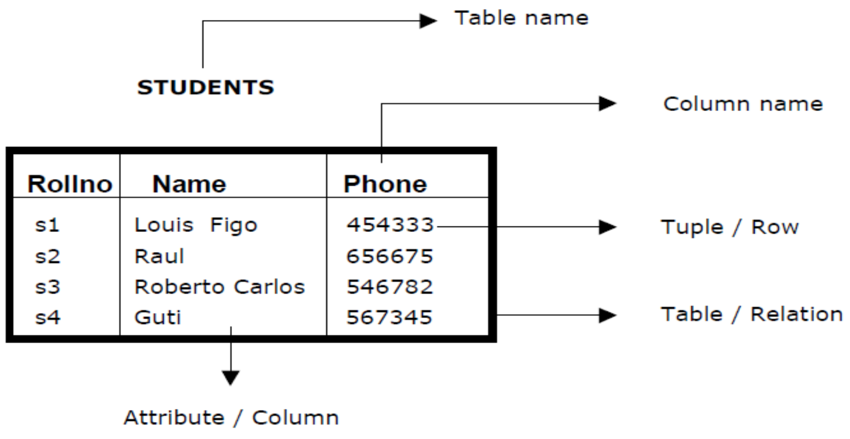

A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs;
each PAINTING is painted by one PAINTER.
1
M
PAINTER
paints
PAINTING
A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs;
each SKILL can be learned by many EMPLOYEEs.
M
N
EMPLOYEE
learns
SKILL
A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE;
each STORE is managed by one EMPLOYEE.
1
1
EMPLOYEE
manages
STORE

Figure 7.5: E-R (Entity-Relationship) Model

7.4 RELATIONAL DATABASE MANAGEMENT SYSTEMS

RDBMS stands for Relational Database Management System. RDBMS data is structured in database tables, fields and records. Each RDBMS table consists of database table rows. Each database table row consists of one or more database table fields. RDBMS store the data into collection of tables, which might be related by common fields (database table columns).

RDBMS also provide relational operators to manipulate the data stored into the database tables. Most RDBMS use SQL as database query language. The most popular RDBMS are MS SQL Server, DB2, Oracle and MySQL. The relational model is an example of record -based model. Record based models are so named because the database is structured in fixed format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record types. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.



Rollno	Name	Phone
s1	Louis Figo	454333
s2	Raul	656675
s3	Roberto Carlos	546782
s4	Guti	567345

Figure 7.6: A table in relational model

Tuple / Row: A single row in the table is called as tuple. Each row represents the data of a single entity.

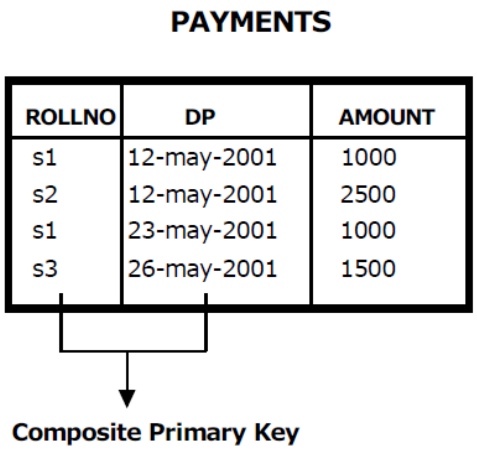
Attribute / Column: A column stores an attribute of the entity. For example, if details of students are stored then student name is an attribute; course is another attribute and so on.

Column Name: Each column in the table is given a name. This name is used to refer to value in the column.

Table Name: Each table is given a name. This is used to refer to the table. The name depicts the content of the table. Primary key and foreign key are very important in relational model.

Primary Key: A table contains the data related entities. If you take STUDETNS table, it contains data related to students. For each student there will be one row in the table. Each student's data in the table must be uniquely identified. In order to identify each entity uniquely in the table, we use a column in the table. That column, which is used to uniquely identify entities (students) in the table is called as primary key. In case of STUDENTS table (see figure 1) we can use ROLLNO as the primary key as it in not duplicated. So a primary key can be defined as a set of columns used to uniquely identify rows of a table. Some other examples for primary keys are account number in bank, product code of products, and employee number of an employee.

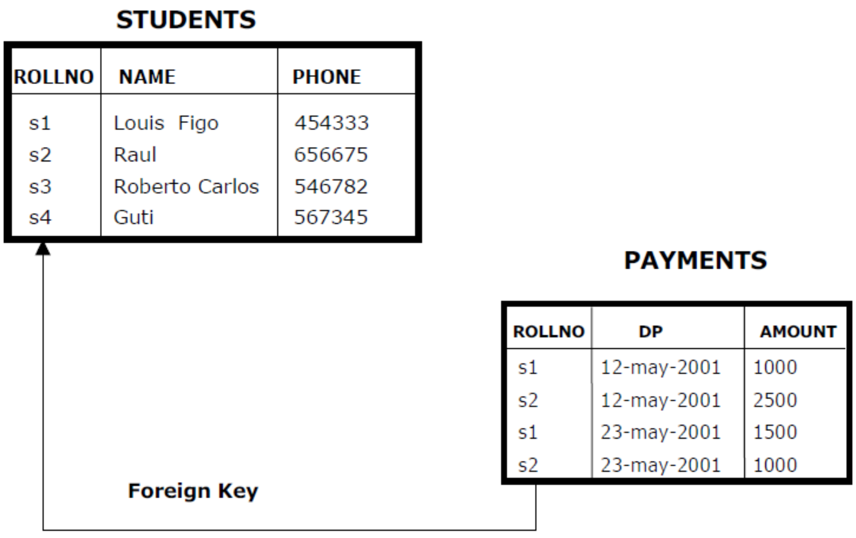
Composite Primary Key: In some tables a single column cannot be used to uniquely identify entities (rows). In that case we have to use two or more columns to uniquely identify rows of the table. When a primary key contains two or more columns it is called as composite primary key. In figure 2, we have PAYMENTS table, which contains the details of payments made by the students. Each row in the table contains roll number of the student, payment date and amount paid. Neither of the columns can uniquely identify rows. So we have to combine ROLLNO and DP to uniquely identify rows in the table. As primary key is consisting of two columns it is called as composite primary key.



ROLLNO	DP	AMOUNT
s1	12-may-2001	1000
s2	12-may-2001	2500
s1	23-may-2001	1000
s3	26-may-2001	1500

Figure 7.7: Composite Primary Key

Foreign Key: In relational model, we often store data in different tables and put them together to get complete information. For example, in PAYMENTS table we have only ROLLNO of the student. To get remaining information about the student we have to use STUDETNS table. Roll number in PAYMENTS table can be used to obtain remaining information about the student. The relationship between entities student and payment is one-to-many. One student may make payment for many times. As we already have ROLLNO column in PAYMENTS table, it is possible to join with STUDENTS table and get information about parent entity (student). Roll number column of PAYMENTS table is called as foreign key as it is used to join PAYMENTS table with STUDENTS table. So foreign key is the key on the many side of the relationship.



STUDENTS

ROLLNO	NAME	PHONE
s1	Louis Figo	454333
s2	Raul	656675
s3	Roberto Carlos	546782
s4	Guti	567345

PAYMENTS

ROLLNO	DP	AMOUNT
s1	12-may-2001	1000
s2	12-may-2001	2500
s1	23-may-2001	1500
s2	23-may-2001	1000

Figure.7.8: Foreign Key

CHECK YOUR PROGRESS

- What is the actual significance of relation in RDBMS.
- Compare attribute and tuple in a relation.

7.5 DATABASE SYSTEM HARDWARE

Database management systems are, in most cases, installed on general-purpose computers. Since the characteristics of the hardware have strongly influenced the development of DBMSs, we discuss some of the most important of these characteristics. Computer memory can be categorized into two classes: Internal memory and External memory.

Although some internal memory is permanent, such as ROM we are interested here only in memory that can be changed by programs. This memory is often known as RAM. This memory is volatile, and any electrical interruption causes the loss of data. By contrast, magnetic disks and tapes are common forms of external memory. They are nonvolatile memory, and they retain their content for practically unlimited amounts of time. The physical characteristics of magnetic tapes force them to be accessed sequentially, making them useful for backup purposes, but not for quick access to specific data.

In examining the memory needs of a DBMS, we need to consider the following issues:

- Data of a DBMS must have a persistent character; in other words, data must remain available long after any program that is using it has completed its work. Also, data must remain intact even if the system breaks down.
- A DBMS must access data at a relatively high rate.

- Such a large quantity of data needs to be stored that the storage medium must be low cost. These requirements are satisfied at the present stage of technological development only by magnetic disks.

7.6 DATABASE SYSTEM SOFTWARE

Users interact with database systems through query languages. The query language of a DBMS has two broad tasks:

- To define the data structures that serve as receptacles for the data of the database.
- To allow the speedy retrieval and modification of data.

Accordingly, we distinguish between two components of a query language: the data definition component and the data manipulation component. The main tasks of data manipulation are data retrieval and data update. Data retrieval entails obtaining data stored in the database that satisfies a certain specification formulated by the user in a query. Data updates include data modification, deletion and insertion. Programming in query languages of DBMSs is done differently from programming in higher-level programming languages. The typical program written in C, Pascal, or PL/1 directly implements an algorithm for solving a problem.

A query written in a database query language merely states what the problem is and leaves the construction of the code that solves the problem to a special component of the DBMS software. This approach to programming is called non procedural. A central task of DBMSs is transaction management. A transaction is a sequence of database operations (that usually consists of updates, with possible retrievals) that must be executed in its entirety or not at all. This property of transactions is known as atomicity.

A typical example includes the transfer of funds between two account records A and B in the database of a bank. Such a banking operation should not modify the total amount of fund s that the bank has in its accounts, which is a clear consistency requirement for the database. The transaction consists of the following sequence of operations:

- Decrease the balance of account A by d dollars;
- Increase the balance of account B by d dollars.

7.7 DIFFERENCE BETWEEN DBMS AND SYSTEM SOFTWARE

A DBMS has to be persistent, that is it should be accessible when the program created the data ceases to exist or even the application that created the data restarted. A DBMS also has to provide some uniform methods independent of a specific application for accessing the information that is stored. RDBMS is a Relational Data Base Management System Relational DBMS. This adds the additional condition that the system supports a tabular structure for the data, with enforced relationships between the tables. This excludes the databases that don't support a tabular structure or don't enforce relationships between tables. You can say DBMS does not impose any constraints or security with regard to data manipulation it is user or the programmer responsibility to ensure the ACID PROPERTY of the database whereas the RDBMS is more with this regard because RDBMS define the integrity constraint for the purpose of holding ACID PROPERTY.

CHECK YOUR PROGRESS

- What do you understand by system hardware?
- Give the difference between system hardware and software.

7.8 INTRODUCTION TO SQL

SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS. SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database.

Structured Query Language (SQL) uses the combination of Relational algebra and Relational calculus. It is a data sub language used to organize, manage and retrieve data from relational database, which is managed by Relational Database Management System (RDBMS). Vendors of DBMS like Oracle, IBM, DB2, Sybase, and Ingress use SQL as programming language for their database. SQL originated with the system R project in 1974 at IBM's San Jose Research Centre. Original version of SQL was SEQUEL which was an Application Program Interface (API) to the system R project. The predecessor of SEQUEL was named SQUARE. SQL-92 is the current standard and is the current version. The SQL language can be used in two ways: Interactively or Embedded inside another program.

The SQL is used interactively to directly operate a database and produce the desired results. The user enters SQL command that is immediately executed. Most databases have a tool that allows interactive execution of the SQL language. These include SQL Base's SQL Talk, Oracle's SQL Plus, and Microsoft's SQL server 7 Query Analyzer. The second way to execute a SQL command is by embedding it in another language such as Cobol, Pascal, BASIC, C, Visual Basic, Java, etc. The result of embedded SQL command is passed to the variables in the host program, which in turn will deal with them. The combination of SQL with a fourth-generation language brings together the best of two worlds and allows creation of user interfaces and database access in one application.

7.6.1 Subdivisions of SQL

Regardless of whether SQL is embedded or used interactively, it can be divided into three groups of commands, depending on their purpose:

- Data Definition Language (DDL).
- Data Manipulation Language (DML).
- Data Control Language (DCL).

Data Definition Language

Data Definition Language is a part of SQL that is responsible for the creation, updation and deletion of tables. It is responsible for creation of views and indexes also. SQL uses the following set of commands to define database schema:

1 - CREATE : Creates new databases, tables, and views from RDBMS.

For example:

```
Create database tutorialspoint;

Create table article;
Create view for_students;
```

2 – DROP : Drops commands, views, tables, and databases from RDBMS.

For example:

```
Drop object_type object_name;

Drop database tutorialspoint;

Drop table article;

Drop view for_students;
```

3 - ALTER: Modifies database schema.

```
Alter object_type object_name parameters;
```

For example:

```
Alter table article add subject varchar;
```

Data Manipulation Language

SQL is equipped with data manipulation language (DML). DML modifies the database instance by inserting, updating, and deleting its data. DML is responsible for all forms data modification in a database. SQL contains the following set of commands in its DML section:

- SELECT/FROM/WHERE
- INSERT INTO/VALUES
- UPDATE/SET/WHERE
- DELETE FROM/WHERE

These basic constructs allow database programmers and users to enter data and information into the database and retrieve efficiently using a number of filter options.

1 - SELECT/FROM/WHERE

SELECT: This is one of the fundamental query command of SQL. It is similar to the projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.

FROM: This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.

WHERE: This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.

For example :

```
Select author_name
From book_author
Where age > 50;
```

This command will yield the names of authors from the relation **book_author** whose age is greater than 50.

2-INSERT INTO/VALUES

This command is used for inserting values into the rows of a table (relation).

Syntax:

```
INSERT INTO table (column1 [, column2, column3 ... ]) VALUES (value1 [,
value2, value3 ... ])
```

Or

```
INSERT INTO table VALUES (value1, [value2, ... ])
```

For example:

```
INSERT INTO tutorialspoint (Author, Subject) VALUES ("anonymous",
"computers");
```

3 - UPDATE/SET/WHERE

This command is used for updating or modifying the values of columns in a table (relation).

Syntax:

```
UPDATE table_name SET column_name = value [, column_name = value ...]
[WHERE condition]
```

For example:

```
UPDATE tutorialspoint SET Author="webmaster" WHERE Author="anonymous";
```

4 - DELETE/FROM/WHERE

This command is used for removing one or more rows from a table (relation).

Syntax:

```
DELETE FROM table_name [WHERE condition];
```

Example :

```
DELETE FROM tutorialspoint
WHERE Author="unknown";
```

Data Control Language

This is the third and least respected sub-language in SQL. The commands that form data control language are related to the security of the database performing tasks of assigning privileges so users can access certain objects in the database. The DCL commands are:

1 – GRANT : SQL GRANT is a command used to provide access or privileges on the database objects to the users.

Syntax:

```
GRANT privilege_name
ON object_name
TO{User_name |public |role_name}
[with GRANT option];
```

Example

GRANT SELECT ON employee TO user1;

This command grants a SELECT permission on employee table to user1. You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

2 - REVOKE : The REVOKE command removes user access rights or privileges to the database objects.

Syntax

```
GRANT privilege_name
ON object_name
FROM {User_name |public |role_name}
```

For Example : REVOKE SELECT ON employee FROM user1;

This command will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

3 - COMMIT : The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

Syntax:

```
COMMIT;
```

4 – ROLLBACK : The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

Syntax :

```
ROLLBACK ;
```

CHECK YOUR PROGRESS

- Define the term SQL.
- Write the name their important commands associated with SQL.

7.9 INTRODUCTION TO ASP

ASP stands for Active Server Pages. Microsoft introduced Active Server Pages in December 1996, beginning with Version 3.0. Microsoft officially defines ASP as: "Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions. ASP enables server side scripting for IIS with native support for both VBScript and Jscript. In other words, ASP is a Microsoft technology that enables you to create dynamic web sites with the help of server side script, such as VBScript and Jscript. ASP technology is supported on all Microsoft Web servers that are freely available. I

7.9.1 ASP file

An ASP file is quite like an HTML file. It contains text, HTML tags and scripts, which are executed on the server. The two widely used scripting languages for an ASP page are VBScript and JavaScript. VBScript is pretty much like Visual Basic, whereas Jscript is the Microsoft's version of JavaScript. However, VBScript is the default scripting language for ASP. Besides these two scripting languages, you can use other scripting language with ASP as long as you have an ActiveX scripting engine for the language installed, such as Perl Script.

7.9.2 ASP Process

As you have learned, scripts in an ASP file are server-side scripts, which means that the scripts are processed on the server and then the result of the scripts will be converted to HTML before sending to the web browser.

To illustrate, let's take a look at these steps

- A user requests a web page in the web browser.
- The browser finds the appropriate web server (like IIS or PWS), and asks for the required age.

- The web server locates the required page, and parses out the ASP code within the ASP script delimiters (<%…%>), produces a standard HTML page. The server sends that HTML page back to the browser, so the user cannot see ASP code.
- The browser executes the client side scripting (like JavaScripts) determining how to display the results.

7.9.3 Active Server Page Objects

Active Server Pages provide several built-in objects that offer programmers straightforward methods for communicating with a Web browser, gathering data sent by an HTTP request and distinguishing between users. Following table provides a brief description of the most commonly used ASP objects.

Table.7.1: Objects and Description

Object name	**Description**
Request	Used to access information passed by an HTTP request.
Response	Used to control the information sent to the client.
Server	Used to access methods and properties on the server.

The Request object is commonly used to access the information passed by a get or post request. This information usually consists of data provided by the user in an XHTML form. The Request object provides access to information (such as cookies) that is stored on a client's machine. This object also provides access to binary information (e.g., a file upload). The Response object sends information, such as XHTML or text to the client. The Server object provides access to methods and properties on the server.

7.9.4 File System Objects

File System Objects (FSOs) enable programmers to manipulate files, directories and drives. FSOs also allow programmers to read and write text and are an essential element for Active Server Pages that persist data. We overview FSO features and then provide a live code example that uses FSOs. FSOs are objects in the Microsoft Scripting Runtime Library. These FSO types include File System Object, File, Folder, Drive and Text Stream. Each type is summarized in following table.

Table.7.2: File System Objects and their Description

Object type	**Description**
File System Object	Interacts with Files, Folders and Drives.
File	Manipulates Files of any type.
Folder	Manipulates Folders (i.e., directories).
Drive	Gathers information about Drives (hard disks, RAM disks—computer memory used as a substitute for hard disks to allow high-speed file operations, CD-ROMs, etc.). Drives can be local or remote.
Text Stream	Reads and writes text files.

The programmer can use File System Objects to create directories, move files, and determine whether a Drive exists. The File object allows the programmer to gather information about files, manipulate files and open files. Property Path contains the File's path in long name format (the operating system does not abbreviate the name when it exceeds the 8.3 format—the format in which the filename contains at most eight characters and the extension contains at most three characters). Property Short Name contains, if applicable, the file name in short name format (a file name exceeding the 8.3 format is abbreviated). For example, "ABCDEFGHIJ.doc" is a file name in long name format. The same file name in short name format might be abbreviated as "ABCDEF~1.doc".

Table.7.3: Methods and their Description

Method	**Description**
CopyFile	Copies an existing File
CopyFolder	Copies an existing Folder
CreateFolder	Creates and returns a Folder
CreateTextFile	Creates and returns a text File
DeleteFile	Deletes a File
DeleteFolder	Deletes a Folder
DriveExists	Tests whether a Drive exists. Returns a Boolean
FileExists	Tests whether a File exists. Returns a Boolean
FolderExists	Tests whether a Folder exists. Returns a Boolean
GetAbsolutePathName	Returns the absolute path as a string
GetDrive	Returns the specified Drive
GetDriveName	Returns the Drive drive name
GetFile	Returns the specified File
GetFileName	Returns the File file name
GetFolder	Returns the specified Folder
GetParentFolderName	Returns a string representing the parent folder name
GetTempName	Creates and returns a string representing a file name
MoveFile	Moves a File
MoveFolder	Moves a Folder

7.10 SESSION TRACKING AND COOKIES

HTTP does not support persistent information that could help a Web server distinguish between clients. In this section, we introduce two related technologies that enable a Web server to distinguish between clients: *session tracking* and *cookies*. Many Web sites provide custom Web pages or functionality on a client-by-client basis. For example, some Web sites allow you to customize their home page to suit your needs. An example of this is the *Yahoo!* Web site (my.yahoo.com), which allows you to customize how the Yahoo! site appears. [*Note*: You need to get a free Yahoo! ID first.]

Another example of a service that is customized on a client-by-client basis is a shopping cart for shopping on the Web. When a purchase is made, the server must distinguish between clients so that the business can assign the proper items and charge each client the proper amount.

7.10.1 Using Sessions

The server performs session tracking by keeping track of when a specific person visits a site. The first time a client connects to the server, the server assigns the client a unique *session ID*, a unique value used to identify each individual user. When the client makes additional requests, the client's session ID is compared with the session IDs stored in the server's memory. Active Server Pages use the *Session* object to manage sessions. The Session object's *Timeout* property specifies the number of minutes that a session exists before it expires. The default value for property Timeout is 20 minutes. Calling Session method *Abandon* can also terminate an individual session.

Users who are not familiar with ASP may input their information in a form and submit the form, and the ASP page generator will create the user's ASP page. This consists of two Active Server Pages linked to each other through HTTP post requests. We use session variables in this to maintain a state between the two ASP pages. Multiple Active Server Pages connected in this manner are sometimes called an *ASP application*.

The first page consists of a form that requests information from the user. When submitted, the form is posted to process page *process.asp*. If there are no errors, process.asp creates the user's ASP page. Otherwise, process.asp redirects the user back to first page. Also, process.asp stores a "welcome back" message in *session variable* "welcome Back". Each time a user submits the form, process.asp stores a new "welcome back" message in the session variable.

CHECK YOUR PROGRESS

- o Which type of applications we can develop using ASP.
- o Write the name of file system objects.

7.11 ActiveX DATA OBJECTS (ADO)

ADO is another data access method. ADO data model has a collection of objects using which you can access and manipulate any database that is based on OLEDB interface. ADO model is simple. It has fewest objects in its object model. ADO is the data model for databases that are accessed using OLEDB interface, where as RDO is the data model for databases that are accessed using ODBC interface.

The architecture of Microsoft Universal Data Access (UDA) can support high-performance data access to relational data sources, non-relational data sources and mainframe/ legacy data sources. The UDA architecture (Fig.7.6(b)) consists of three primary components. The OLE DB, the core of the UDA architecture, provides low-level access to any data source. The Open Database Connectivity (ODBC) is a C programming-language library that uses SQL to access data. ActiveX Data Objects (ADO) are simple object models that provide uniform access to any data source by interacting with OLEDB. [Note: OLE DB implements a minimum set of data access services for ADO.]

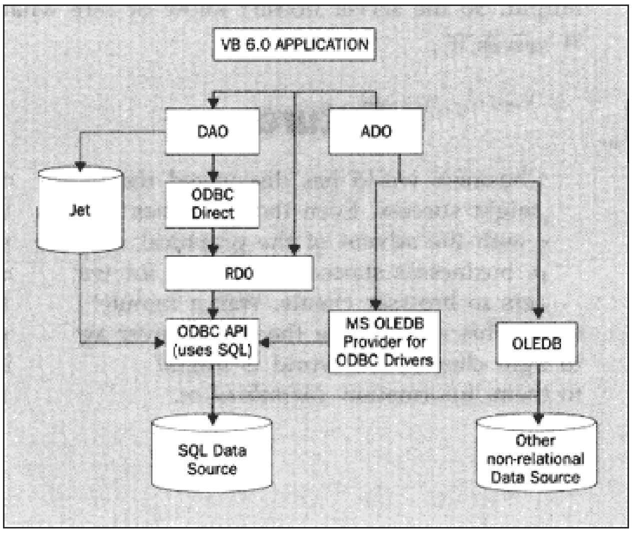


Figure.7.9 (a): Universal Data Access Architecture

Figure.7.9 (b): Universal Data Access Architecture

7.12 SERVER-SIDE ActiveX COMPONENTS

Server-side script functionality is extended with server-side ActiveX components—ActiveX controls that typically reside on the Web server and do not have a graphical user interface. These components make features accessible to the ASP author. Following table summarizes some of the ActiveX components included with Internet Information Services (IIS).

Table.7.4: ActiveX Components

Component name	Description
MSWC	BrowserType Gathers information about the client's browser (e.g., type, version)
MSWC.Content	Rotator Rotates HTML content on aWeb page
MSWC.NextLink	Links Web pages together
MSWC.PageCounter	The number of times a Web page has been requested
MSWC.Counters	Provides general-purpose persistent counters
MSWC.MyInfo	Provides information about aWeb site (e.g., owner name, owner address).
Scripting.FileSystemObject	Provides an object library for accessing files on the server or on the server's network
ActiveX Data Objects (ADO) Data Access Components	Provides an object library for accessing databases

CHECK YOUR PROGRESS

- Define the term ADO.
- Write some ADO components and their meaning.

7.12 SUMMARY

- Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts.
- DBMS is an aggregation of data, hardware, software, and users that help an enterprise manage its operational data.
- DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too.
- DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database.
- A single row in the table is called as tuple and a column stores an attribute of the entity.
- SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus.

- Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions.
- ADO is the data model for databases that are accessed using OLEDB interface, where as RDO is the data model for databases that are accessed using ODBC interface.

7.13 TERMINAL QUESTIONS

1. What do you mean by Database?
2. Give the meaning and main functions of DBMS.
3. Write the name of data models.
4. Compare DBMS and RDBMS.
5. What is the difference between attribute and tuple?
6. Why SQL is very important for any Database?
7. Define different SQL languages with at least one command associated with it.
8. What is ASP?
9. Draw the diagram for Universal Data Access.
10. Justify the role of ActiveX. Write the name and functions of various ActiveX components.

Unit 8 : Web Resources and XML

Structure

8.0 INTRODUCTION

Today, most of the information we need is available on Internet easily. This information is on several websites. You can search this information using different search engines like google, yahoo etc. In general, Web pages and documents on the Internet that provides useful information are the web resources. While an online resource is typically data and educational in nature, any support software available online can be considered a resource.

The World Wide Web (WWW) is now one of our primary information repositories - a vast digital library of documents, software, images, and so on, covering a multitude of subjects and application areas. Still, as we search the information on the Web through search engines like google using different web browsers like Internet Explorer, Google Chrome, Mozilla Firefox etc.

Even so, not only is this infrastructure fundamental to the operation of the Web, but to ensure that the Web continues to deliver information to us, efficiently and reliably, it will need to evolve. The core mechanisms, such as Internet Protocol (IP), are developing to accommodate future growth and future needs. At the application level, initiatives are also under way to impose a more coherent structure on the information resources themselves.

Hypertext markup language (HTML) plays a very important role in creating web pages for the websites or you can say for Internet applications. Websites may be static or dynamic depending on the requirement of the client. For static website; you can use HTML, but for dynamic website you would require Dynamic hypertext markup language (DHTML). DHTML is the combination of three advance technologies: CSS, Javascript, and XML. Extensible markup language or XML in short, is the extension of HTML with advance features. XML is basically used for exchanging the data over Internet.

More or less, directly or indirectly, XML is being used in all the Internet technologies. You can say that a web based application cannot be built without XML. Its parsers DOM and SAX are the key components on which it is based. Other technologies that work to develop the websites are ASP, JSP, PHP etc.

8.1 OBJECTIVES

- To know about web services
- What is WWW
- What is a website
- To know about HTML and DHTML
- Comparison between HTML and DHTML
- XML overview and its components
- Difference between XML and HTML
- How to create a XML program
- To know how to exchange the data over web

8.2 WEB RESOURCES

The basic meaning of the web resource is which can be identified. In other words web resource is the target of uniform resource locator. This is anything that can be named, addressed, identified; controlled, developed using a technique is called a web resource.

The concept of a web resource is primitive in the web architecture, and is used in the definition of its fundamental elements. The term was first introduced to refer to targets of uniform resource locators (URLs), but its definition has been further extended to include the ce of any (RFC 3986), or internationalized resource identifier (RFC 3987). In the Semantic Web, abstract resources and their semantic properties are described using the family of languages based on Resource Description Framework (RDF).

In the early specifications of the web (1990–1994), the term *resource* is barely used at all. The web is designed as a network of more or less static addressable objects, basically files and documents, linked using uniform resource locators (URLs). A web resource is implicitly defined as something which can be identified. The identification deserves two distinct purposes: naming and addressing; the latter only depends on a protocol. It is notable that RFC 1630 does not attempt to define at all the notion of resource; actually it barely uses the term besides its occurrence in URI, URL and URN, and still speaks about "Objects of the Network".

Examples for the web resource may include an electronic document, an image, a service (e.g., "today's weather report for New Delhi"), and a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources. The resource is the conceptual mapping to an entity or set of entities, not necessarily the entity which corresponds to that mapping at any particular instance in time.

Web Resources rely on a special handler that is named WebResource.axd, which is designed to retrieve assembly resources and serve them to the Web browser. The handler type for WebResource.axd is AssemblyResourceLoader.

When a request comes in from the client for WebResource.axd, the handler looks for the Web Resource identifier in the QueryString method of the Request object. Based on the value of the Web Resource identifier, the handler then tries to load the assembly that contains this resource. If this operation is successful, the handler will then look for the assembly attribute and load the resource stream from the assembly. Finally, the handler will grab the data from the resource stream and send it to the client together with the content type that you specify in the assembly attribute.

CHECK YOUR PROGRESS

- What do you understand by a web resource?
- Give the name of files responsible for web resource.

8.3 XML (EXTENSIBLE MARKUP LANGUAGE)

XML is a language defined by the World Wide Web Consortium (W3C, at http://www.w3c.org), the body that sets the standards for the Web. As you know that earlier HTML was known as SGML. SGML appeared along with Netscape navigator with version 1.0. XML, too, is a text-based markup language derived from Standard Generalized Markup Language (SGML). XML stands for Extensible Markup Language. XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data.

When people refer to XML, they typically refer to XML and its related technologies (Fig-8.1). XML is a meta-language that describes the content of the document (self-describing data). It does not specify the tag set or grammar of the language.

- Java = portable programs
- XML = portable data

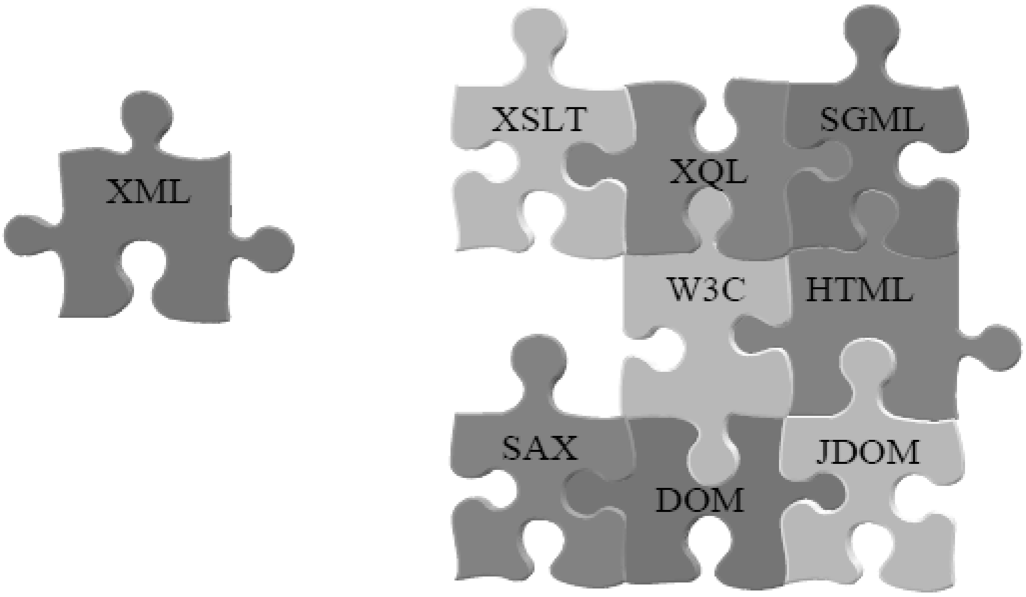


Figure.8.1: XML and related Technologies

8.3.1 Markup Languages

Markup languages are all about describing the form of the document—that is, the way the content of the document should be interpreted. The markup language that most people are familiar with today, of course, is HTML, which you use to create standard Web pages. Here's a sample HTML page:

```
<HTML>
   <HEAD>
      <TITLE>Hello From HTML</TITLE>
   </HEAD>
   <BODY>
      <CENTER>
         <H1> Hello From HTML </H1>
      </CENTER>
      Welcome to the wild and woolly world of HTML.
   </BODY>
</HTML>
```

8.3.2 What Does XML Look Like?

So what does XML look like, and how does it work? Here's an example that mimics the HTML page just introduced:

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
   <GREETING>
      Hello From XML
   </GREETING>
```

```
    <MESSAGE>
        Welcome to the wild and woolly world of XML.
    </MESSAGE>
</DOCUMENT>
```

Here's how this one works:

I start with the XML processing instruction <?xml version="1.0" encoding="UTF-8"?> (all XML processing instructions start with <? and end with ?>), which indicates that I'm using XML version 1.0 (the only version currently defined) and the UTF-8 character encoding, an 8-bit condensed version of Unicode

8.3.3 Characteristics of XML

- **XML is extensible:** XML allows you to create your own self-descriptive tags, or language, that suits your application.
- **XML carries the data, does not present it:** XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard:** XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

8.3.4 XML usage

- XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable. Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

Following example shows how XML markup looks, when embedded in a piece of text:

```
<message>
<text>Hello, world!</text>
</message>
```

This snippet includes the markup symbols, or the tags such as <message>...</message> and <text>...</text>. The tags <message> and </message> mark the start and the end of the XML code fragment. The tags <text> and </text> surround the text Hello, world!.

8.3.5 XML vs HTML

- o XML basically separates contents (both data and language) from presentation; HTML specifies the presentation.
- o HTML explicitly defines a set of legal tags as well as the grammar (intended meaning) eg. <table> …………..</table>. On the other hand XML allows any tags or grammar to be used (hence called extensible).
- o Both are based on Standard Generalized Markup Language (SGML)

8.4 LOGICAL STRUCTURES OF XML

8.4.1 XML Declaration

The XML document can optionally have an XML declaration. It is written as below:

<?xml version="1.0" encoding="UTF-8"?>

Where *version* is the XML version and *encoding* specifies the character encoding used in the document. This is known as file descriptor.

8.4.2 Syntax Rules for XML Declaration

- o The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case.
- o If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- o The XML declaration strictly needs be the first statement in the XML document.
- o An HTTP protocol can override the value of *encoding* that you put in the XML declaration.

8.4.3 Tags and Elements

An XML file is structured by several XML-elements, also called XML-nodes or XMLtags. XML-elements' names are enclosed by triangular brackets < > as shown below:

```
<element>
```

8.4.4 Syntax Rules for Tags and Elements

Each XML-element needs to be closed either with start or with end elements as shown below:

```
<element>....</element>
```

or in simple-cases, just this way:

```
<element/>
```

8.4.5 Attributes

An *attribute* specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes. For example:

```
<a href="http://www.tutorialspoint.com/">Tutorialspoint!</a>
```

Here, *href* is the attribute name and *http://www.tutorialspoint.com/* is attribute value.

Syntax Rules for XML Attributes

- Attribute names in XML (unlike HTML) are case sensitive. That is, *HREF* and *href* are considered two different XML attributes.
- Same attribute cannot have two values in a syntax. The following example shows incorrect syntax because the attribute *b* is specified twice:

  <a b="x" c="y" b="z">....</a>
- Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks. Following example demonstrates incorrect xml syntax:

  <a b=x>....</a>, In this syntax, the attribute value is not defined in quotation marks.

CHECK YOUR PROGRESS

- What do you understand by a XML?
- Why we need XML?

8.5 XML NAMESPACES

A namespace is a set of names in which all names are unique. For example, the names of my children could be thought of as a namespace, as could the names of California corporations, the names of C++ type identifiers, or the names of Internet domains. Any logically related set of names in which each name must be unique is a namespace.

Namespaces make it easier to come up with unique names. Imagine how hard it would be to name your next child if the name had to be unique across the face of the earth. Restricting uniqueness to a more limited context, like my set of children, simplifies things tremendously. When I name my next child, my only consideration is that I don't use the same name that I already used for one of my other children. Another set of parents can choose the same name I choose for one of their children, but those names will be part of distinct namespaces and, therefore, can be easily distinguished.

Before a new name is added to a namespace, a namespace authority must ensure that the new name doesn't already exist in the namespace. In some scenarios this is trivial as it is in child naming. In others it's quite complex. Today's many Internet naming authorities present a good example. If this step is skipped, however, duplicate names will eventually corrupt the namespace, making it impossible to refer to certain names without ambiguity. When this happens, the set of names is no longer officially considered a namespace—by definition a namespace must ensure the uniqueness of its members.

8.5.1 Namespace Declaration

Namespaces are declared as an attribute of an element. It is not mandatory to declare namespaces only at the root element; rather it could be declared at any element in the XML document. The scope of a declared namespace begins at the element where it is declared and applies to the entire content of that element, unless overridden by another namespace declaration with the same prefix name where, the content of an element is the content between the <opening-tag> and </closing-tag> of that element.

A namespace is declared as follows:

```
<someElement xmlns:pfx="http://www.foo.com" />
```

In the attribute xmlns:pfx, xmlns is like a reserved word, which is used only to declare a namespace. In other words, xmlns is used for binding namespaces, and is not itself bound to any namespace. Therefore, the above example is read as binding the prefix "pfx" with the namespace "http://www.foo.com."

It is a convention to use XSD or XS as a prefix for the XML Schema namespace, but that decision is purely personal. One can choose to use a prefix ABC for the XML Schema namespace, which is legal, but doesn't make much sense. Using meaningful namespace prefixes add clarity to the XML document. Note that the prefixes are used only as a placeholder and must be expanded by the namespace-aware XML parser to use the actual namespace bound to the prefix. In Java analogy, a namespace binding can be correlated to declaring a variable, and wherever the variable is referenced, it is replaced by the value it was assigned.

In our previous namespace declaration example, wherever the prefix "pfx" is referenced within the namespace declaration scope, it is expanded to the actual namespace (http://www.foo.com) to which it was bound:

In Java: String pfx = "http://www.library.com"

In XML: <someElement xmlns:pfx="http://www.foo.com" />

Although a namespace usually looks like a URL, that doesn't mean that one must be connected to the Internet to actually declare and use namespaces. Rather, the namespace is intended to serve as a virtual "container" for vocabulary and un-displayed content that can be shared in the Internet space. In the Internet space URLs are unique—hence you would usually choose to use URLs to uniquely identify namespaces. Typing the namespace URL in a browser doesn't mean it would show all the elements and attributes in that namespace; it's just a concept.

8.6 XML DOCUMENTS

A data object is an *XML document* if it is well-formed, as defined in this specification. A well-formed XML document may in addition be valid if it meets certain further constraints. Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup.

8.6.1 Well-Formed XML Documents

A textual object is a well-formed XML document if:

- Taken as a whole, it matches the production labeled *document*.
- It meets all the well-formedness constraints given in this specification.
- Each of the parsed entities which is referenced directly or indirectly within the document is *wellformed*.

Example: Document

[1] document ::= prolog element Misc*

Matching the *document* production implies that:

- It contains one or more elements.
- There is exactly one element, called the *root*, or document element, no part of which appears in the content of any other element. For all other elements, if the start-tag is in the content of another element, the end-tag is in the content of the same element. More simply stated, the elements, delimited by start- and end-tags, nest properly within each other.

As a consequence of this, for each non-root element C in the document, there is one other element P in the document such that C is in the content of P, but is not in the content of any other element that is in the content of P. P is referred to as the *parent* of C, and C as a *child* of P.

8.7 XML VOCABULARIES

“XML Vocabularies a collection of element and attribute names with definitions of their meanings and their structural relationships and constraints". A *schema* or *DTD* might be part of that, but the emphasis is on what the elements mean, which is likely to involve documentation in English or other natural languages. For example the "MusicXML schema" allows you to distinguish valid instances from invalid instances, but the "MusicXML vocabulary" tells you that a particular element/attribute represents a musical note with a given pitch, duration, and loudness.

There are two type of Vocabularies:

1. Common Semantic Vocabularies
2. Specific Semantic Vocabularies

8.7.1 Common Semantic Vocabularies

This section contains XML vocabularies that can be shared among multiple industries or disciplines. Which are as below:

- Address XML

- Computing Environment XML
- Content Syndication XML
- Customer Information XML
- Electronic Data Interchange (EDI) XML
- Geospatial XML
- Human XML
- Localization XML
- Math XML
- Open Applications Group Integration Specification (OAGIS)
- Open Office XML
- Topic Maps XML
- Trade XML
- Translation XML
- Universal Business Language (UBL)
- Universal Data Element Framework (UDEF)

8.7.2 Specific Semantic Vocabularies

This section contains XML vocabularies that can be shared among multiple industries or disciplines. Which are as below:

- Accounting XML
- Advertising XML
- Astronomy XML
- Building XML
- Chemistry XML
- Construction XML
- Education XML
- Finance XML
- Food XML
- Government XML
- Healthcare XML
- Human Resources XML
- Instruments XML
- Insurance XML
- Legal XML
- Manufacturing XML
- News XML
- Oil and Gas XML
- Photo XML
- Physics XML

- Publishing XML
- Real Estate XML

CHECK YOUR PROGRESS

- What do you understand by an XML document?
- Give the meaning of XML vocabulary.

8.8 XML Document Object Model (DOM)

The XML Document Object Model (DOM) class is an in-memory representation of an XML document. The DOM allows you to programmatically read, manipulate, and modify an XML document. The **XmlReader** class also reads XML; however, it provides non-cached, forward-only, read-only access. This means that there are no capabilities to edit the values of an attribute or content of an element, or the ability to insert and remove nodes with the **XmlReader**. Editing is the primary function of the DOM.

It is the common and structured way that XML data is represented in memory, although the actual XML data is stored in a linear fashion when in a file or coming in from another object. The following is XML data.

The following illustration shows how memory is structured when this XML data is read into the DOM structure.

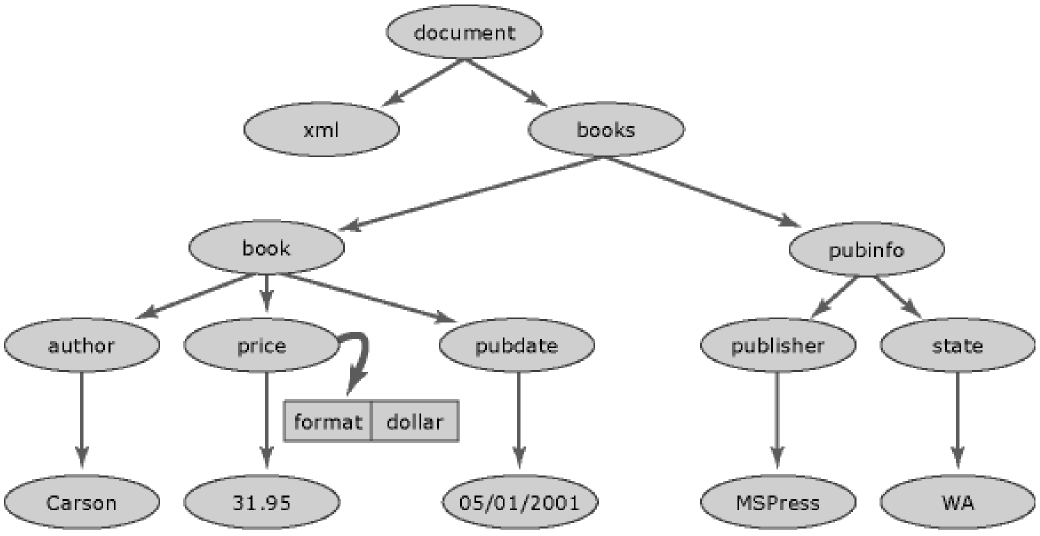


Figure.8.2: XML document structure

Within the XML document structure, as shown in the figure 8.1, each circle represents a node, which is called an XmlNode object. The XmlNode object is the basic object in the DOM tree. The XmlDocument class, which extends XmlNode, supports methods for performing operations on the document as a whole (for example, loading it into memory or saving the XML to a file. In addition, XmlDocument provides a means to view and manipulate the nodes in the entire XML document.

Both XmlNode and XmlDocument have performance and usability enhancements and have methods and properties to:

- o Access and modify nodes specific to the DOM, such as element nodes, entity reference nodes, and so on.
- o Retrieve entire nodes, in addition to the information the node contains, such as the text in an element node.
- o Nodes have a single parent node, a parent node being a node directly above them. The only nodes that do not have a parent is the Document root, as it is the top-level node and contains the document itself and document fragments.
- o Most nodes can have multiple child nodes, which are nodes directly below them. The following is a list of node types that can have child nodes.

One characteristic of the DOM is how it handles attributes. Attributes are not nodes that are part of the parent, child, and sibling relationships. Attributes are considered a property of the element node and are made up of a name and a value pair. For example, if you have XML data consisting of format="dollar" associated with the element price, the word format is the name, and the value of the format attribute is dollar. To retrieve the format="dollar" attribute of the *price* node; you call the *GetAttribute* method when the cursor is located at the price element node.

As XML is read into memory, nodes are created. However, not all nodes are the same type. An element in XML has different rules and syntax than a processing instruction. Therefore, as various data is read, a node type is assigned to each node. This node type determines the characteristics and functionality of the node.

The DOM is most useful for reading XML data into memory to change its structure, to add or remove nodes, or to modify the data held by a node as in the text contained by an element. However, other classes are available that are faster than the DOM in other scenarios. For fast, non-cached, forward-only stream access to XML, use the XmlReader and XmlWriter. If you need random access with a cursor model and XPath, use the XPathNavigator class.

CHECK YOUR PROGRESS

- o What is main function of DOM?

8.9 XML PARSERS

XML parsers are software packages that you use as part of an application such as Oracle 8i (which includes good XML support) or as part of your own programs. For example, IBM AlphaWorks XML for Java (XML4J) parser; it is written in Java and connects well to your own Java code. Here's a list of some of the parsers out there:

- o **SAX:** This is the abbreviation for Simple API for XML. SAX is a well-known parser that uses event-based parsing. I'll use SAX in this book.

- **expat:** This famous XML parser was written in the C programming language by James Clark. This parser is used in Netscape Navigator 6 and in the Perl language's XML::Parser module.
- **expat as a Perl Module:** XML::Parser is maintained by Clark Cooper.
- **TclExpat:** This is expat written for use in the Tcl programming language by Steve Ball. Superceded by TclXML.
- **LT XML:** This is an XML developers' toolkit from the Language Technology Group at the University of Edinburgh.
- **XML for Java –(XML4J):** From IBM AlphaWorks, this is a famous and very widely used XML parser that adheres well to the W3C standards.
- **XML Microsoft's validating XML processor:** This parser requires Internet Explorer 4.01 SP1 and later in order to be fully functional.
- **Lark:** This is a non-validating XML processor that was written in Java by Tim Bray, and it's one of the famous ones that have been around a long time.
- **XP:** XP is a non-validating XML processor written in Java by James Clark.
- **Python and XML Processing Preliminary XML Parser:** This parser offers XML support to the Python programming language.
- **TclXML:** This XML parser was written in Tcl by Steve Ball.
- **XML Testbed:** This parser was written by Steve Withall.
- **SXP:** Silfide XML Parser (SXP) is another famous XML parser and, in fact, a complete XML Application Programming Interface (API) in Java.
- **The Microsoft XML Parser:** The parser used in Internet Explorer is implemented as a COM component at.
- **OmiMark 5 Programming Language:** Includes integrated support for parsing and validation of XML.
- **Java Standard Extension for XML:** Because XML and Sun Microsystem's Java is such a popular mix, Sun is getting into the act with its own Java package for XML.

8.10 APPLICATIONS OF XML

The world of XML is huge these days; in fact, XML is now used internally even in Netscape and Microsoft products, as well as installations of programming languages such as Perl. It's useful and encouraging to see how XML is being used today in these XML-based languages. It's a new piece of terminology: As you know, XML is a meta-markup language, so it's actually used to create languages. The languages so created are applications of XML, so they're called XML applications.

Note that the term XML application refers to an application of XML to a specific domain, such as MathML, the mathematics markup language; it does not refer to a program that uses XML (a fact that causes a lot of confusion among people who know nothing about XML).

You can see the advantage to various groups (such as physicists or chemists) for defining their own markup languages, allowing them to use the symbols and graphics of their discipline in customized browsers. Some of the applications are:

- o The first application of XML is to configure the various files which are necessary and used in J2EE architectures. As you know that servlet acts as a controller where web.xml file is used for the mapping.
- o It can be used in media for data interchange so that many formats of data support.
- o As you also know that today business-to-business (B2B) transactions on the web are widely used. It could also be done using XML only. Some well known transactions are: Electronic business orders (ebXML), Financial Exchange (IFX), and Messaging Exchange (SOAP).

8.11 WHY XML IS IMPORTANT

There are a number of reasons for XML's surging acceptance:

8.11.1 Plain Text

Since XML is not a binary format, you can create and edit files with anything from a standard text editor to a visual development environment. That makes it easy to debug your programs, and makes it useful for storing small amounts of data. At the other end of the spectrum, an XML front end to a database makes it possible to efficiently store large amounts of XML data as well. So XML provides scalability for anything from small configuration files to a company-wide data repository.

8.11.2 Data Identification

XML tells you what kind of data you have, not how to display it. Because the markup tags identify the information and break up the data into parts, an email program can process it, a search program can look for messages sent to particular people, and an address book can extract the address information from the rest of the message. In short, because the different parts of the information have been identified, they can be used in different ways by different applications.

8.11.3 Style

When display is important, the style-sheet standard, XSL, lets you dictate how to portray the data. For example, the style-sheet for:

<to>you@yourAddress.com</to>

Or can say:

1. Start a new line.
2. Display "To:" in bold, followed by a space

3. Display the destination data.

Which produces :

To: you@yourAddress

Of course, you could have done the same thing in HTML, but you wouldn't be able to process the data with search programs and address-extraction programs and the like. More importantly, since XML is inherently style-free, you can use a completely different stylesheet to produce output in postscript, TEX, PDF, or some new format that hasn't even been invented yet. That flexibility amounts to what one author described as "future-proofing" your information. The XML documents you author today can be used in future document-delivery systems that haven't even been imagined yet.

8.11.4 Inline Reusabiliy

One of the nicer aspects of XML documents is that they can be composed from separate entities. You can do that with HTML, but only by linking to other documents. Unlike HTML, XML entities can be included "in line" in a document. The included sections look like a normal part of the document -- you can search the whole document at one time or download it in one piece. That lets you modularize your documents without resorting to links. You can single-source a section so that an edit to it is reflected everywhere the section is used, and yet a document composed from such pieces looks for all the world like a one-piece document.

8.11.5 Linkability

The ability to define links between documents is now regarded as a necessity. This initiative lets you define two-way links, multiple-target links, "expanding" links (where clicking a link causes the targeted information to appear inline), and links between two existing documents.

8.11.6 Easily Processed

Regular and consistent notation makes it easier to build a program to process XML data. For example, in HTML a `<dt>` tag can be delimited by `</dt>`, another `<dt>`, `<dd>`, or `</dl>`. That makes for some difficult programming. But in XML, the `<dt>` tag must always have a `</dt>` terminator, or else it will be defined as a `<dt/>` tag. That restriction is a critical part of the constraints that make an XML document well-formed. Otherwise, the XML parser won't be able to read the data. And since XML is a vendor-neutral standard, you can choose among several XML parsers, any one of which takes the work out of processing XML data.

8.11.7 Hierarchical

Finally, XML documents also benefit from their hierarchical structure. Hierarchical document structures are, in general, faster to access because you can drill down to the part you need, like stepping through a table of contents. They are also easier to rearrange, because each piece is delimited. In a document, for example, you could move a heading to a new location and drag everything under it along with the heading, instead of having to page down to make a selection, cut, and then paste the selection into a new location.

8.12 SUMMARY

- A web resource is implicitly defined as something which can be identified. The identification deserves two distinct purposes: naming and addressing; the latter only depends on a protocol.
- In common, a Web resource is anything that has an identity, typically the identity would be a Uniform Resource Identifier (URI).
- XML stands for E**X**tensible **M**arkup **L**anguage.
- XML was designed to store and transport data.
- XML was designed to be both human- and machine-readable.
- XML is self describing meta data
- XML is a text-based markup language that is fast becoming the standard for data interchange on the Web. As with HTML, you identify data using tags (identifiers enclosed in angle brackets, like this: <...>). Collectively, the tags are known as "markup".
- But unlike HTML, XML tags tell you what the data *means*, rather than how to display it. Where an HTML tag says something like "display this data in bold font" (`<b>...</b>`), an XML tag acts like a field name in your program. It puts a label on a piece of data that identifies it (for example: `<message>...</message>`).
- The minimal prolog contains a declaration that identifies the document as an XML document, like this:

  <?xml version="1.0"?>
- DOCTYPE defines the root element and location of DTD
- Document Type Definition defines the grammar of the document
- The processing of DTD is expensive
- Schema uses XML to specify the grammar

8.13 TERMINAL QUESTIONS

1. Give the real meaning of web resource.
2. Define the web resources.
3. What does XML stand for?
4. How can you use XML?

5. What do you mean by parsers?
6. Write the code for XML file descriptor.
7. Compare DOM and SAX parser.
8. Give the names of some applications where you can easily use XML.
9. What is the difference between HTML and XML?
10. Is there any other option of XML?

U. P. RAJARSHI TANDON
OPEN UNIVERSITY

BCA-E9

Bachelor in Computer Applications

Block

4

ADVANCE JAVA CONCEPTS

Course Design Committee

Course Preparation Committee

Dr. Krishan Kumar **Author**
Assistant Professor,
Department of Computer Science,
Gurukula Kangri Vishwavidyalaya, Haridwar (UK)

Dr. Ravendra Singh **Editor**
Reader, Computer Science & Information Technology
MJP Rohilkhand University, Bareilly (UP) INDIA

Dr. Ashutosh Gupta
Director In-charge
School of Computer & Information Sciences,
UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor, School of Sciences,
UPRTOU, Allahabad

BLOCK-4 INTRODUCTION

Block-4 basically contains two units in all which mainly revolves around the concepts of advance Java. This block incorporates Unit-9 and Unit-10. As we know that advance Java or J2EE (Java 2 Enterprise Edition) is the extension of core java or Java 1. Advance Java has the features like servlet, beans, JSP etc.

Unit-9 describes the role and functionality of Java Servlets. Servlet fulfills the requirement of request made at the client-side by running Java programs at server side. In this unit, the main focus is to understand the important advance java concept i.e. servlet. Today, this is the demand and need of most of the computer programs to be deployed on the Internet. As we know servlet is very popular server side programming which is different from the client side java programs. The servlet technology is the foundation of web application development using the Java programming language. As we all know, process creation is an expensive operation that consumes a lot of CPU cycles and computer memory. These all issues of CGI are handled in servlet because the concept of multithreading.

Therefore, understanding the servlet technology and its architecture is important if you want to be a servlet developer. Even if you plan to develop your Java web application using JSP pages alone, understanding the servlet technology helps you build a more efficient and effective JSP application. It also acts the controller in model view controller architecture.

Unit-10 explains about another popular technology of Java which basically provides a template for HTML, Java, and JSP itself syntax. These are also the Java programs which can be run st server side with very less coding and time saving approach. It is difficult for people who are not Java programmers to implement, maintain and extend a Web application that consists of primarily of servlet. The solution to this problem is Java Server Pages (JSP) an extension of servlet technology that separates the presentation from the business logic. This lets Java programmers and Web site designers focus on their strengths writing Java code and designing Web pages, respectively. Java Server Pages simplify the delivery of dynamic Web content. They enable Web application programmers to create dynamic content by reusing predefined components and by interacting with components using server-side scripting.

Java Server Page programmers can use special software components called Java Beans and custom tag libraries that encapsulate complex, dynamic functionality. A Java Bean is a reusable component that follows certain conventions for class design that are discussed in the Java Beans specification. Custom-tag libraries are a powerful feature of JSP that allows Java developers to hide complex code for database access and other useful services for dynamic Web pages in custom tags. Web sites use these custom tags like any other Web page element to take advantage of the more complex functionality hidden by the tag. Thus, Web-page designers who are not familiar with Java can enhance Web pages with powerful dynamic content and processing capabilities.

Unit - 9 : Introduction to Servlets

Structure

9.0 INTRODUCTION

In this unit, the main focus is to understand the important advance Java concept i.e. servlet. Today, this is the demand and need of most of the computer programs to be deployed on the Internet. As we know servlet is very popular server side programming which is different from the client side Java programs. The servlet technology is the foundation of web application development using the Java programming language. It is one of the most important Java technologies, and it is the underlying technology for another popular Java technology for web application development: Java Server Pages (JSP).

As the Internet became more and more popular, however, the number of users visiting a popular web site increased exponentially, and it became apparent that CGI had failed to deliver scalable Internet applications. The flaw in CGI is that each client request makes the web server spawn a new process of the requested CGI program. As we all know, process creation is an expensive operation that consumes a lot of CPU cycles and computer memory. These all issues of CGI are handled in servlet because the concept of multithreading.

Therefore, understanding the servlet technology and its architecture is important if you want to be a servlet developer. Even if you plan to develop your Java web application using JSP pages alone, understanding the servlet technology helps you build a more efficient and effective JSP application. It also acts the controller in model view controller architecture.

In this unit we will come to know about the about the servlet programming and further the usefulness of JSP programming. JSP is a template in which we can keep java code, html and jsp code simultaneously at one place together.

9.1 OBJECTIVES

The main aim of this unit is to introduce the servlet technology and make you comfortable with it by presenting step-by-step instructions that enable you to build and run a servlet application. In particular, this unit aims to achieve the following objectives:

- Motivation and origination of servlet
- Basics of servlet
- Servlet life cycle
- How to write the code and run your first servlet program
- The benefits of servlet
- Demerits of servlet
- Single tier and multitier application
- Difference between servlet and JSP

9.2 OVERVIEW OF SERVLETS

Servlets are used primarily with web servers, where they provide a Java-based replacement for CGI scripts. They can be used to provide dynamic web content like CGI scripts. Advantages of servlets over CGI scripts:

- Servlets are persistent between invocations, which dramatically improves performance relative to CGI programs.
- Servlets are portable among operating systems and among servers.
- Servlets have access to all the APIs of the Java platform (e.g. a servlet can interact with a database using JDBC API).

Servlets are a natural fit if you are using the web for enterprise computing. Web browsers then function as universally available thin clients; the web server becomes middleware responsible for running applications for these clients. Thus the user makes a request of the web server, the server invokes a servlet designed to handle the request, and the result is returned to the user in the web browser. The servlet can use JNDI, Java IDL, JDBC, and other enterprise APIs to perform whatever task is necessary to fulfill the request.

- Servlets can be used when collaboration is needed between people. A servlet can handle multiple requests concurrently, and can synchronize requests. So servlets can support on-line conferencing.
- Servlets can forward requests to other servers and servlets. Thus, servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical service over several servers, according to task type or organizational boundaries.

9.2.1 The Servlet Life Cycle

When a client (web browser) makes a request involving a servlet, the web server loads and executes the appropriate Java classes. Those classes generate content (e.g. HTML), and the server sends the contents back to the client. From the web browser's perspective, this isn't any different from requesting a page generated by a CGI script, or standard HTML. On the server side there is one important difference: persistence. Instead of shutting down at the end of each request, the servlet remains loaded, ready to handle the subsequent requests. Each request is handled by a separate thread. These threads share code and data (instance vars). So try to avoid using instance vars, else be sure to use them in synchronized blocks.

- The request processing time for a servlet can vary, but is typically quite fast when compared to a similar CGI program. The advantage in the servlet is that you incur the most of the startup overhead only once.
- When a servlet loads, its *init()* method is called. You can use *init()* to create I/O intensive resources, such as database connections, for use across multiple invocations. If you have a high-traffic site, the performance benefits can be quite dramatic.Instead of creating thousands of database connections, the servlet needs to create a connection only once.
- The servlet's *destroy()* method can clean up resources when the server shuts down.

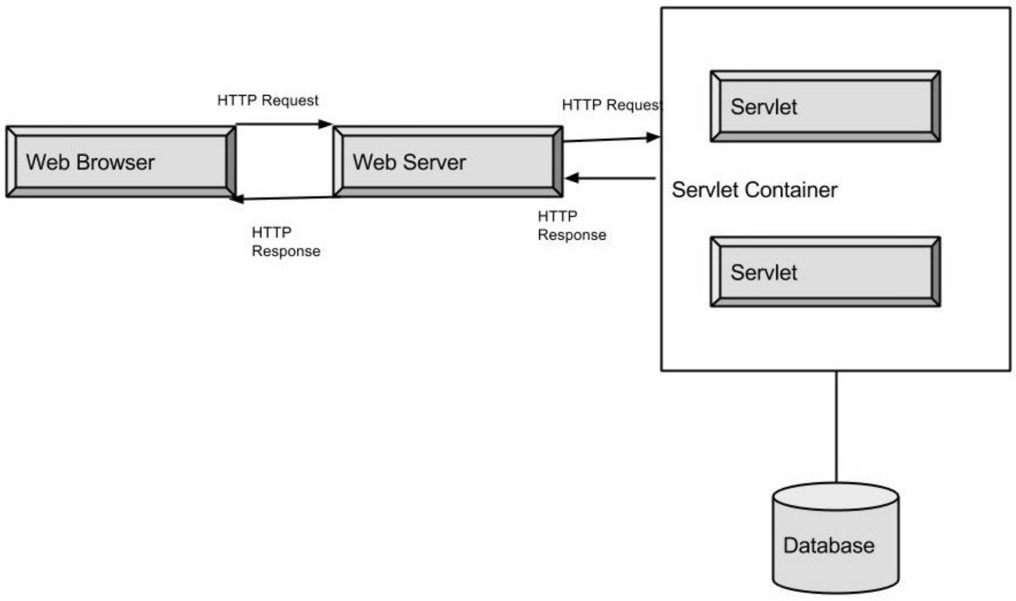
Because servlets are persistent, you can eliminate a lot of file system and/or database accesses altogether. E.g., to implement a page counter, you can simply store a number in a static variable, rather than consult a file (or database) for every request. Thus you need to read and write to disk only occasionally to save state.

Since a servlet remains active, it can perform other tasks when it is not servicing client request, such as running a background thread (where clients connect to the servlet to view the result) or even acting as an RMI host, enabling a single servlet to handle connections from multiple types of clients. E.g., a servlet that accepts transactions from both an HTML form and an applet using RMI.

9.3 SERVLET ARCHITECTURE

It is necessary to understand the architecture before knowing the concept of servlet. the process of Servlets in a Web Application is shown in the Figure 9.1.The process can be summarized as follows:

- A client sends a request to a Web Server, through a Web Browser.
- The Web Server searches for the required Servlet and initiates it.
- The Servlet then processes the client request and sends the response back to the server, which is then forwarded to the client.



○

Figure 9.1: Process of Servlets in a Web Application

9.3.1 Servlet Basics

The Servlet API consists of two packages, *javax.servlet*, and *javax.servlet.http*. The javax is there because servlets are a standard extension to Java, rather than a mandatory part of the API. Thus JVM developers are not required to include classes for them in their Java development and execution environments. Sun has kept the distribution of the servlets API separate from the Java 2 platform because the Servlet API is evolving much faster than the core Java SDK. The JSDK includes the necessary servlet classes and a small servletrunner application for development and testing.

HTTP Servlets

The *HttpServlet* class is an extension of *GenericServlet* class that includes methods for handling HTTP specific data. *HttpServlet* defines a number of methods, such as *doGet()*, and *doPost()*, to handle particular types of HTTP requests (GET, POST, etc.). These methods are called by the default implementation of the *service()* method, which figures out the kind of request being made and then invokes the appropriate method. Method service() is defined in *GenericServlet* class.

Example 1: Simple Hello Servlet

```
import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

public class HelloWorldServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws

ServletException, IOException {
```

```
resp.setContentType("text/html");

PrintWriter out = resp.getWriter();

out.println("<HTML>");

out.println("<HEAD><TITLE>First Servlet</TITLE></HEAD>");

out.println("<BODY><H1>Hello World!</H1></BODY>");

out.println("</HTML>");

   }

}
```

doGet() method

The *doGet()* method is called whenever anyone requests a URL that points to this servlet. The servlet is installed in the *servlets* directory (this is similar to the cgi-bin directory for CGI programs), and its URL is http://site:8080/servlet/HelloWorldServlet. The *doGet()* method is actually called by the default *service()* method of *HttpServlet*. The *service()* method is called by the web server when a request is made of *HelloWorldServlet*; the method determines what kind of HTTP request is being made and dispatches the request to the appropriate *doXXX()* method (in this case, *doGet()*). *doGet()* is passed two objects, *HttpServletRequest* ,and *HttpServletResponse,* that contain information about the request and provide a mechanism for the servlet to provide a response, respectively.

Example 2: This example deals with forms and dynamic HTML. The HTML form that calls the servlet using a GET request is as follows:

```
<HTML>
<HEAD><TITLE> Greetings Form</TITLE></HEAD>
<BODY>
<FORM METHOD=GET   ACTION="/servlet/HelloServlet">
What is your name?
<INPUT TYPE=TEXT   NAME=username SIZE=20>
<INPUT TYPE=SUBMIT   VALUE="Introduce Yourself">
</FORM>
</BODY>
</HTML>
```

The form submits a variable named username to the URL /servlet/HelloServlet. The servlet is as follows:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
public class HelloServlet extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
resp.setContentType("text/html");
PrintWriter out = resp.getWriter();
out.println("<HTML>");
out.println("<HEAD><TITLE>Finally, interaction!</TITLE></HEAD>");
out.println("<BODY><H1>Hello," + req.getParameter("username") + "!</H1>" );
out.println(</BODY>");
out.println("</HTML>");
    }
}
```

The getParameter() method of HttpServletRequest is used to retrieve the value of the form variable. When a server calls a servlet, it can also pass a set of request parameters.

The POST request

The POST request is designed for posting information to the server, although in practice it is also used for long parameterized requests and larger forms, to get around limitations on the length of URLs. The *doPost()* method is the corresponding method for POST requests. If your servlet is performing database updates, charging a credit card, or doing anything that takes an explicit client action, you should make sure that this activity is happening in a *doPost()* method.

This is because POST requests are not idempotent, which means that they are not safely repeatable, and web browsers treat them specially. E.g. a browser cannot bookmark them. GET requests are idempotent, so they can safely be bookmarked, and a browser is free to issue the request repeatedly (say to get some information from the web server) without necessarily consulting the user. Hence it is not appropriate to charge a credit card in a GET method.

To create a servlet that can handle POST requests, you need to override the default doPost() method from HttpServlet and implement the necessary functionality. If necessary, your application can implement different code in doPost() and doGet(). For instance, the doGet() method might display a postable data entry form that the doPost() method processes. doPost() can even call doGet() at the end to display the form again.

Difference between doGet and doPost method

	doGet()	**doPost()**
History	Parameters remain in browser history because they are part of the URL	Parameters are not saved in browser history.

Bookmarked	Can be bookmarked.	Can not be bookmarked.
BACK button/re-submit behavior	GET requests are re-executed but may not be re-submitted to server if the HTML is stored in the browser cache.	The browser usually alerts the user that data will need to be re-submitted.
Encoding type (enctype attribute)	application/x-www-form-urlencoded	multipart/form-data or application/x-www-form-urlencoded Use multipart encoding for binary data.
Parameters	can send but the parameter data is limited to what we can stuff into the request line (URL). Safest to use less than 2K of parameters, some servers handle up to 64K	Can send parameters, including uploading files, to the server.
Hacking	Easy to hack	Difficult to hack
Restrictions on form data type	Yes, only ASCII characters allowed.	No restrictions. Binary data is also allowed.
Security	GET is less secure compared to POST because data sent is part of the URL. So it's saved in browser history and server logs in plaintext.	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs.
Restrictions on form data length	Yes, since form data is in the URL and URL length is restricted. A safe URL length limit is often 2048 characters but varies by browser and web server.	No restrictions
Usability	GET method should not be used when sending passwords or other sensitive information.	POST method used when sending passwords or other sensitive information.
Visibility	GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount	POST method variables are not displayed in the URL.

	of information to send.	
Cached	Can be cached	Not cached

Servlet Responses

In the case of an Http servlet, the response can include three components: a status code, any number of HTTP headers, and a response body. You use setContentType() method of the response object passed into the servlet to set the type of the response. Examples include "text/html" for text, "image/gif" for returning a GIF file from the database, and "application/pdf" for Adobe Acrobat files.

ServletResponse and *HttpServletResponse* each define two methods for producing output streams, *getOutputStream()* and *getWriter()*. The former returns a *ServletOutputStream* that can be used for text or binary data. The latter returns a *java.io.PrintWriter* object used for text data.

You can use *setStatus()* or *sendError()* method to specify the status code sent back to the server. Examples include, 200 ("OK"), 404 ("Not Found") etc. The *sendRedirect()* method allows you to issue a page redirect. Calling this sets the Location header to the specified location and uses the appropriate status code for a redirect.

Servlet Requests

When a servlet is asked to handle a request, it typically needs specific information about the request so that it can process the request appropriately. For example, a servlet may need to find out about the actual user who is accessing the servlet, for authentication purposes.

ServletRequest and *ServletRequest* provide these methods. Examples include *getProtocol()* (protocol used by request), *getRemoteHost()* (client host name), *getServerName()* (name of the web server), *getServerPort()* (port number the web server listens at), *getParameter()* (access to request parameters as form variables), and *getParameterValues()* (returns an array of strings that contains all the values for a particular parameter).

Example 3 : This servlet requests information to restrict access.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class SecureRequestServlet extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
  resp.setContentType("text/html");
PrintWriter out = resp.getWriter();
out.println("<HTML>");
out.println("<HEAD><TITLE>Semi-Secure Request</TITLE></HEAD>");
out.println("<BODY>");
String remoteHost = req.getRemoteHost();
String scheme = req.getScheme();
String authType = req.getAuthType();
if((remoteHost == null) || (scheme == null) || (authType == null)) {
out.println("Request information was not available");
return;
}
// look for a secure HTTP connection from a .gov host using Digest style ///authentication
if(scheme.equalsIgnoreCase("https") && remoteHost.endsWith(".gov") &&
authType.equals("Digest")) {
out.println("Special, secret information");
}
else
{
out.println("You are not authorized to read this information");
}
out.println("</BODY></HTML>");
 }
   }
```

Error Handling

If the error is part of a servlet's normal operation, such as when a user forgets to fill in a required form field, then write an error message to the servlet's output stream. If the error is a standard HTTP error, use the *sendError()* method of *HttpServletResponse* to tell the server to send a standard error status code. E.g. if a file was not found,

```
resp.sendError(HttpServletResponse.SC_NOT_FOUND);
```

Example 4: This servlet serves HTML files.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class FileServlet extends HttpServlet {
```

```
public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
            ServletException, IOException {
            File r;
      FileReader fr;
      BufferedReader br;
              try {
r = new File(req.getParameter("filename"));
fr = new FileReader(r);
br = new BufferedReader(fr);
if (!r.isFile()) {
    resp.sendError(resp.SC_NOT_FOUND);
    return;
}
       }
       catch (FileNotFoundException e) {
resp.sendError(resp.SC_NOT_FOUND);
return;
       }
         catch (SecurityException se) {    //Be unavailable permanently
throw(new UnavailableException(this, "Servlet lacks appropriate
privileges"));
       }
      resp.setContentType("text/html");
      PrintWriter out = resp.getWriter();
      String text;
      while ((text = br.readLine()) != null)
out.println(text);
      br.close();
    }

  }
```

Servlet Initialization

When a server loads a servlet for the first time, it calls the servlet's *init()* method. In its default implementation, *init()* handles some basic housekeeping, but a servlet can override the method to perform other tasks. This includes performing I/O intensive tasks such as opening a database connection. You can also create threads in *init()* to perform other tasks such as pinging other machines on the network to monitor the status of these machines. When an actual request occurs, the service methods can use the resources created in *init().* The default *init()* is not a do-nothing method. You must call always call *super.init()* as the first action in your own init() routines.

The server passes the *init()* method a *ServletConfig* object. This object encapsulates the servlet initialization parameters, which are accessed via the *getInitParameter()* and *getInitParameterNames()* methods. *GenericServlet* and *HttpServlet* both implement the *ServletConfig* interface, so these methods are always available in a servlet. Consult your server documentation on how to set the initialization parameters.

Each servlet also has a *destroy()* method that can be overwritten. This method is called when a server wants to unload a servlet. You can use this method to free important resources.

Example 5 : This is a persistent counter servlet that saves its state between server shutdowns. It uses the *init()* method to first try to load a default value from a servlet initialization parameter. Next the *init()* method tries to open a file named */data/counter.dat* and read an integer from it. When the servlet is shut down, the *destroy()* method creates a new *counter.dat* file with the current hit-count for the servlet.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LifeCycleServlet extends HttpServlet {
int timesAccessed;
public void init(ServletConfig conf) throws ServletException  {
super.init(conf);
// Get initial value
try {
   timesAccessed = Integer.parseInt(getInitParameter("defaultStart"));
}
catch (NullPointerException e) {
   timesAccessed = 0;
}
```

```
catch (NumberFormatException e) {
  timesAccessed = 0;
}
// Try loading from the disk
try {
  File r = new File(“./data/counter.dat”);
  DataInputStream ds = new DataInputStream(new FileInputStream(r));
  timesAccessed = ds.readInt();
}
catch (FileNotFoundException e){
 //Handle error
}
catch (IOException e) {
 //Handle error
}
finally {
  ds.close();
}
    }//end init()


public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
          ServletException, IOException {


    resp.setContentType(“text/html”);
    PrintWriter out = resp.getWriter();
    timesAccessed++;


    out.println(“<HTML>”);
    out.println(“<HEAD><TITLE>Life Cycle Servlet</TITLE></HEAD>”);
          out.println(“<BODY>”);
```

```
            out.println("I have been accessed " + timesAccessed + "time[s]");
            out.println("</BODY></HTML>");
}
public void destroy() {
            // Write the integer to a file
            File r = new File("./data/counter.dat");
            try {
            DataOutputStream dout = new DataOutputStream(new
            FileOutputStream(r));
            dout.writeInt(timesAccessed);
            }
            catch (IOException e) {
            // Handle error, maybe log the error
            }
            finally (
             dout.close();
            }
}
}
```

CHECK YOUR PROGRESS

- What do you understand by servlet?
- Compare and contrast servlet with CGI.
- Write a small program using servlet to print the "This is advance Java/J2EE".
- Explain out.println syntax.

9.4 SMALL APPLICATION FOR BANK ATM

This servlet implements an ATM display. The doGet() method displays the current account balance and provides a small ATM control panel for making deposits and withdrawals. The control panel uses a

POST request to send the transaction back to the servlet, which performs the appropriate action and calls doGet() to redisplay the ATM screen with the updated balance. This example also demonstrates thread safety.

Example

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;
public class AtmServlet extends HttpServlet {
Account act;
public void init (ServletConfig conf) throws ServletException {
super.init();
act = new Account();
act.balance = 0;
}
public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
          ServletException, IOException {
     resp.setContentType("text/html");
     PrintWriter out = resp.getWriter();
               out.println("<HTML><BODY>");
     out.println("<H2> First Bank of Java ATM</H2>");
     out.println("Current Balance: <B>" + act.balance + "</B><BR>");
     out.println("<FORM METHOD=POST   ACTION=/servlet/AtmServlet>");
               out.println("Amount: <INPUT TYPE=TEXT    NAME=AMOUNT
SIZE=3><BR>");
     out.println("INPUT TYPE=SUBMIT  NAME=DEPOSIT
                                                  VALUE=\"Deposit\">");
     out.println("INPUT TYPE=SUBMIT  NAME=WITHDRAW
                                                  VALUE=\"Withdraw\">");
      out.println("</FORM>");
      out.println("</BODY></HTML>");
}
 public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
```

```
            ServletException, IOException {
int amt = 0;
try {
amt = Integer.parseInt(req.getParameter(“AMOUNT”));
}
catch (NullPointerException e) {
// No amount Parameter passed
}
catch (NumberFormatException e) {
//Amount parameter was not a number
}
synchronized(act) {
if (req.getParameter(“WITHDRAW”) != null) && (amt < act.balance))
   act.balance = act.balance – amt;
if (req.getParameter(“DEPOSIT”) != null) && (amt > 0)
   act.balance = act.balance + amt;
} //end synchronized block


doGet(req, resp); // Show ATM screen
    } // end doPost()
public void destroy() {
// save balance to a file before servlet is unloaded. If servlet used JDBC to // write to a database, need
to destroy all database resources here.
}
class Account {
public int balance;
}
}
```

9.5 THE TOMCAT SERVLET CONTAINER

A number of servlet containers are available today. The most popular one—and the one recognized as the official servlet/JSP container—is Tomcat. Originally designed by Sun Microsystems, Tomcat source code was handed over to the Apache Software Foundation in October 1999. In this new home, Tomcat was included as part of the Jakarta Project, one of the projects of the Apache Software Foundation. Working through the Apache process, Apache, Sun, and other companies—with the help of volunteer programmers worldwide—turned Tomcat into a world-class servlet reference implementation. Two months after the handover, Tomcat version 3.0 was released. Tomcat went through several 3.x releases until version 3.3 was introduced.

The successor of version 3.3 is the current version, version 4.0. The 4.0 servlet container (Catalina) is based on a completely new architecture and has been developed from the ground up for flexibility and performance. Version 4.0 implements the Servlet 2.3 and JSP 1.2 specifications, and it is this version you will be using in this course.

Another popular servlet container is JRun from Allaire Corporation. JRun is available in three editions: Developer, Professional, and Enterprise. The Developer edition is free but not licensed for deployment. The Professional and Enterprise editions grant you the license for deployment with a fee.

Tomcat by itself is a web server. This means that you can use Tomcat to service HTTP requests for servlets, as well as static files (HTML, image files, and so on). In practice, however, since it is faster for non-servlet, non-JSP requests, Tomcat normally is used as a module with another more robust web server, such as Apache web server or Microsoft Internet Information Server. Only requests for servlets or JSP pages are passed to Tomcat.

To write a servlet, you need at least version 1.2 of the Java Development Kit. If you have not already downloaded one, you can download JDK 1.2 from http://java.sun.com/j2se. The reference implementation for both servlets and JSP are not included in J2SE, but they are included in Tomcat. Tomcat is written purely in Java.

9.6 ADVANTAGES OF A SERVLET

- Servlets provide component based and platform-independent methods for building Web based applications.
- Each Request is run in a separate thread ,so servlet request processing is faster than CGI.
- Servlets overcomes the limitations of CGI program.
- Servlets run on Java Virtual Machine and in any platform and it is simple to write.
- Servlet are more powerful and the performance is better.
- Servlets are platform-independent.
- Servlet technology, in addition to improved performance, offers security, robustness, object orientation, and platform independence.
- As mentioned in the definition of servlets are fully integrated with the Java language and its standard APIs. Hence JDBC for Java database connectivity is also integrated in it.
- A servlet handles concurrent requests
- Handling HTTP requests and send text and data back to the client is made easy by servlet request and response objects.

Gradually, new and better technologies will replace CGI as the main technology for web application development. The world has witnessed the following technologies trying to dominate web development:

- ColdFusion. Allaire's ColdFusion provides HTML-like custom tags that can be used to perform a number of operations, especially querying a database. This technology had its glamorous time in the history of the World Wide Web as the main technology for web application programming. Its glorious time has since gone with the invention of other technologies.
- Server-side JavaScript (SSJS). SSJS is an extension of the JavaScript language, the scripting language that still rules client-side web programming. SSJS can access Java classes deployed at the server side using the LiveWire technology from Netscape.
- PHP. PHP is an exciting open-source technology that has matured in recent years. The technology provides easy web application development with its session management and includes some built-in functionality, such as file upload. The number of programmers embracing PHP as their technology of choice has risen sharply in recent years.
- Servlet. The servlet technology was introduced by Sun Microsystems in 1996. This technology is the main focus of this book and will be explained in more detail in this and coming chapters.
- JavaServer Pages (JSP). JSP is an extension of the servlet technology. This, too, is the center of attention in this book.
- Active Server Pages (ASP). Microsoft's ASP employs scripting technologies that work in Windows platforms, even though there have been efforts to port this technology to other operating systems. Windows ASP works with the Internet Information Server web server. This technology will soon be replaced by Active Server Pages.NET.
- Active Server Pages.NET (ASP.NET). This technology is part of Microsoft's .NET initiative. Interestingly, the .NET Framework employs a runtime called the Common Language Runtime that is very similar to Java Virtual Machine and provides a vast class library available to all .NET languages and from ASP.NET pages. ASP.NET is an exciting technology. It introduced several new technologies including state management that does not depend on cookies or URL rewriting.

9.7 DISADVANTAGES OF A SERVLET

- Servlets often contain both business logic and presentation logic so it makes application difficult to understand.
- You would need JRE to be installed to run a servlet program.
- Difficult to map with the use of XML.

CHECK YOUR PROGRESS

- What do you understand by Tomcat?
- Give the advantages and disadvantages of Servlet.

9.8 SINGLE & MULITI-TIER APPLICATIONS

The Java Enterprise Edition (or simply J2EE) platform uses a distributed multi-tiered application model for enterprise applications. Application logic is divided into components according to function, and the various application components that make up a Java EE application are installed on different machines depending on the tier in the multi-tiered Java EE environment to which the application component belongs. Figure 9.2 shows two multi-tiered Java EE applications divided into the tiers described in the following list:

- Client-tier components run on the client machine.
- Web-tier components run on the Java EE server.
- Business-tier components run on the Java EE server.
- Enterprise information system (EIS)-tier software runs on the EIS server.

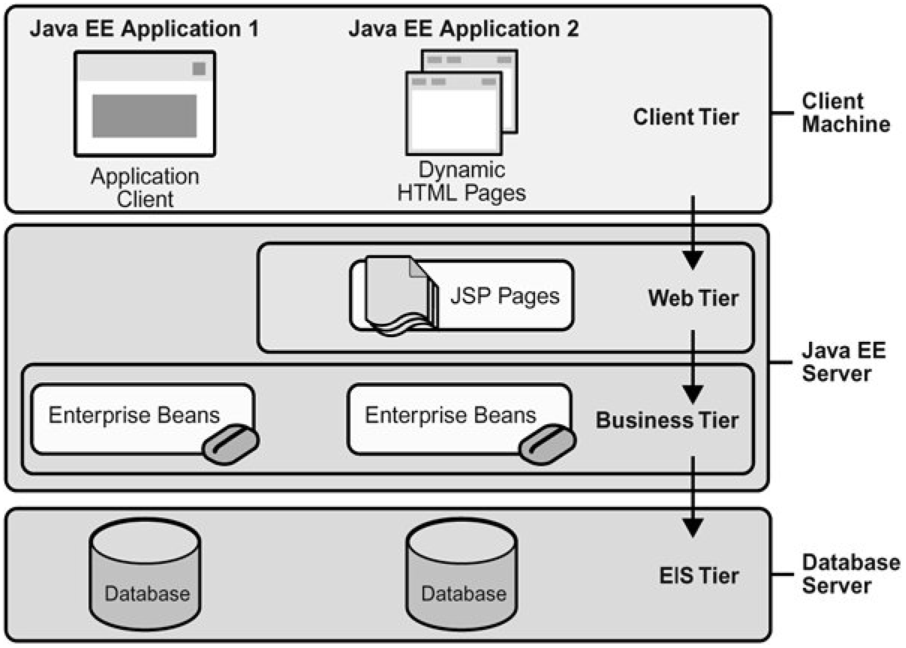


Figure 9.2: Multi-tiered Web Application

Although a Java EE application can consist of the three or four tiers shown in Figure 9.2, Java EE multi-tiered applications are generally considered to be *three-tiered applications* because they are distributed over three locations: client machines, the Java EE server machine, and the database or legacy machines at the back end. Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage.

9.9 SUMMARY

Java servlets are extensions to a Web server that allow Web content to be created dynamically in response to a client request. They are managed by a servlet engine, which loads and initializes them, passes them a number of requests for servicing, and then unloads them. Servlets have key advantages over other server-side programming environments:

- Better performance because they remain resident and can run in multiple threads simultaneously
- Simplicity because they require no client software installation other than a Web browser
- Session tracking
- Access to Java technology, including threading, networking, and database connectivity

Servlets operate in a fixed lifecycle, providing callback methods to a servlet engine for being initialized, handling requests, and terminating. The API provides two threading models: the default being a single instance running multiple threads, and the alternative single threaded model.

The principal classes and interfaces in the servlet API are:

- The Servlet interface, which prescribes the callback methods that must be implemented
- GenericServlet, a base class that implements the Servlet interface methods
- HttpServlet, an HTTP-specific subclass of GenericServlet
- ServletRequest, which encapsulates information about the client request
- ServletResponse, which provides access to an output stream for results to be returned to the client
- The ServletContext interface, which allows a group of servlets to interoperate with each other in a Web application

9.9 TERMINAL QUESTIONS

1. What do you understand by a web application?
2. Write the relation of Java technology with servlet.
3. Compare client and server side java programs.
4. Write the main features of servlet.
5. Explain the meaning and life cycle of a servlet.
6. Give advantages and disadvantages of a servlet technology.
7. Write a program to print the "hello world" using servlet.
8. What is the difference between HttpServlet and HttpRequest.
9. Write the name and role of important predeifined objects used in servlet.
10. What do you understand by multi-tiered application?

Unit-10 : Introduction to Java Server Pages

Structure

10.0 INTRODUCTION

Earlier, before this chapter, you learned how to generate dynamic Web pages using servlets. You probably have already noticed in some basic examples that most of the code in our servlets generated output that consisted of the HTML elements that composed the response to the client. Only a small portion of the code dealt with the business logic. Generating responses from servlets requires that Web application developers be familiar with Java. However, many people involved in Web application development, such as Web site designers, do not know Java.

It is difficult for people who are not Java programmers to implement, maintain and extend a Web application that consists of primarily of servlets. The solution to this problem is JavaServer Pages (JSP) an extension of servlet technology that separates the presentation from the business logic. This lets Java programmers and Web-site designers focus on their strengthswriting Java code and designing Web pages, respectively. JavaServer Pages simplify the delivery of dynamic Web content. They enable Web application programmers to create dynamic content by reusing predefined components and by interacting with components using server-side scripting.

JavaServer Page programmers can use special software components called JavaBeans and custom tag libraries that encapsulate complex, dynamic functionality. A JavaBean is a reusable component that follows certain conventions for class design that are discussed in the JavaBeans specification. Custom-tag libraries are a powerful feature of JSP that allows Java developers to hide complex code for database access and other useful services for dynamic Web pages in custom tags. Web sites use these custom tags like any other Web page element to take advantage of the more complex functionality hidden by the tag. Thus, Web-page designers who are not familiar with Java can enhance Web pages with powerful dynamic content and processing capabilities.

10.1 OBJECTIVES

In particular, this unit aims to achieve the following objectives:

- Fundamental concepts of JSP
- Where JSP lies
- Relation between servlet and JSP
- Introduction to Tomcat
- How to run first JSP program usinf Tomcat or other web server
- JSP objects
- A small application using JSP

10.2 INTRODUCTION TO JAVA SERVER PAGES (JSP)

It is a Java technology that allows software developers to create dynamically-generated web sites, with HTML, XML, or other document types, in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

The JSP syntax adds additional XML-like tags, called JSP actions, to be used to invoke built-in functionality. Additionally, the technology allows for the creation of JSP tag libraries that act as extensions to the standard HTML or XML tags. Tag libraries provide a platform independent way of extending the capabilities of a Web server.

As shown in the diagram basically JSP programs are converted in servlets. JSPs are compiled into Java servlets by a JSP compiler. Whenever a reuest comes from the client sied that is to say from the user It is handled by the container or web container eg Apache Tomcat. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, or it may generate byte code for the servlet directly. JSPs can also be interpreted on-the-fly, reducing the time taken to reload changes. JSP basically presents dynamic contents to the users and handles the presentation logic in different architectures like Model View Controller (MVC).

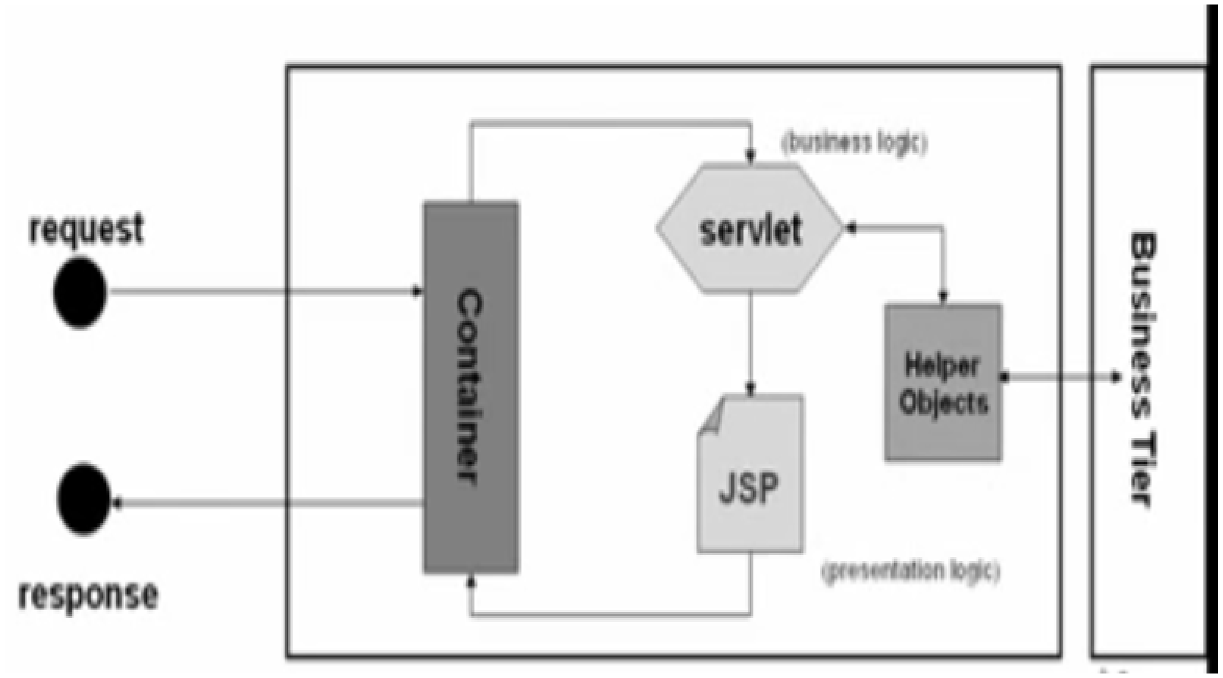


Figure 10.1: Architecture of JSP

10.3 HOW JSP WORKS

A JSP page exists in three forms:

- **JSP source code** this is the form the developer actually writes. It exists in a text file with an extension of .jsp, and consists of a mix of HTML template code, Java language statements, and JSP directives and actions that describe how to generate a Web page to service a particular request.
- **Java source code** The JSP container translates the JSP source code into the source code for an equivalent Java servlet as needed. This source code is typically saved in a work area and is often helpful for debugging.
- **Compiled Java class** Like any other Java class, the generated servlet code is compiled into byte codes in a .class file, ready to be loaded and executed.

The JSP container manages each of these forms of the JSP page automatically, based on the timestamps of each file. In response to an HTTP request, the container checks to see if the .jsp source file has been modified since the .java source was last compiled. If so, the container retranslates the JSP source into Java source and recompiles it. Figure 5-1 illustrates the process used by the JSP container.

When a request for the JSP page is made, the container first determines the name of the class corresponding to the .jsp file. If the class doesn't exist or if it's older than the .jsp file (meaning the JSP source has changed since it was last compiled), then the container creates Java source code for an equivalent servlet and compiles it. If an instance of the servlet isn't already running, the container loads the servlet class and creates an instance. Finally, the container dispatches a thread to handle the current HTTP request in the loaded instance.

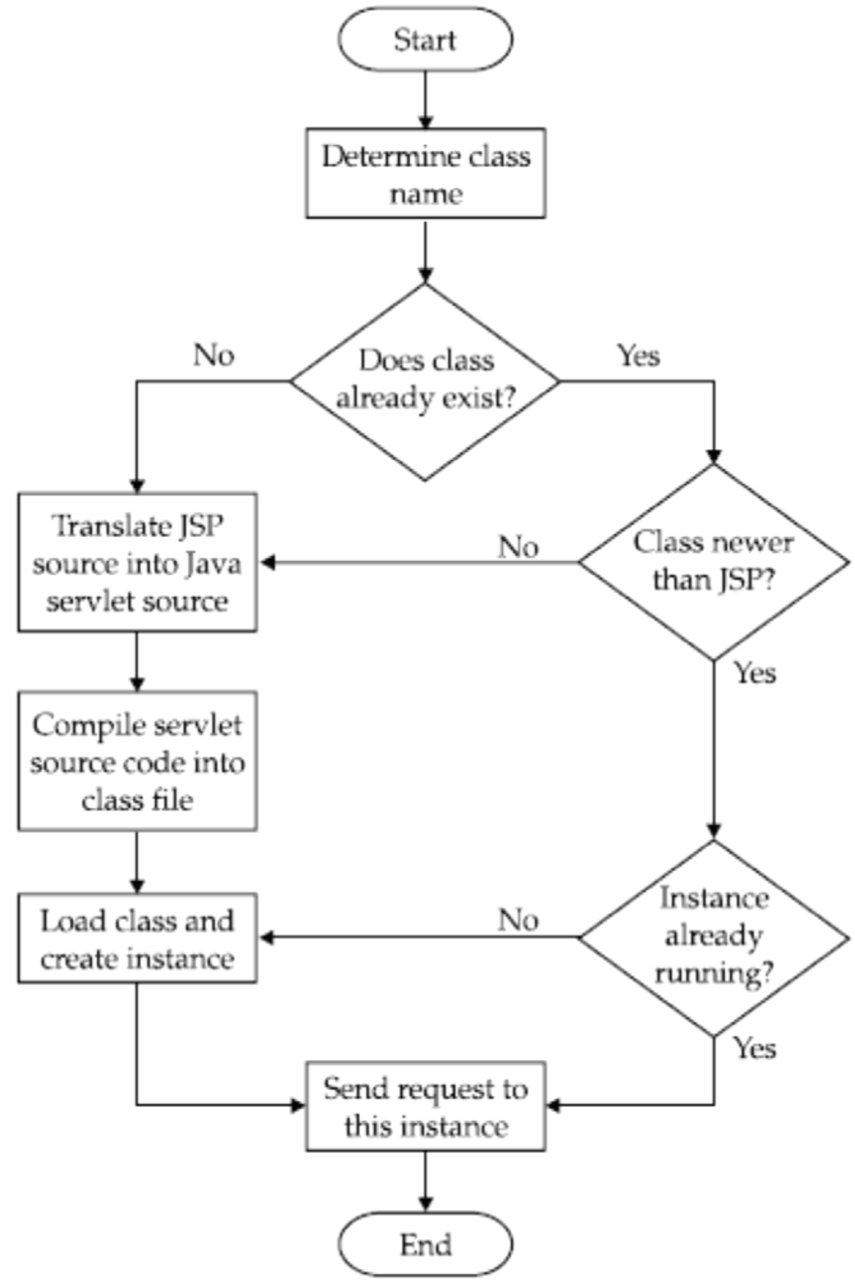

Start
Determine class name
Does class already exist?
No
Yes
Translate JSP source into Java servlet source
Class newer than JSP?
No
Yes
Compile servlet source code into class file
Load class and create instance
Instance already running?
No
Yes
Send request to this instance
End

Figure 10.2: Logic Used by a JSP Container to manage JSP translation

Example

```
<%@ page session="false"     %>

<%@ page import="java.io.*"  %>

<%@ page import="java.text.*" %>

<%@ page import="java.util.*" %>

<%@ page import="java.lang.*" %>
```

```
<%-- Give your comments here --%>
<%-- Prints a conversion table of miles per gallon to kilometers per liter --%>

<%!

private static final DecimalFormat FMT = new DecimalFormat("#0.00");

%>

<HTML>

<HEAD><TITLE>Fuel Efficiency Conversion Chart</TITLE></HEAD>

<BODY>

<H3>Fuel Efficiency Conversion Chart</H3>

<table border = 1 cellpadding = 3 cellspacing = 0>

<TR>

<TH>Kilometers per Liter</TH>

<TH>Miles per Gallon</TH>

</TR>

<%

for (double kpl = 5; kpl <= 20; kpl += 1.0) {

double mpg = kpl * 2.352146;

%>

<TR>

<TD><%= FMT.format(kpl)%></TD>

<TD><%= FMT.format(mpg)%></TD>

</TR>

<%

}

%>

</table>

</BODY>

</HTML>
```

10.4 JSP & SERVLETS

Architecturally, JSP may be viewed as a high-level abstraction of Servlets that is implemented as an extension of the Servlet 2.1 API. Both Servlets and JSPs were originally developed at Sun Microsystems. Starting with version 1.2 of the JSP specification, Java Server Pages have been developed under the Java Community Process.

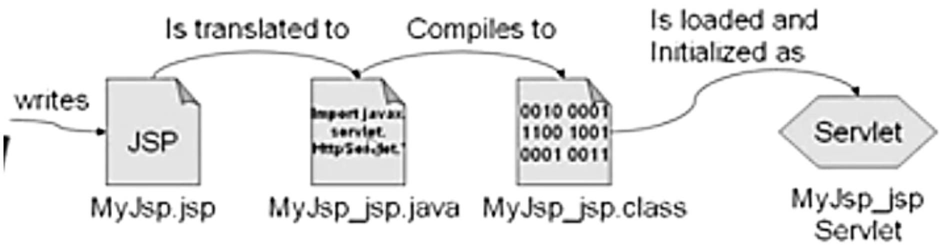


Figure 10.3: Conversion of JSP program into Servlet

When we compare both the technologies we find many differences but both handles dynamic data (insert, update, delete, modify) and run within a web container (e.g Apache Tomcat). Ultimately we can say that a JSP is a Servlet.

Table 10.1 : Comparison between JSP and Servlets

JSP	**Servlets**
Handles business logic	Handles presentation logic
Lifecycle methods init()- can be overridden service()- can be overridden destroy()- can be overridden	**Lifecycle methods** jspInit()- can be overridden _jspService()- cannot be overridden jspDestroy()- can be overridden

Html within java
 	
out.println("<html><body>");
 	
out.println("Time is"+new Date());
 	
out.println("</body></html>");
	Java within html
	
	<html><body>
	
	Time is<%= new Date());
	
	</body></html>");

Servlet (and JSP) offers the following benefits that are not necessarily available in other technologies:

- **Performance:** The performance of servlets is superior to CGI because there is no process creation for each client request. Instead, each request is handled by the servlet container process. After a servlet is finished processing a request, it stays resident in memory, waiting for another request.
- **Portability:** Similar to other Java technologies, servlet applications are portable. You can move them to other operating systems without serious hassles.
- **Rapid development cycle:** As a Java technology, servlets have access to the rich Java library, which helps speed up the development process.
- **Robustness:** Servlets are managed by the Java Virtual Machine. As such, you don't need to worry about memory leak or garbage collection, which helps you write robust applications.
- **Widespread acceptance:** Java is a widely accepted technology. This means that numerous vendors work on Java-based technologies. One of the advantages of this widespread acceptance is that you can easily find and purchase components that suit your needs, which saves precious development time.

10.5 JSP SYNTAX

A Java Server Page may be broken down into the following pieces:

- Static data such as HTML
- JSP directives such as the include directive
- JSP scripting elements and variables
- JSP actions
- Custom tags with correct library.

10.6 JSP DIRECTIVES

JSP directives control how the JSP compiler generates the servlet. Directives are instructions to the JSP container that describe what code should be generated. They have the general form:

```
<%@ directive-name [attribute="value" attribute="value" ...] %>
```

Zero or more spaces, tabs, and newline characters can be after the opening <%@ and before the ending %>, and one or more whitespace characters can be after the directive name and between attributes/value pairs. The only restriction is that the opening <%@ tag must be in the same physical file as the ending %> tag. The JSP 1.1 specification describes three standard directives available in all compliant JSP environments:

- page
- include
- taglib

Although the specification declares that no custom directives can be used in the JSP 1.1 environment, this leaves open the possibility that user-defined directives may be included in a later specification. The next three sections provide an overview of each of these directives.

10.6.1 The page Directive

The **page** directive is used to provide instructions to the container that pertain to the current JSP page. Yo may code page directives anywhere in your JSP page. By convention, page directives are coded at the to of the JSP page. Following is the basic syntax of page directive:

```
<%@ page attribute="value" %>
```

You can write XML equivalent of the above syntax as follows:

```
<jsp:directive.page attribute="value" />
```

The page directive has several attributes. Following is the list of attributes associated with page directive:

Table 10.2: Attributes of Directive

Attribute	**Purpose**
Buffer	Specifies a buffering model for the output stream.
autoFlush	Controls the behavior of the servlet output buffer.
contentType	Defines the character encoding scheme.
errorPage	Defines the URL of another JSP that reports on Java unchecked runtime exceptions.
isErrorPage	Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute.
Extends	Specifies a superclass that the generated servlet must extend
Import	Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes. Results in a Java **import** statement being inserted into the resulting file.
Info	Defines a string that can be accessed with the servlet's getServletInfo() method.
isThreadSafe	Defines the threading model for the generated servlet.
Language	Defines the programming language used in the JSP page.
Session	Specifies whether or not the JSP page participates in HTTP sessions
isELIgnored	Specifies whether or not EL expression within the JSP page will be ignored.
isScriptingEnabled	Determines if scripting elements are allowed for use.

Some commonly used attributes for the directives are explaines below.

import

Results in a Java **import** statement being inserted into the resulting file.

contentType

specifies the content that is generated. This should be used if HTML is not used or if the character set is not the default character set.

errorPage

Indicates the page that will be shown if an exception occurs while processing the HTTP request.

isErrorPage

If set to false if needed, it indicates that this is the error page. Default value is true.

isThreadSafe

Indicates if the resulting servlet is thread safe.

autoFlush

To auto flush the contents. A value of true, the default, indicates that the buffer should be flushed when it is full. A value of false, rarely used, indicates that an exception should be thrown when the buffer overflows. A value of false is illegal when also using buffer="none".

session

To maintain session. A value of true (the default) indicates that the predefined variable session (of type HttpSession) should be bound to the existing session if one exists, otherwise a new session should be created and bound to it. A value of false indicates that no sessions will be used, and attempts to access the variable session will result in errors at the time the JSP page is translated into a servlet.

buffer

To set Buffer Size. The default is 8k and it is advisable that you increase it.

language

Defines the scripting language used in scriptlets, expressions and declarations. Right now, the only possible value is "java".

extends

Defines the super class of the class this JSP will become. You won't use this unless you REALLY know what you're doing - it overrides the class hierarchy provided by the Container.

info

Defines a String that gets put into the translated page, just so that you can get it using the generated servlet's inherited getServletInfo() method.

pageEncoding

Defines the character encoding for the JSP. The default is "ISO-8859-1"(unless the contentType attribute already defines a character encoding, or the page uses XML document syntax).

EXAMPLE

```
<%@ page session="false"    %>
<%@ page import="java.io.*"  %>
<%@ page import="java.text.*" %>
<%@ page import="java.lang.*" %>

<html>
<head><title>BCA</title></head>
<body>Hello world</body>
</html>
```

10.6.2 The include Directive

The include directive informs the JSP compiler to include a complete file into the current file. It is as if the contents of the included file were pasted directly into the original file. This functionality is similar to the one provided by the C preprocessor. Included files generally have the extension ".jsp"

Example

```
<%@ include file="hello.jsp" %>
```

The include directive contrasts with the <jsp:include> action, which merges the output of another file at request time into the response output stream. Either element can be used to include standard headers and footers or other common text in JSP pages.

10.6.3 The include Directive

The taglib directive makes custom actions available in the current page through the use of a tag library. The syntax of the directive is:

```
<%@ taglib uri="tagLibraryURI" prefix="tagPrefix" %>
```

10.7 JSP IMPLICIT OBJECTS

The following JSP implicit objects are exposed by the JSP container and can be referenced by the programmer:

out

The JspWriter used to write the data to the response stream (output page).

page

The servlet itself.

pageContext

A PageContext instance that contains data associated with the whole page. A given HTML page may be passed among multiple JSPs.

request

The HttpServletRequest object that provides HTTP request information.

response

The HttpServletResponse object that can be used to send data back to the client.

session

The HttpSession object that can be used to track information about a user from one request to another.

config

Provides servlet configuration data.

application

Data shared by all JSPs and servlets in the application.

exception

Exceptions not caught by application code.

10.8 SCRIPTING ELEMENTS

There are *three* basic kinds of scripting elements that allow java code to be inserted directly into the servlet.

1. A ***declaration*** tag places a variable definition inside the body of the java servlet class. Static data members may be defined as well. Also inner classes should be defined here. Declaration tags also allow methods to be defined.

   <%! int xVariable = 1; %>

2. A ***scriptlet*** tag places the contained statements inside the _jspService() method of the java servlet class.

   <% int xVariable = 1; out.println(xVariable); %>

3. An *expression* tag places an expression to be evaluated inside the java servlet class. Expressions should not be terminated with a semi-colon.

   <%! int x=2; %>   this is declaration

   <%= x%>   this is expression

   This gives the output 2

10.9 JSP ACTIONS

JSP actions are XML tags that invoke built-in web server functionality. They are executed at runtime. Some are standard and some are custom (which are developed by Java developers). The following list contains the standard ones:

jsp:include:

Similar to a subroutine, the Java servlet temporarily hands the request and response off to the specified Java Server Page. Control will then return to the current JSP, once the other JSP has finished. Using this, JSP code will be shared between multiple other JSPs, rather than duplicated.

Example

```
<html>
<head></head>
<body>
<jsp:include page="mycommon.jsp" >
<jsp:param name="extraparam" value="myvalue" />
</jsp:include> name:<%=request.getParameter("extraparam")%
</body>
</html>
```

jsp:param

Can be used inside a jsp:include, jsp:forward or jsp:param block. Specifies a parameter that will be added to the request's current parameters.

jsp:forward

Used to hand off the request and response to another JSP or servlet. Control will never return to the current JSP.

Example

```
<jsp:forward page="subpage.jsp" >
<jsp:param name="forwardedFrom" value="this.jsp" />
</jsp:forward>
```

In this forwarding example, the request is forwarded to "subpage.jsp". The request handling does not return to this page.

jsp:plugin

Older versions of Netscape Navigator and Internet Explorer used different tags to embed an applet. This action generates the browser specific tag needed to include an applet.

jsp:fallback

The content to show if the browser does not support applets.

jsp:getProperty

Gets a property from the specified JavaBean.

jsp:setProperty

Sets a property in the specified JavaBean.

jsp:useBean

Creates or re-uses a JavaBean available to the JSP page.

10.10 SUMMARY

- The JSP development environment provides a means for generating HTML pages dynamically with server-side Java programming. The syntax allows most of the HTML to be coded directly, with sections marked off for Java code that controls the page generation. There is support for including other resources, both static and dynamic.
- JSP is the extension of servlets that provides a template for java code, html, and jsp syntax. It is very popular server side java technology. You may be able to author a JSP page without understanding the underlying API.
- Mastering the classes and interfaces in the javax.servlet.jsp package and understanding how JSP extends the servlet technology provides you with the knowledge to write more powerful and efficient code.
- You have learned the three types of JSP elements: directives, scripting elements, and action elements.
- You also have learned declarations, scripting elements, and action elements, and have been shown how to use all of them.

10.11 TERMINAL QUESTIONS

1. What do you understand by Java Server pages?
2. Write the relation between Servlet and JSP.
3. Give the name of JSP elements.
4. How we can write and run a JSP program?
5. Compare and contrast JSP and servlet.
6. What are JSP directives? Explain.
7. Explain the scripting elements using appropriate syntax.
8. Write a program using JSP to print a Fibonacci series for N number.
9. Write a program in JSP to reverse a number.

Note

Note