LOVELY PROFESSIONAL UNIVERSITY

WEB PROGRAMMING

Edited By

Mandeep Kaur

Printed by
EXCEL BOOKS PRIVATE LIMITED
A-45, Naraina, Phase-I,
New Delhi-110028
for
Lovely Professional University
Phagwara

SYLLABUS

Web Programming

Objectives: To impart the skills needed for web programming, web administration, and web site development. After studying this course student can develop; static web pages; dynamic web pages; data Processing on web pages.

S. No.	Description
1.	**Internet Fundamentals:** Introduction to Internet, Web browser, web page, website, homepage, hyperlinks, hypermedia, HTTP, WWW, Web server, Client server architecture model for web requests, URL.
2.	**Creating static web pages:** HTML document structure, singular and paired tags, test formatting, hyperlinks, adding images, audio and video, creating lists, tables, forms, frames, using multiple windows for web pages.
3.	**Cascading Style Sheets:** Style tag, DIV and SPAN, Internal and External stylesheets, Creating and using Classes, applying style on text and images.
4.	**Scripting Language:** Java Script programming, Data Types, Variables, Arrays, Operators. Loops, functions, Dialog boxes, String Manipulation functions, Using Timer in web page. Setting and Getting date object in a web page.
5.	**DOM Model:** Events handling through JavaScript, How to use forms in JavaScript.
6.	**ASP:** Introduction to asp, installing IIS, ASP variable, ASP operators, conditional, loops and case statements and arrays.
7.	**ASP Web Forms:** Introduction to CGI, Client side and server side scripting, building and processing web forms.
8.	**ASP Objects:** Response, Request, Server, Session, Application. Purpose of Global.asa file, #include, Recordset objects.
9.	**ASP Cookies and Caching:** Procedures, Cookies, ASP file system, send e-mail, Caching: page, data, fragment, output.
10.	**Database Connectivity:** Open and Close a connection, reading from the database, inserting, deleting and updating the database records Building Database Applications Using ActiveX Data Objects.

CONTENT

Unit 1:	Internet Fundamentals *Mandeep Kaur, Lovely Professional University*	1
Unit 2:	Creating Static Web Pages *Mandeep Kaur, Lovely Professional University*	34
Unit 3:	Text Formatting *Mandeep Kaur, Lovely Professional University*	70
Unit 4:	Cascading Style Sheets *Mandeep Kaur, Lovely Professional University*	88
Unit 5:	Scripting Language *Mandeep Kaur, Lovely Professional University*	131
Unit 6:	DOM Model *Mandeep Kaur, Lovely Professional University*	169
Unit 7:	Introduction to ASP (Active Server Pages) *Kumar Vishal, Lovely Professional University*	187
Unit 8;	ASP Web Forms *Kumar Vishal, Lovely Professional University*	204
Unit 9:	ASP Objects *Kumar Vishal, Lovely Professional University*	217
Unit 10:	Working with Response Object *Mandeep Kaur, Lovely Professional University*	241
Unit 11:	Using Request Objects *Mandeep Kaur, Lovely Professional University*	253
Unit 12:	Recordset Object *Mandeep Kaur, Lovely Professional University*	266
Unit 13:	ASP Cookies and Caching *Kumar Vishal, Lovely Professional University*	283
Unit 14:	Database Connectivity *Mandeep Kaur, Lovely Professional University*	296

Unit 1: Internet Fundamentals

Notes

CONTENTS

Objectives
Introduction
1.1 Evolution of Internet
1.1.1 Internet Usages
1.2 Web Browsers
1.2.1 How does a Web Browser Work?
1.3 Web Pages
1.3.1 Dynamic Web Pages
1.4 Website
1.5 Home Pages
1.6 Hyperlinks
1.7 Hypermedia
1.7.1 Hypermedia – Limitations, Problems
1.7.2 Hypermedia – Basic Hypermedia Model
1.8 HTTP (Hypertext Transfer Protocol)
1.8.1 Understanding Hypertext Transport Protocol (HTTP)
1.8.2 Hypertext: The Motion of the Web
1.8.3 Understanding Hypertext Markup Language (HTML)
1.9 WWW
1.9.1 Terminology
1.9.2 Internet and www Milestones
1.10 Web Server
1.11 Client/Server Architecture Model for Web Request
1.12 URL
1.13 Summary
1.14 Keywords
1.15 Review Questions
1.16 Further Readings

Objectives

After studying this unit, you will be able to:

- Scan the introduction to internet
- Describe Web browsers, web page, website and home page

Notes

- Demonstrate the hyperlinks and hypermedia
- Recognize HTTP and WWW
- Explain Web Server
- Scan the client server architecture model

Introduction

"The Internet" is probably one of the most overused technical term in common man's lingo today. Though everyone seems to have some ideas and opinions about the Internet yet not many can define it precisely. There is no single, generally agreed upon answer to the question– "What is the Internet?" - because the Internet is a different thing to different people. Consider the following few expressions in this context.

- Internet, the mega network connecting millions of people, is a tremendous phenomenon. The web, an offshoot of this global resource has revolutionized the way information is passed on.
- It is the name given for a vast, worldwide system consisting of people, information, and computers.
- It is a network of networks.
- It is an unlimited commercial opportunity.
- It is a set of computers talking over fibre-optics, phone lines, satellite links, and other media.
- It is cyber space where data surfing can be done.
- It is an ocean of information.
- It is a gold mine of professionals from all fields sharing information about their work.
- It is a world wide interconnected system of thousands of computer networks, each network in turn linking thousands of computers together.

In short, we can say that the Internet is a vast collection of globally available information, which can be accessed electronically – information, which is of practical use for business, research, study, and technical purposes. It is a means for electronic commerce – marketing, buying, services, economic, and financial data research. It is a collection of hundreds of libraries and archives that will open to your fingertips. It is also a vast store of information relating to your hobbies, travel, health, entertainment, games, software, etc.

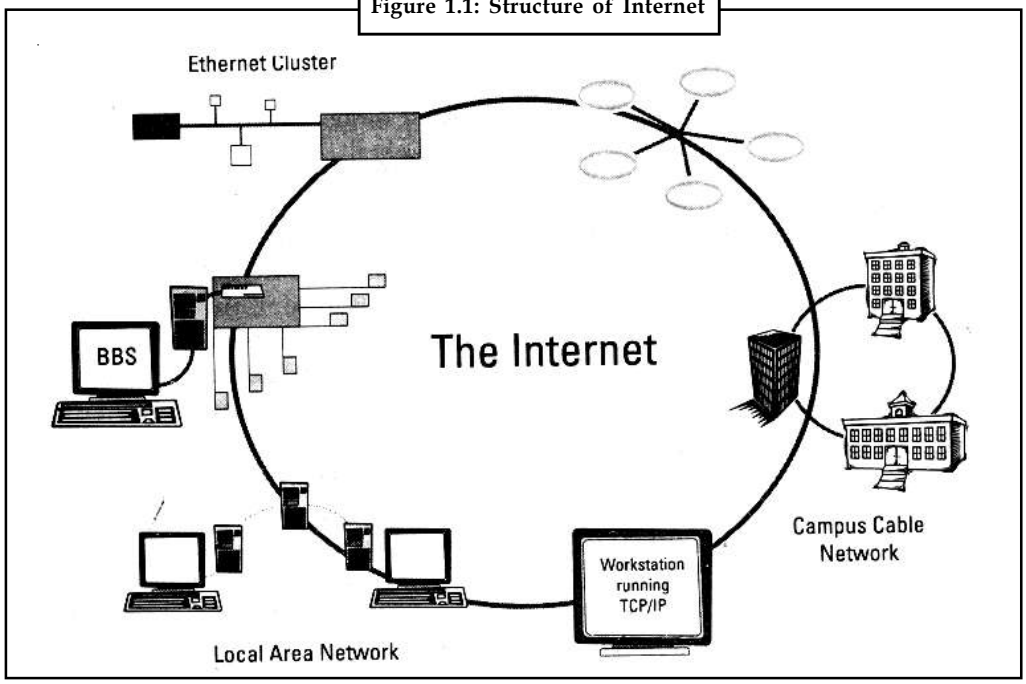
Today, the information can be in the form of Text, Images, Animation, Sound, Video, etc., tomorrow it would probably be in the form of smell, touch, taste or some energized form. If information can be put on computers, it can be digitized and can be made available on the Internet. The only catch is - How fast? Even the future may not be able to tell.

To be technically correct, we can say that the Internet is "an ever growing wide area network of millions of computers and computer networks across the globe, which can exchange information through standard rules (protocols)". The same is illustrated in the figure 1.1.

Each computer participating in the Internet has a unique address. Information is divided into packets, which may travel through different paths to the destination address where it is recombined into its original form.

Notes

Figure 1.1: Structure of Internet



A few computers can be interconnected to share resources and exchange information using a number of network technologies. Further, such networks can be interconnected to form even larger networks. The Internet is a similar network that connects millions of networks worldwide.

Notes Note that we use "The Internet", with definite article "The" and not just "Internet". This is because though there are many networks existing today the one we call "Internet" is a single network consisting of all the participating networks.

The first business computer appeared in 1960 and since then information technology has changed the way business or commerce is conducted across the globe. Companies are re-orienting themselves in the present competitive era where product life-cycles are coming down everyday. Globalization of markets is taking place with formation of trade blocks across the globe, and world moving towards a global village. Complex issues of cultural divide within even the same trade block forces the businesses to learn more about the customer, if possible to an individual level.

From mainframes to accounting systems, the Personal Computer (PC) revolution, local area networks, electronic data interchange, client/server design, and enterprise resource planning have all had a hand in shaping today's business organization. The past few years have been Internet years, however, when companies worldwide have embraced a change without equal. It is a change that promises to have more impact and be more lasting than anything we have seen to date. Technology is setting the pace for how a company does business, how it launches new products and enters a new market, how it deals with suppliers, and how it communicates with customers and others in the new marketplace.

We are now living in the ICE (Information Communication of Entertainment) age which reflects synergetic combination of Information, Communication and Entertainment. The primary technology for this transformation is Internet-the global data network. The customer has become more important with the advent of Internet and Worldwide Web (WWW). These customers fall in the high-income groups and are more demanding on quality and price. This makes the business very unconventional and highly unpredictable.

Notes

1.1 Evolution of Internet

How old is the Internet? Here is a short historical background. The concept, which gave birth to the Internet, as it exists today, started with a project called the ARPANET (around late 1960s) which was sponsored by the United States Department of Defence, Advanced Research Project Agency (ARPA). The Department of Defence (DOD) was interested in building a network that could maintain itself under adverse conditions. The original idea was to build a network capable of carrying military and government information during a "nuclear event". Thus, the predecessor of Internet, as often called ARPANET, came into being.

Till late 1970s, ARPANET operated to provide connection facility to around a dozen of research sites. Later in 1982, ARPA established the Transmission Control Protocol (TCP) and Internet Protocol (IP), as the protocol suite, commonly known as TCP/IP, which is a connection protocol between sites, which is still in use today and is the primary method of connecting to the Internet.

This technique of information access, which started as a method of sharing files amongst researchers was slowly adopted by a wider clientele and after 1982 the network started expanding faster like the big bang.

The credit remains with the National Science Foundation (NSF), USA, whose network of strategically located supercomputers (NSFnet) allowed people to share information from home or their institutions. The other networks which developed in parallel, like BITNET of IBM, X.25 based in Europe and UUCP based in Bell Labs got slowly absorbed or got connected to Internet, in order to provide a unified information sharing medium. The network although started in US, slowly started reaching other countries by connecting the networks of other countries.

Internet in India started on an auspicious day, 15th of August, 1995. This in a way marks the beginning of free information flow from every nook and corner of the world and thus could well be called an "Independence day" for Information Age in the country.

Did u know? **What is GIAS?**

Videsh Sanchar Nigam Limited (VSNL) started a service called GIAS (Gateway Internet Access Service) to allow the Indian users a bite of the Internet.

1.1.1 Internet Usages

The Internet is an important tool for practically everybody. The applications are endless limited only by our imagination. Whatever information is required, it is either already available on the Internet or it is soon going to be available. Here are some interesting application areas:

- Electronic mail, which was until recently considered only an internal mechanism of an enterprise, is quickly becoming the most widely used application on the Internet. The most common of the communication methods used by people on the Internet is the private letter, written by one individual to another (on any subject and in any language), and sent between any two connected Internet sites or through an Internet E-Mail gateway to or from a service which provides an Internet gateway.
- The ability to exchange visual information in readable and reusable formats — such as charts, figures, tables, images, databases, software code — opens up possibilities for collaboration at the global as well as local levels. With the trend specialization, the ability not only to communicate but also to actually work with colleagues in the same field scattered all over the world makes long distance collaboration feasible.
- The resources for on-line research are multiplying at an astounding rate. Searchable databases library holdings, alerting services, pre-prints, and other information systems are all changing

Notes

the way research is done. And it is not only the research community that is responsible for this. Library shelves are overflowing with journals and proceeding, and with acquisitions budgets receiving deep cuts, a likely scenario for the future is one in which libraries archive electronically, share holdings, and become information clearing houses instead of closets.

- Another very important application of Internet is multimedia. Live music concerts, radio broadcasts, live or recorded television shows, interactive audio and webphone, and video conferencing are no more a dream on Internet, even for a desktop PC user.

Internet provides a variety of information to everybody ranging from entertainment to serious business application to areas of daily life such as:

- Magazines and newspapers
- Household shopping items
- Ordering novelties from anywhere in the world
- Radio and TV broadcast schedules and sometimes the broadcast itself
- Tour and travel plan guides and bookings, etc.
- Health consultation
- Tips for doing various things
- Talking to friends and relatives in any part of the globe
- Games of various kinds
- Language interpreter
- On-line education course material, examination conduction, advertising on popular information sites, making payments on the net and getting an item, net banking.

Self Assessment

Fill in the blanks:

1. A few computers can be interconnected to share resources and exchange information using a number of ……
2. The original idea was to build a network capable of carrying military and government information during a "……".
3. …… databases library holdings, alerting services, pre-prints, and other information systems are all changing the way research is done.

1.2 Web Browsers

A web browser is the software program used to access the World Wide Web that is the graphical portion of the Internet. The first browser, called NCSA Mosaic, was developed at the National Centre for Supercomputing Applications lab at Illinois in the early 1990s. The easy-to-use point-and-click interface fuelled the growth of the Web. The web browser software program facilitates a user to display and interact with text, images, videos, music and other information typically located on a web page at a website on the www or a LAN. Text and images on a web page normally designed to provide hyperlinks to other web pages at the same or different website. Thus web browser enables point to point click to reach directly to the targetted web pages.

Some of the popular web browsers are Internet Explorer, Mozilla Firefox, Safari, AOL Explorer, etc. Web browsers belong to HTTP user agent category. Browsers also provide in accessing information provided by web servers in private networks or content in file systems.

Notes

Web browsers talk with web servers using HTTP (hypertext transfer protocol) to retrieve web pages. HTTP enables web browsers to provide information to web servers to retrieve web pages from them. Different web pages have a URL address starting with http:// for HTTP access. There may different other URL types and their corresponding protocols and most of the browsers supports them. FTP (file transfer protocol) is one of the examples of such types. Other examples are rtsp: for RTSP (real-time streaming protocol) and https: for HTTPS (an SSL encrypted version of HTTP).

The file format for a Web page is normally HTML (hyper-text markup language) and is identified in the HTTP protocol with a MIME content type. Most of the browsers support different formats such as the JPEG, PNG and GIF image formats including HTML. The combination of HTTP content type and URL protocol specification enables designers to embed images, animations, video, sound and streaming media into a web page or to make them accessible through the web page.

Like much of the internet, the World Wide Web operates on a client/server model. You run a Web client on your computer — called a Web browser — such as Netscape Communicator or Microsoft's Internet Explorer. That client contacts a Web server and requests information or resources. The Web server locates and then sends the information to the Web browser, which displays the results. When Web browsers contact servers, they are asking to view pages built with Hypertext Markup Language (HTML). They interpret those pages and display them on your computer. They display application programs, animations, and similar material created with programming languages such as Java and ActiveX, and scripting languages such as JavaScript. Sometimes, home pages contain links to files that the Web browser cannot play or display, such as sound or animation files. In that case, you will need a plug-in or a helper application.

Task You configure your Web browser to use the helper application or plug-in whenever it encounters a sound or animation file that the browser cannot run or play. Analyze this statement practically.

Over the years, Web browsers have become increasingly sophisticated. They have become full-blown software suites that can do everything from video-conferencing to letting you create and publish HTML pages. They have also begun to blur the line between your local computer and the Internet-in essence, they can make your computer and the Internet function as a single system. To facilitate the usage of computers over Internet, Microsoft has integrated web browsing and the Internet directly into the operating system. For example, with Internet Explorer 4.0 and above, and with Windows 98, the Windows desktop can be HTML based. This means Web links can be directly embedded into the desktop.

So, you can have links to your favourite Web pages right on the desktop. And even applications such as word processors now have Web capabilities bust into them — such as being able to browse the Web, or build home pages. Even more significantly, using technology that Microsoft calls Active Desktop, internet based desktop components can live on the desktop. These components can be things such as stock tickers, which deliver live Web content directly to the desktop. You don't need to fire up your Web browser to get the information; it's delivered straight to your Windows desktop without your having to do anything. Both Microsoft and Netscape have also built entire suites of software around their browsers. Netscape for example, calls its suite Netscape Communicator. Communicator includes modules for reading newsgroups; for reading, sending and managing internet mail; for audio conferencing; for collaborative work on whiteboard applications in which people can view and mark up the same documents simultaneously; and more. These enhancements will help user in an era of collaborative computing and communication.

- It allows you to enter the address of the site you want to jump to (called a URL-or uniform resource locator), or to jump there by clicking hotspots, high lighted words, buttons, pictures, or icons called hyperlinks on your screen.

Notes

- It formats Web documents for display on your screen.
- It allows you to back up and go forward through pages you have already visited.
- It allows you to copy text from the screen and paste it into a word processing program.
- It allows you to print the document you see on the screen.
- It makes it possible to transfer files-text, graphics, movies, animations, sounds, and programs – from other computers to your computer (called downloading).
- It allows you to send and receive e-mail and other Internet services such as ftp (file transfer protocol), gopher, and Usenet news groups.

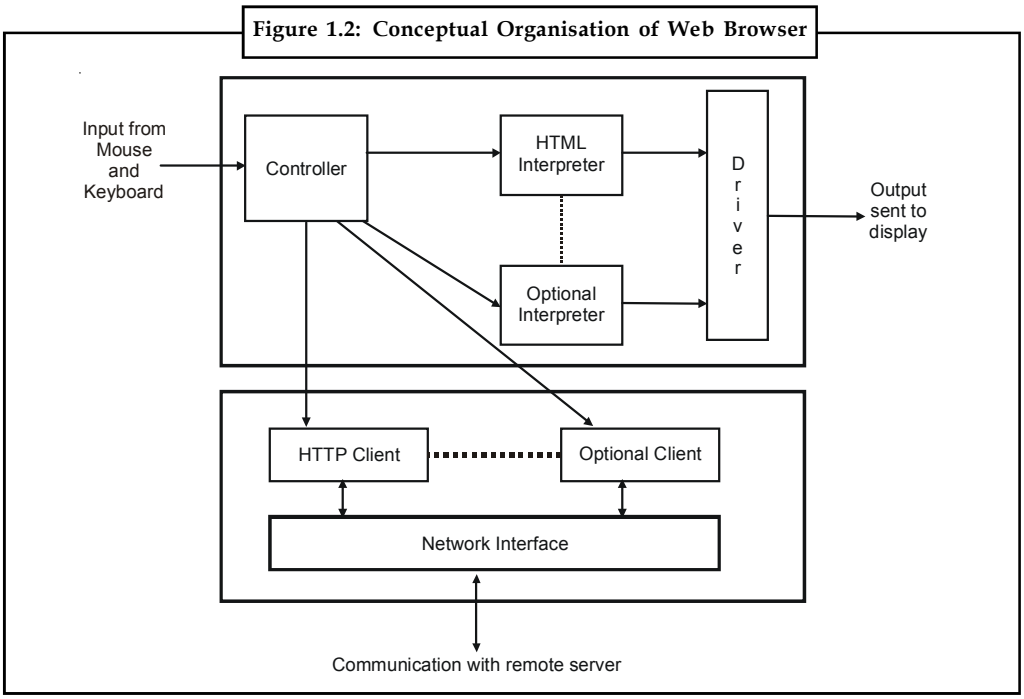
In it's basic form a Web browser is an application that uses information stored at locations around the World Wide Web, in order to perform a local task. To get the desired information from the Web, a browser uses the client-server paradigm. When given the URL of a document, the browser becomes a client that contacts a Web Server on the computer specified in the URL, to request the document. When a browser interacts with a Web server, the two programs follow the Hyper Text Transport Protocol (HTTP). HTTP allows a browser to request a specific item, which the server then returns.

Caution To ensure that browser and server can inter-operate unambiguously, HTTP defines the exact format of request sent from a browser to a server as well as the format of the servers reply.

While a Web server performs a straightforward task of waiting for a request and sending reply, the browser handles most of the details of document access and display. To be able to perform this tasks the Web browser contains some software components that interact to provide the illusion of seamless service.

The conceptual organization of a browser is depicted in Figure 1.2.

Figure 1.2: Conceptual Organisation of Web Browser



Notes

A browser consists of a set of clients, a set of interpreters and a controller that mange them. The controller forms the central piece of the browser. It interprets both mouse clicks and keyboard inputs, and calls other components to perform operations specified by the user.

Example: When you press a link, the controller will call a client to fetch the requested document from the remote server, and an interpreter to display the document for the user.

Besides an HTTP client and an HTML interpreter a browser can contain components that enable the browser to perform additional tasks. Many browsers include an FTP client that is used to access the file transfer service and an e-mail client that enable the browser to send and receive mail.

The user doesn't invoke such services explicitly; instead, the service is invoked automatically, when it is needed to perform a task. For Example, file transfer can be associated with a selectable object on the screen, when a user selects the item the browser uses the FTP client to obtain a copy of the file.

How to Access the World Wide Web — Web Browsers

There are two types of browsers:

1. ***Graphical:*** Text, images, audio, and video are retrievable through a graphical software program such as Netscape Navigator and Internet Explorer. These browsers are available for both Windows-based and Macintosh computers. Navigation is accomplished by pointing and clicking with a mouse on highlighted words and graphics. The current version of Navigator is contained within a suite of programs called Netscape Communicator. You can install a graphical browser such as Netscape Navigator in your Windows-based or Macintosh machine. Navigator is available for downloading on the Netscape home page: http://home.netscape.com. To use the program to access the Web, you need an Ethernet connection or a dialup connection known as a SLPP or PPP. The latter may be obtained from an Internet Service Provider. For more information, see How to connect to the Internet.
2. ***Text:*** Lynx is a browser that provides access to the Web in text-only mode. Navigation is accomplished by highlighting emphasized words in the screen with the arrow up and down keys, and then pressing the forward arrow (or Enter) key to follow the link. This browser is available through your personal IBM, VAX, or UNIX account on campus.

Extending the Browser — Helper Applications and Plug-Ins

Software programs may be configured to a Web browser in order to enhance its capabilities. When the browser encounters a sound, image or video file, it hands off the data to other programs, called helper applications, to run or display the file. Working in conjunction with helper applications, browsers can offer a seamless multimedia experience. Many helper applications are available for the free.

File formats requiring helper applications are known as MIME types. MIME stands for the Multimedia Internet Mail Extension, was originally developed to help e-mail software handle a variety of binary (non-ASCII) file attachments. The use of MIME has expanded to the Web. For example, the basic MIME type handled by Web browsers is text/HTML associated with the file extension.HTML.

A common helper application utilized on the Web is the Adobe Acrobat Reader. The Acrobat Reader allows you to view documents created in Adobe's Portable Document Format. These documents are the MIME type application/pdf and are associated with the file extension .pdf. When the Acrobat Reader has been configured to your browser, the program will open and display the file requested when you click on a hyperlinked file name with the suffix .pdf.

Notes

Plug-ins are software programs that extend the capabilities of a Web browser in a specific way, such as the ability to play audio files or view video movies from within Navigator. Web browsers are often standardized with a small suite of plug-ins. Additional plug-ins may be obtained at the browser's Web site, at special download sites on the Web, or from the home pages of the companies that created the programs. The number of available plug-ins is increasing rapidly.

Example: For example, nearly 200 plug-ins are available for downloading at the Netscape site.

Once a plug-in is configured to your browser, it will automatically launch when you choose to access a file type that it uses.

Netscape Communicator can be downloaded with a variety of helper applications and plug-ins configured to the browser, including:

- Cosmo Player to view 3D sites created with Virtual Reality Modelling Language (VRML) (file suffixes .wrl, .wrz)
- Netscape Media Player for streaming audio metafiles (file suffix .lam)
- Live Audio for sound files (file suffixes .au, .aiff, .wav, .midi, .la, .lma) QuickTime Player for video (file suffix .mov)
- NPAI32 Dynamic Link Library for video in Windows (file suffix .avi)

Beyond Plug-Ins — Active X

ActiveX is a technology developed by Microsoft, which may make plug-ins less necessary. ActiveX offers the opportunity to embed animated objects, data, and computer code on Web pages. A web browser supporting ActiveX can render most items encountered on a Web page. For example, Active X allows users to view three-dimensional VRML worlds in a Web browser without the use of a VRML plug-in. As another example of the power of ActiveX, this technology can allow you to view and edit PowerPoint presentations directly within your Web browser. ActiveX is supported by the Internet Explorer and Netscape Navigator 4.x browsers.

The Experience of The Web

Today's World Wide Web(WWW) presents an ever-diversified experience of multimedia, programming languages, and real-time communication. There is no question that, it is a challenge to keep up with the rapid pace of developments. The following presents a brief description of some of the more important trends to watch.

Multimedia

The Web has become a broadcast medium. It is possible to listen to audio and video over the Web both prerecorded and live. For example, you can visit the sites of various news organisations and view the same videos shown on the nightly television news. Several plug-ins are available for viewing these videos. For example, Apple's Quick Time Player downloads files with the .mov extension and displays these as "movies" in a small window on your computer screen. Quick Time files can be quite large, and it may take patience to wait for the entire movie to download into your computer before you can view it.

The problem of slow downloads times has been answered by a revolutionary development in multimedia capability: streaming data. In this case, audio or video files are played as they are downloading, or streaming, into your computer. Only a small wait, called buffering, is necessary

Notes

before the file begins to play. The RealPlayer plug-in plays streaming audio and video files. Extensive files such as interviews, speeches and hearings work very well with the RealPlayer. The RealPlayer is also ideal for the broadcast of real-time events. These may include press conferences, live radio and television broadcasts, concerts, etc. A list of sites that make use of the RealPlayer is available at http://www.albany.edu/library/internet/net_info/realaudio.html. The Windows Media Player is another streaming media player.

Notes A list of sites that make use of this player is available at http://wmg.netcastnetwork.com/. Many sites offer the option to use one player or the other.

Shockwave presents another multimedia experience. Shockwave allows for the creation and implementation of an entire multimedia display combining graphics, animation and sound.

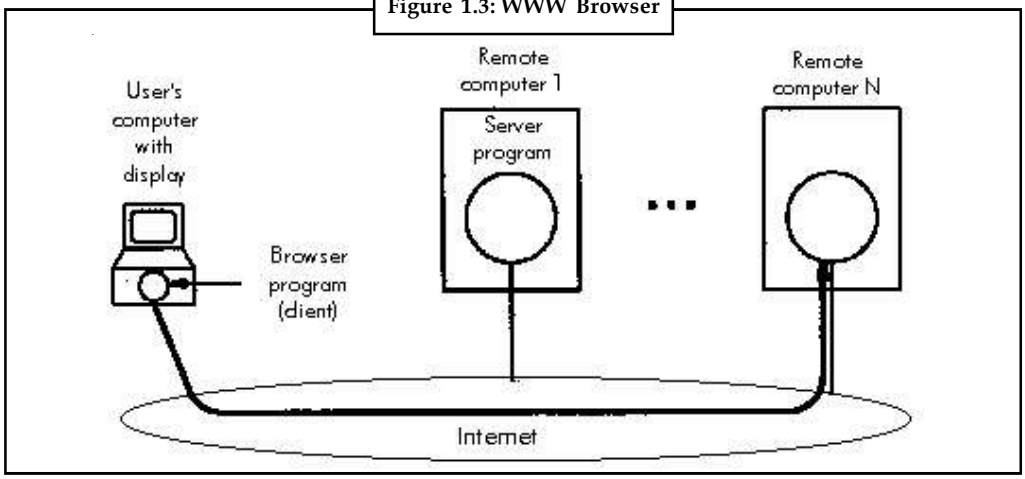
Sound files, including music, may also be heard on the Web. It is not uncommon to visit a Web page and hear background music. Sound files are also available for downloading independent of Web page visits. Sound files of many types are supported by the Web with the appropriate helper applications.

Live cams are another aspect of the multimedia experience available on the Web. Live cams are video cameras that send their data in real time to a Web server. These cams may appear in all kinds of locations, both serious and whimsical: an office, on top of a building, a scenic locale, a special event, and so on.

1.2.1 How does a Web Browser Work?

Web browsers consist of software that runs on your computer and displays home pages on the Web. There are clients for PC, Macintosh and UNIX computers. A Web browser displays information on your computer, by interpreting the Hypertext Markup language (HTML) that is used to build home pages on the Web. Home pages usually display graphics, sound and multimedia files as well as links to other pages, files that can be downloaded and other Internet resources.

Figure 1.3: WWW Browser



The coding in the HTML files tells your browser how to display the text, graphics, links, and multimedia files on the home page. The HTML file that your browser loads to display the home page does not actually have the graphics, sound, multimedia files, and other resources on it. Instead, it contains HTML references to those graphics and files. Your browser uses those references to find the files on the server and then display them on the home page. The Web

Notes

browser also interprets HTML tags as links to other Web sites or to other Web resources – these include graphics, multimedia files, newsgroups, or files which can be downloaded. Depending on the kind of link, it will perform different actions.

Example: If the HTML code specifies the link as another home page, the browser will retrieve the Uniform Resource Locator (URL) specified in the HTML file when the user clicks on the underlined link on the page. If the HTML code specifies a file to be downloaded, the browser will download the file to your computer.

The Server Side

The Web follows a client-server model. The server is called Web Server and the clients Browsers. The user usually runs a client program (or web browser) that can communicate with a server program (or web server) on a (remote) host computer. To access the information, the client sends the users' request to the server. The server handles the request and sends the response to the user.

A Web Server is the most visible part of your Internet. The Web server hosts Web pages and these pages generally represent the organization and its business. These Web Pages are files in a specialized format known as Hyper Text Mark-up Language (HTML). A Web Server communicates with browsers using the HTTP (Hyper Text Transfer Protocol), which runs over TCP/IP.

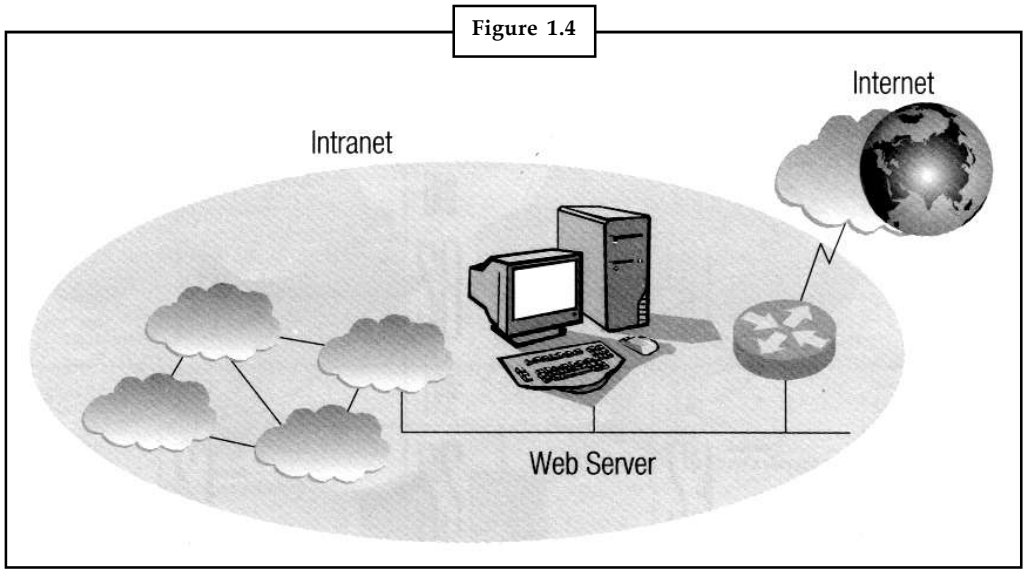
HTML files are text files with special tags. HTML files do not contain proprietary custom symbols or formatting characters. All formatting is specified with special combinations of ASCII text characters. HTML files can include text or graphic links, which users can click on to move to another location in the same file or to another file on any web server in the world.

Implementation and maintenance of Web server are very critical functions. These can range from personal Web Servers to Heavy-duty freeware, shareware to commercial ones.

Did u know? On which factor the choice of Web Server depends?

The choice of Web Servers would depend on the volume of the Web pages that you expect to put up and the number of clients accessing them.

Figure 1.4



Notes

Some of the basic features your Web server should have:

- ***HTTP 1.1 Compliance:*** The benefits include persistent connection, pipelining, and caching directives for performance, host headers for multiple Web sites.
- ***Browser Neutral Host Header Support:*** The Web server should be able to host multiple Web sites using a single IP address. This will also help Web browsers that do not support host headers.
- ***Discussion Groups:*** The Web server should include support for creating local discussion groups for a single server using the NNTP protocol.
- ***Mail enabled administrators*** should be able to receive mail driven by events on the Web server.

Other features of Web server include:

- Enhanced integrated set-up and administration such as Web based administration, Website operators, per Website bandwidth throttling, and configuration replication.
- Reliable Web application development and deployment which includes transactional active server pages, crash protection, crash recovery, integrated fail over clustering support, component load/unload.
- Standards based data access like database connectivity.
- Integrated authentication and security with support for integrated certificate server, domain blocking.

Rich content management and control like integrated indexing and searching, content expiration, document footers, custom errors, custom headers, one-to-one content replication, redirects, complete and comprehensive documentation.

The Client Side

While the server side of the web stores and servers the in-coming requests, the requests are made to the server by a client software (machine). The client side runs on a local machine and may have some processing capabilities. The Internet browsers are one class of such client side software, e-mail clients are another.

Task Give the answers in one line.

1. What is web browser?
2. Define Domain name system.

Self Assessment

Fill in the blanks:

4. Text and images on a web page normally designed to provide …… to other web pages at the same or different website.
5. Browsers also provide in accessing information provided by web servers in …… networks or content in file systems.

Notes

6. Web browsers consist of software that runs on your computer and displays …… on the Web.
7. The Web server should include support for creating local discussion groups for a single server using the ……. protocol.

1.3 Web Pages

The Internet has evolved as a power tool, user friendly and most commercially popular technology. Anyone with Personal Computer (PC) connected to the Internet, a browser (a program designed to search for and bring in internet resources) and few plugs-ins (specialized program) can surf the unique representation of an enterprise products or services on the Internet.

The worldwide web is also known as the 'www' or the 'web'. It is an architectural framework of information system fully implemented in 1994 on the Internet. It contains millions of electronic documents called Web Pages. A web page contains text and graphics (drawings) which are linked to related information. The name 'web' is based on the fact that the text, pictures, animation, sound and information that make up a document may come from anywhere in the world. Thus, a single document can be perceived to stretch 'web like' throughout the world.

The Web is not internet. At times, people confuse the two terms that are related but not identical in meaning. The internet evolved from the military ARPANET in 1960s with the purpose of' creating a network that would continue to work as a whole, when parts of it collapse'. The Internet means a network infrastructure that is built on certain standards, which are followed by all participants to connect to each other. The Internet Protocol (IP) defines how the flow of information is organised. But it does not specify the types of information or services to be exchanged. World Wide Web lays down the specifications of information and services to be exchanged with its Hyper-Text Transfer Protocol (HTTP). Thus, the web offers the exchange of documents via HTTP.

Besides the www, there are other protocols that enable people to communicate via e-mail (POP3, SMTP, IMAP), chat online (IRC) or participate in newsgroups (NNTP). The www is thus one of the numerous services offered on the internet. It does not specify if a certain web page is available in the internet intranet or extranet. The World Wide Web provides a simple-to-use web interface that allows people with very little knowledge in computing to access web services all over the Internet. The web services include content products and services, which can be viewed or ordered through the web browser. It may be noted that the web browser allows customers to self-service themselves over the web.

Notes Note that when you are on the Web, you are on the Internet but not the other way round.

Example: Those sending e-mail are not on the Web, unless they are sending e-mail via a Web browser.

The beginning of the Internet dates back to the late 1960s, when the Advanced Research Projects Agency (ARPA) of the Department of Defense (DOD) formed ARPANET. The agency was established in the late 1950s to develop information technologies to help the United States to counter the Soviet launch of Sputnik. Consequently, the early ARPANET consisted primarily of research universities and military contractors with computers linked by telephone lines leased from AT&T. The chronological development of the ARPANET and the milestones leading to the Internet reflect a history of top-down government sponsored initiatives.

Notes

Task Answers the following questions:

1. Distinguish between Internet and world wide web.
2. What is the historical base for the development of internet?

1.3.1 Dynamic Web Pages

The popularity of the static, billboard style, Web page is declining. Dynamic pages can customize the response on the server to offer personalization based on cookies and information, it can get from the visitor. Web server content developers are creating dynamic Web pages with data from the databases and other data sources, such as real-time stock market data feeds. Web technology is being used to create new client/server applications because of the ability to dynamically create Web pages on the fly.

Many Web servers do not have database or other data access mechanisms built-in. They rely on the Common Gateway Interface (CGI). Dynamic web pages are typically written in various scripting languages or technologies such as ASP, PHP, Perl or JSP.

Advantages

- Offers highly personalized and customized visitor options.
- Database access improves the personalized experience (as opposed to using just client side cookies).
- Scripts can read in data sources and display it differently depending on how it is run.
- Can create the illusion of being updated regularly using time and date sensitive routines (or even randomizers) to display pre-written text.

Disadvantages

- Personalized pages are not very cache friendly.
- Requires a basic minimum knowledge of the language being used.
- Scripts need more consideration, when uploading and installing, particularly to *nix servers.

Self Assessment

Fill in the blanks:

8. Many ……. do not have database or other data access mechanisms built-in.
9. The web services include content products and services, which can be viewed or ordered through the …….

1.4 Website

A website is a collection of web pages (documents that are accessed through the Internet), such as the one you're looking at now. A web page is what you see on the screen when you type in a web address, click on a link, or put a query in a search engine. A web page can contain any type of information, and can include text, color, graphics, animation and sound.

Notes

When someone gives you their web address, it generally takes you to their website's home page, which should introduce you to what that site offers in terms of information or other services. From the home page, you can click on links to reach other sections of the site.

Notes A website can consist of one page, or of tens of thousands of pages, depending on what the site owner is trying to accomplish.

Why Do People Visit Websites?

Generally, people look at websites for two primary reasons:

1. To find information they need. This could be anything from a student looking for pictures of frogs for a school project, to finding the latest stock quotes, to getting the address of the nearest Thai restaurant.
2. To complete a task. Visitors may want to buy the latest best-seller, download a software program, or participate in an online discussion about a favorite hobby.

The main thing to remember in creating a website is that you're not creating the website for you; you already know about the information or service you have to offer. You're creating the site for your visitors, so it should contain the content they want, and be organized in a way that makes sense, even to an outsider.

We'll tell you how to create and improve your website in further articles, but the main thing to remember is this: A website is a means of communication, and it is only successful when its message is received by the intended user.

1.5 Home Pages

The World Wide Web consists of files, called pages or home pages, containing links to resources throughout the Internet. Web pages can be created by user activity.

Example: If you visit an Internet search engine and enter keywords on the topic of your choice, a page will be created containing the results of your search.

Access to Web pages may be accomplished by:

1. Entering an Internet address and retrieving a page directly.
2. Browsing through pages and selecting links to move from one page to another.
3. Searching through subject directories linked to organized collections of Web pages.
4. Entering a search statement at a search engine to retrieve pages on the topic of your choice.

Task Analyze the major advantages of home pages.

1.6 Hyperlinks

A hyperlink is a word, phrase, or image that you can click on to jump to a new document or a new section within the current document. Hyperlinks are found in nearly all Web pages, allowing

Notes

users to click their way from page to page. Text hyperlinks are often blue and underlined, but don't have to be. When you move the cursor over a hyperlink, whether it is text or an image, the arrow should change to a small hand pointing at the link. When you click it, a new page or place in the current page will open.

Hyperlinks, often referred to as just "links," are common in Web pages, but can be found in other hypertext documents. These include certain encyclopedias, glossaries, dictionaries, and other references that use hyperlinks. The links act the same way as they do on the Web, allowing the user to jump from page to page. Basically, hyperlinks allow people to browse information at hyperspeed.

Did u know? What is the core definition of hyperlinks?

An element in an electronic document that links to another place in the same document or to an entirely different document.

Self Assessment

Fill in the blanks:

10. A website is a collection of …….
11. Hyperlinks, often referred to as just "links," are common in Web pages, but can be found in other ……. documents.

1.7 Hypermedia

Hypermedia is a style of building systems for organizing, structuring and accessing information around a network of multimedia nodes connected together by links (Conclin, 1987). The general Structure of hypermedia allowed hypermedia to be applied to a wide variety of task domains. We can distinguish hypermedia systems in two generations (Halasz, 1988). First generation hypertext systems were mainframe based, text-only systems for augmenting the performance of information processing environments, storing the whole world's literature or for supporting traditional writing and reading. Transition from hypertext to hypermedia took place with the Second Generation Systems.

First Generation Systems — Introduction

We can divide hypermedia systems in two generations (Halasz, 1998). The first generation included systems such as Xanadu, ZOG, NLS/Augment, Hypertext Editing System, FRESS, Dynabook. They were mainframe-based text-only hypertext systems. They had support for multiple users sharing the hypermedia information network. The main characteristic of first generation systems is their target task domain and scope. They had been proposed as the mechanism for storing and retrieving the whole world's literacy, as a natural mechanism for reflecting the mind, as an augmentation environment for supporting users, as a replacement of traditional text writing and reading. They were primarily for authoring purposes and thus navigational aid capabilities were limited. They didn't provide any particular mechanism to extent the environment or to customize it to particular user needs. Nodes were untyped, without supporting composites. Links were single direction, single destination. The only structure supported beyond graphs was hierarchical structure. Graphical browsers were non-existent and concepts like guided tours or metaphors were not used.

Notes

Caution Search mechanism was limited to simple string search. The user interface was based in simple text monitors and it was frame based.

The second generation of Hypermedia

The second generation of Hypermedia began in 80s with the arrival of workstation-based research oriented systems like Notecards, Neptune, Intermedia, KMS, Writing Environment, Emacs/ INFO, Document Examiner. These systems are similar in concept to the first generation systems. In addition, they supported graphics or animation nodes and they had more advanced user interfaces. In contrast to first generation systems second generation are designed to support one user or a small group of users. These research oriented systems were followed in next years by a number of systems running in platforms like PC and Macintosh. These systems which include Guide, HyperTies, Hypercard are more limited in functionality and scope than the workstation-based systems but in general have quite the same capabilities with them.

These systems were quite similar in concept with first generation hypertext systems, but they were workstation and PC based, with more sophisticated graphics interfaces and support to other forms of information such as graphics, sound, animation, video.

1.7.1 Hypermedia — Limitations, Problems

While hypermedia has become more popular and hypermedia systems come into more widespread use, limitations and shortcomings of current hypermedia are becoming increasingly apparent (Halasz, 1988). The simple basic hypermedia model is not rich enough to support the organising, structuring and accessing tasks required by many applications (Hammond, 1993). Problems like user Disorientation, development of user Cognitive Overhead and manual construction of information network dominate current hypermedia systems (Ramaiah, 1992). Hypermedia Problems are interrelated (Charles, 1993).

Example: When users are disoriented the development of high cognitive overhead is very possible.

Additionally, applicability of general purpose hypermedia systems to specific tasks is problematic for casual non programmer users.

Manual Definition of Hypermedia Information Network

Manual discovery and definition of hypermedia links is a very painful and time consuming process with doubtful results. One important issue, especially in large hypermedia systems is identifying the links between pieces of data. Depending on the type of a hypermedia system, links can be navigational and organisational. Navigational links refer to another node, and they give users the ability to "move" from one node to another. Organisational links implement a network of "related to a subject" information nodes. Apart from these essential types of links several other types of links can exist to increase the functionality, efficiency and productivity of the "running" hypermedia system. All this types of links must be discovered and identified by the developer (author) of a hypermedia system. Then these links must be explicit defined in the hypermedia system. The existence of these links will "create" the network of ideas that the hypermedia system has the intention to demonstrate to the hypermedia user-explorer. This process of explicit discovering, identifying and defining links between nodes of information can be compared to the process of knowledge acquisition that

Notes

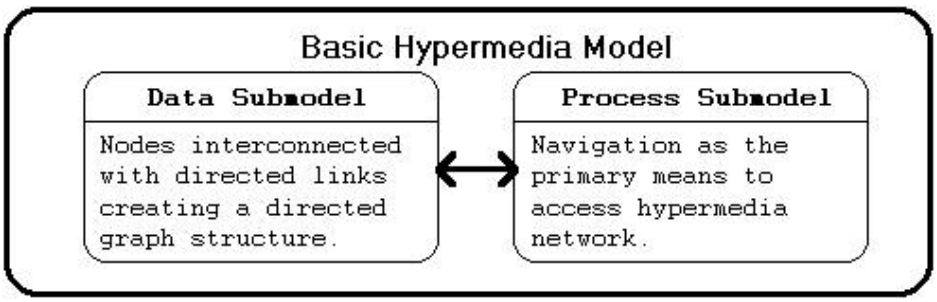
a knowledge engineer undertakes to build a knowledge base. As the expense and effort to acquire knowledge form experts in a specific domain is large, the same is the effort for discovering and defining explicit links between nodes in hypermedia. This is a very painful, time consuming process with doubtful completeness, accuracy and consistency of the final hypermedia "running" system.

1.7.2 Hypermedia — Basic Hypermedia Model

Virtually, all hypermedia systems are founded in basic hypermedia model. Likewise, a large part of current hypermedia research assumes the underlying existence of this basic model (Rivlin et al., 1994). Thus, it would be useful if we define the basic hypermedia model. We can divide basic hypermedia model in two distinguished but interdependent submodels. The first one is the data submodel. According to this submodel Nodes are interconnected with directed Links forming the structure of a directed graph (Parunak, 1991). Addition, deletion, update of nodes, links are valid operations. Process submodel is the second element of basic hypermedia model. This submodel concerns the information access mechanisms of information network. Figure 1.5 shows the data and process submodels of basic hypermedia model. This model is foremost characterised by its generality, flexibility and incompleteness.

Example: E.F. Codd in his ACM Turing award lecture (Codd, 1981) defines a data model as a combination of a data structure, operations and integrity rules. In this sense, hypermedia data model is incomplete, since it does not define any constraints (e.g. is valid the insertion of a link without associate it to a destination node, see Gronbajek et all, 1994) for determining the consistency of information network. On the other side, process submodel is very primitive, without using detailed specifications for defining navigation access (e.g, how a user activates a link).

Figure 1.5



Hypermedia - Process Submodel

Hypermedia concept is not only organised multimedia data interconnected with links. Navigation, whereby the user moves through the hypermedia network by activating and following links from one node to another, is another defining feature of hypermedia (Nielsen, 1990B). Navigation is the primary means to access information in hypermedia network, composing the most essential aspect of basic hypermedia process submodel. The basic characteristic of navigational access is that users navigate by self motivation without having any external navigational aid. Some primary navigational functionality like the ability to

Notes

backtrack to previous visited node, or to move the very first visited node, could be regarded as part of basic process submodel.

1.8 HTTP (Hypertext Transfer Protocol)

The standard Web transfer protocol is HTTP. It transmits hypertext over networks. The name is somewhat misleading in that HTTP is not a protocol for transferring hypertext; rather, it is a protocol for transferring information with the efficiency necessary for making hypertext jumps. The data transferred by the protocol can be plain text, hypertext, audio, images or any other Internet accessible information.

The HyperText Transport Protocol (HTTP) is an application-level Protocol used by Web client and Web servers to communicate with each other. HTTP has been in use since 1990. The HTTP is a transaction-oriented client/server protocol. To provide reliability, HTTP makes use of TCP. Although the use of TCP for the transport connection is very common, it is not formally required by the standard. As and when ATM networks become commercially available the HTTP requests, replies can be carried in AAL5 just as well.HTTP is a "stateless" protocol. Each transaction is treated independently. Therefore, a typical implementation will create a TCP new connection between client and server for each transaction and then terminate the connection as soon as the transaction is complete. Each interaction consists of one ASCII request, followed by one RFC 822 MIME-like response i.e., Messages are in a format similar to that used by Internet Mail and the Multipurpose Internet Mail Extensions (MIME).

Task HTTP is constantly evolving, several versions are in use and others are under development. Analyse

The World Wide Web provides a single interface for accessing all these protocols. This creates a convenient and user-friendly environment. It is no longer necessary to be conversant in these protocols within separate, command-level environments. The Web gathers, together these protocols into a single system. Because of this feature, and because of the Web's ability to work with multimedia and advanced programming languages, the World Wide Web is the fastest-growing component of the Internet.

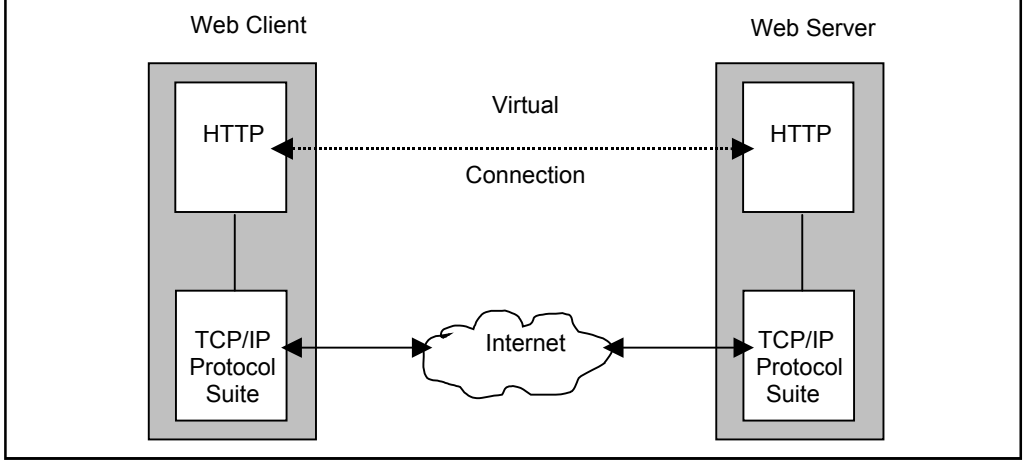
1.8.1 Understanding Hypertext Transport Protocol (HTTP)

HTTP is a request/response protocol. A web client establishes a connection with a Web server and sends a resource request. The request contains a request method, protocol version, followed by a MIME-like message. The message contains request modifiers, client information, and possible body content.

The Web server responds with a status line, including the message's protocol version and a success or error code. It is followed by a MIME-link message containing server information, entity meta-information, and possible body content. Figure 1.6 shows where the HTTP layer fits into Web client and servers.

Notes

Figure 1.6: The Web Client Communicates with the Web Server using an HTTP Virtual Circuit



Details of HTTP can be found in the following Request for comments (RFC):

- HTTP 1.0 specifications are described in RFC 1945:

http://www.cis.ohio-state.edu/htbin/rfc/rfc1945.html

- MIME specifications are described in RFC 1521:

http://www.cis.ohio-state.edu/htbin/rfc/rfc1521.html

1.8.2 Hypertext: The Motion of the Web

The operation of the Web relies primarily on hypertext as its means of information retrieval. HyperText is a document containing words that connect to other documents. These words are called links and are selectable by the user. A single hypertext document can contain links to many documents. In the context of the Web, words or graphics may serve as links to other documents, images, video, and sound. Links may or may not follow a logical path, as each connection is programmed by the creator of the source document. Overall, the WWW contains a complex virtual web of connections among a vast number of documents, graphics, videos and sounds.

Producing hypertext for the Web is accomplished by creating documents with a language called Hyper Text Markup Language, or HTML. With HTML, tags are placed within the text to accomplish document formatting, visual features such as font size, italics and bold, and the creation of hypertext links. Graphics may also be incorporated into an HTML document. HTML is an evolving language, with new tags being added as each upgrade of the language is developed and released.

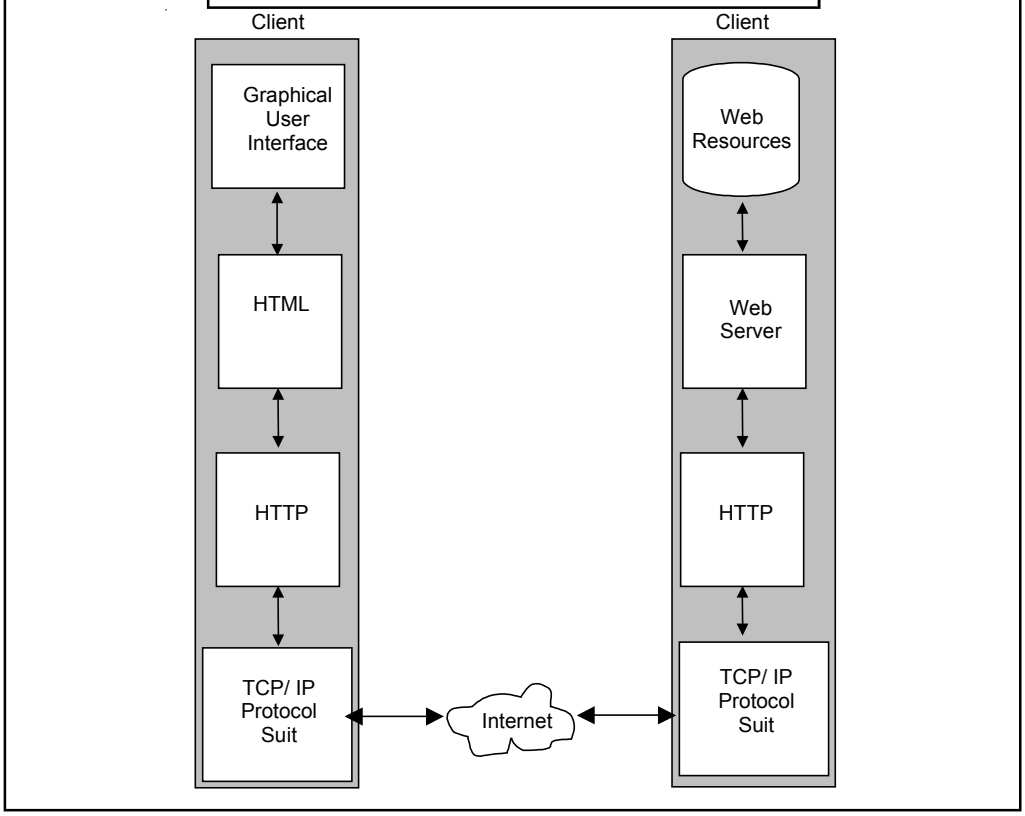
Notes The World Wide Web Consortium, led by Tim Berners-Lee, co-ordinates the efforts of standardising HTML.

1.8.3 Understanding Hypertext Markup Language (HTML)

The Hypertext Markup Language is a document-layout, hyperlink specification, and markup language. Web clients use it to generate resource requests for Web servers, and to process output returned by the Web server for presentation. A markup language described what text means and what it is supposed to look like. Figure 1.7 shows where the HTML layer fits into Web clients.

Notes

Figure 1.7: HTML is transported between the Web client and Web server over the HTTP Virtual Connection



A fundamental property of HTML is that the text it describes can be rendered on most devices.

Self Assessment

Fill in the blanks:

12. ……. are more limited in functionality and scope than the workstation-based systems but in general have quite the same capabilities with them.
13. The Web server responds with a status line, including the message's ……. version and a success or error code.
14. ……. use it to generate resource requests for Web servers, and to process output returned by the Web server for presentation.

1.9 WWW

The World Wide Web and Internet have impacted the world including business, social, political life in the last few years. It is expected that this trend will certainly continue well into the future too. You must be very much familiar with the term World Wide Web, which is also known as web or WWW or W3 and has established itself as the most popular part of the Internet by far. It is an incredible mines of information, once you start searching anything ranging from documents to pictures to software, it almost appears limitless. It provides you documents, sound files, view images, animation, and video, speak and hear voice, and view programs that run on practically on any software in the world. Therefore, it facilitates the rich and diverse communication by

Notes

enabling you to access and interact with text, graphics, animation, photos, audio and video. It has now become so simple for you to understand how the web works and what it is. Its implementation is based on client server system and employs your personal computer as client, web browser software, a connection to an Internet service provider, servers, routers and switches to direct the flow of information. You may be aware of all terms used in the formation of a web except web browser.

A browser is software, which your computer uses to view WWW documents and access the Internet. The browser program residing in your computer facilitates you with the advantages of text formatting, hypertext links, images, sounds, motion, and other features. Internet Explorer and Netscape are some of the widely used browsers. Browsers have sub programs called plug-ins to handle the documents you find on the Web. It may also other plug-ins stored elsewhere in your computer.

Web is very simple to use. Whenever you wish to visit any website, say your institute's website, you simply enter the address or URL of the website in your web browser to forward your request to the web server of the institute to provide you the intended web page. The institute's web server then sends your request on the Internet to find the intended website. Once it is obtained, the web server returns the same to your computer where the browser loaded with different plug-ins interprets the data, displaying it on your computer screen. The intended web page, which is now available on your desktop, may have click able links. On clicking on the same, you may visit other pages. In this manner, the information scattered across the globe can be linked together.

It now becomes essential to explain as to how the different web pages with different text format and standards could be linked to a particular web page. The binding forces that hold the Web together are the hypertext and the hyperlinks. The hyperlink allows electronic files on the Web to be linked so you can jump easily between them using hypertext protocol. As you have learnt that web browsers that enables you to access the Web also distinguish between web pages and other types of data on the Internet because web pages are written in a computer language called Hypertext Markup Language or HTML.

The World Wide Web is an information space that provides resources, which are identified by global identifiers called Uniform Resource Identifiers (URI). There is software, which in conjunction with servers, proxies, spiders, browsers and multimedia applications enables an user to identify and explore resources on web space.

Did u know? What will be the definition of WWW in networking language?

In networking language, www is a client-server information system that utilizes the Internet to access computers containing millions of hypertext documents.

Advantages

Many companies have understood the advantages of having a presence on the World Wide Web and have successfully addressed their corporate objectives by integrating their web site as part of their business strategy. As we are aware that a website can generate awareness of the products and services of the company and provide a global storefront for the company 24 hours a day with automating many business procedures. It is relatively inexpensive and versatile for establishing and maintaining a website. Its interactive feature make it superior than other advertising mediums. Below are the some advantages offered by WWW:

- ***Presence on the web:*** It enables businesses to be in touch with several million people who have access to the World Wide Web with more and more added everyday. No business can afford to ignore this many potential customers.
- ***Networking:*** It helps in developing lines of communication to enhance contact with potential clients and organizations. It helps in speedy and reliable communication and advertisement.

Notes

- ***Provide business information:*** It facilitates the websites to publish business services, hours, location, phone and e-mail for the public to view like any printed form of advertising. Unlike the conventional advertisement, the website provides instant communication with information about the business that may change regularly.
- ***Service to customers:*** A website provides access to business information and services to their customers online that may not be available any other way. The customers can be from anywhere in the world and shop in online stores like never before and from the comfort of their homes. They can easily and quickly search the database to locate the exact item that they were looking for and purchase it online.
- ***Conduct Business:*** A website may provide means of doing business.
- ***Provide files to download:*** Details and information of products and services in the form of pamphlets, brochures, advertisements, and even a demonstration video can be down loaded from the company's website.
- ***Remote office access:*** It facilitates offices and employees to be in touch with one another from remote places to accomplish their tasks effectively.

1.9.1 Terminology

- ***World Wide Web (the Web, WWW, W3):*** It is defined as a client-server information system using the Internet to access computers containing millions of hypertext documents.
- ***Web page:*** It is a single hypertext document written in Hypertext Markup Language (HTML) and described in HTML basics. This normally contains the basic information and links to navigate in the websites to which it belongs.
- ***Website:*** It is written in HTML and is a collection of linked Web pages on a Web server. Web server is the machine where a website is located or hosted. It may be organization owned or Internet Service Provider (ISP) owned.
- ***Home Page:*** It provides a point of entry to a Website with help. It also contains all relevant links of that particular website.
- ***Web client:*** It refers to the computer and software used to access a website and web pages.
- ***Web server:*** It is the computer or server which provides a space for hosting a website. The web client access web servers to retrieve information from a website.
- ***Web browser:*** It is the client software used to explore and display web pages from a website.
- ***Search engines:*** They are software that enables searching of the content available on Internet.
- ***Hypertext:*** It defines the documents containing embedded links (hyperlinks) to other documents or other parts of the same document.
- ***HTML:*** Hypertext Markup Language defines the rules for formatting a web page so that a web browser displays the page properly. The documents available on Web are developed, using the programming language called HTML, which may be considered as the foundation of the Web. It is the HTML and other programming that makes possible hyperlinks. The click ability feature of hyperlinks makes the web unique. The working of hypertext links depends upon the URL. Each hyperlink contains URL and you send a request by clicking on the link so that the requested document can be retrieved on your machine from other computer, it may be anywhere in the world. TCP/IP and HTML could make this possible.

Notes

Notes We may now define the HyperText Mark-up Language (HTML) as a standard format for documents on the WWW, which could be viewed by using WWW browsers.

- ***URL:*** It denotes Uniform Resource Locator. It is the address of a document on the World Wide Web. Web browsers enable a person to enter a known address of a web server or a specific document within that server. Addresses begin with http://, ftp://, gopher://, WAIS://, file:// etc.
- ***Protocols:*** They are sets of communication rules that enable client machines and servers to communicate accurately with each other.
- ***Hyper Text Transport Protocol (HTTP):*** Hypertext Transfer Protocols are the rules that enable the transmission of web documents from one computer to another via the Internet. WWW is a client/server-computing environment. A client computer by clicking on a link requests a document from Web server. In order to serve the request of client, Web server uses a protocol called HTTP or Hyper Text Transport Protocol.

1.9.2 Internet and www Milestones

Early 1960s—Genesis of networking at The RAND Corporation, in a series of reports by Paul Baran. Also, Leonard Kleinrock's thesis "Communication Nets: Stochastic Message Early Flow and Delay" at MIT created the model for performance evaluation and network 1960 design. The concept of a "mesh network" of minicomputers that would use packet $ switching (in contrast to circuit switching used in phone connections) to communicate over phone lines was a revolutionary notion at the time. Until then, computer communications had centered on mainframes and point-to-point links.

1965: One of the first networking experiments took place when the TX-2 computer at M.I.T.'s Lincoln Laboratory was connected to a Scientific Data Systems Q-23 computer in Santa Monica, California [HAF96].

1968: A request for proposal was floated to create the ARPANET; Bolt, Beranek, and Newman (BBN) was awarded the prime contract. ARPA awarded other contracts to AT& T for communications circuits, Network Analysis Corporation for designing the 1968 network topology, the University of California at Los Angeles (UCLA) for a "network measurement center", and Stanford Research Institute (SRI) for a "network information center" Other sites on the nationwide net included the University of Utah in Salt Lake City and the University of California at Santa Barbara (UCSB).

1969: The first ARPANET node was installed at UCLA in September 1969, thus launching the first packet switching network connecting SRI, UCSB, and University of Utah. The actual ARPANET network that resulted used special-purpose computers known as IMPs (interface message processors) to dismantle information into small chunks called packets, transmit the packetized information to a destination computer known by an address, check for transmission errors, retransmit damaged packets, and reassemble packets at the destination sites.

Did u know? **What is Network Communication Protocol (NCP)?**

To interface between the IMPs and proprietary software on the multivendor host computers, the ARPANET researchers created Network Communication Protocol (NCP).

1971—In just two short years, approximately twenty nodes were installed, and ARPA was funding thirty different university sites as part of the ARPANET program.

Notes

1972: e-mail was invented by accident, when two programmers at BBN decided to send 1971 each other messages, not merely transfer files. Ray Tomlinson (BBN is credited with, using the ARPANET to place the world's first e-mail message in 1973).

1973-74: In the mid-1970s, the Transport Control Protocol/Internet Protocol (TCP/IP) was developed by Vint Cerf to link different packet networks. The purpose of TCP/IP was to connect different networks (copper wire, radio, microwave) and still enable the host computers to talk to each other coherently. TCP/IP is capable of connecting multiple independent networks through routers (or gateways).

1975: In July 1975 ARPA transferred management of ARPANET and Network Measurement Center from BBN and UCLA to the Defense Communications Agency (DCA; now called Defense Information Systems Agency). It was expected that direct experience with packet switching by DCA would ultimately be of wider benefit to the Department of Defense.

1978: The U.S. government decreed that TCP/IP be the preferred way to send information 1978? from one computer to another. This caused computer vendors to wake up and realize that TCP/IP is here to stay.

1980: DARPA funded the development of Berkeley UNIX. TCP/IP was made part of the 1980 operating system. The government had considered buying AT&T UNIX but felt that it didn't have enough features, primarily TCP/IP.

1983: Transition from the original ARPANET protocol, the Network Communication Protocol (NCP), to TCP. At this time only a few hundred host computers were on the nascent Internet.

1980-86: From 1980 to 1986, NSF supported the development of CSNET, a computer science research network. CSNET was a network of networks, one component of which used the TCP protocols over an X.25 public data network showing the power of a 1980.86 layered architecture. CSNET also included the ARPANET and PHONENET, a telephone-based electronic mail relaying system. By 1985, CSNET had links to over 170 university, industrial, and government research organizations and numerous gateways to networks in other countries.

1982-86: In 1982 a report, "Large Scale Computing in Science and Engineering", recommended the establishment of NSF-funded supercomputing centers as well as a high-speed network to connect them. These centers would offer an opportunity to make progress in science and engineering research. By the year 1985, NSF announced five awards, and by 1986 the Cornell Theory Center, the Illinois National Center for Supercomputing Applications (NCSA), the Pittsburgh Supercomputing Center, the San Diego Supercomputer Center, and the Princeton John von Neumann Center were up and running.

1986: NSF initiated a new program of networking and computer support for supercomputing centers to be used by researchers. This program began with a memorandum of understanding with ARPA to allow NSF-funded supercomputer centers and selected researchers to use the ARPANET. Believing that ARPANET was I not suitable, NSF instituted the NSF Connections program in 1986 to broaden the base of network users with their own computer facilities and eventually to help universities achieve access to supercomputers (by supplying hardware and telecommunications lines for direct, point-to-point connections). In 1986, it launched the NSFNET network backbone program.

1987: CSNET merged with BITNET, a worldwide network connecting IBM mainframes that was initiated in 1980-81. CSNET operations were continued under the Corporation for Research and Education Networking (CREN), whose operating costs were completely covered by member organizations' dues.

1987: After significant congestion was experienced in 1987, the backbone was upgraded from 56 kbps to TI service (1.5 Mbps) and became operational in 1988.

Notes

1988: The Internet virus is unleashed by a graduate student at Cornell University, focusing attention on network vulnerability to security threats. Immediate steps were taken to make the network more secure.

1990: Twenty years after its birth at UCLA, ARPANET was officially decommissioned; its descendant, the NSFNET, inherited its role as the research and education 1990 communities' backbone network. The first relay between a commercial electronic mail carrier (MCI Mail) and the Internet took place through the Clearinghouse for Networked Information.

Its mission accomplished, CSNET service was discontinued. For the first time, commercial networks were connected to the NSFNET backbone through the Commercial Internet Exchange (CIX) Association. CIX was formed by General Atomics (CERFnet), Performance Systems International, Inc. (PSINet), and UUNET Technologies, Inc. (AlterNet).

1991: Anew bread of distributed information services called Wide Area Information Servers (WAIS) released by the now-bankrupt Thinking Machines Corporation; Gopher was released by the University of Minnesota, and the World Wide Web was announced on alt.hypertext by Tim Berners-Lee of CERN.

The U.S. government made a decision to turn NSFNET into a faster research network called National Research and Education Network (NREN) as defined in the High-Performance Computing Act of 1991.

1993: National Information Infrastructure announcement sparks interest in the Information 1993 Superhighway. Businesses and media suddenly realised there was something called the Internet and began to take an interest in its exploitation.

1994: Two million copies of a freeware mosaic -a multimedia browser for the WWW, written by Marc Andreesen, at that time an undergraduate student at the University 1993.94 of Illinois at Urbana Campaign -were distributed over the Internet and attained incredible popularity. This milestone event represents anew chapter in electronic commerce.

1995: The old NSFNET backbone is decommissioned and anew architecture based on Network Access Points (NAPs) is installed.

2000: IT Act 2000 passed by the Government of India.

Task Write the full form of the following abbreviations:

1. HTTP
2. www
3. ARPA
4. DOD
5. NAP

Self Assessment

Fill in the blanks:

15. The World Wide Web is an information space that provides resources, which are identified by global identifiers called …….
16. Hypertext Markup Language defines the rules for formatting a ……. so that a web browser displays the page properly.

Notes

1.10 Web Server

Although technically not a component of Microsoft Speech Server (MSS), a Web server running Microsoft Internet Information Services (IIS) is an integral part of a complete MSS deployment scenario. (IIS is included with Microsoft Windows Server 2003 but it must be explicitly installed. Click Add/Remove Windows Components and select both IIS and ASP.NET.)

The Web server hosts the speech application, including application resources such as prompt databases and grammar files. It generates the application Web pages that contain HTML, SALT, and script. A Telephony Application Services SALT interpreter, or a client device with the Speech Add-in installed, connects to the Web server.

The Web server hosts ASP.NET pages (.aspx) for applications developed using the Microsoft Speech Application SDK (SASDK), or using ASP.NET Speech Controls.

Speech Application Deployment Service

Speech Application Deployment Service (SADS) provide a way for MSS to manage the deployed speech application. When installed on the Web server, SADS appears in the Microsoft Management Console (MMC) snap-in for MSS, providing the ability to add, remove and update deployed applications.

Prompt Database

A prompt database is an application-specific repository of prerecorded sound files used by the speech output engines of Speech Engine Services. To improve prompt quality, and therefore the quality of the speech output, application developers may hire a professional to record the prompts for the database.

Grammar Files

A grammar file contains a structured list of words and phrases that SAPI parses for the Speech Engine Services speech recognition (SR) engine. Grammars are specific to applications developed for the Speech Application Platform.

Task Analyze What are the advantages of Web server?

1.11 Client/Server Architecture Model for Web Request

A web is similar to the server in client/server technology. The server, in client/server technology, usually connects to a database. The client, in client/server technology, makes a data request to the server, processes the returned data, and presents the result through a graphical user interface.

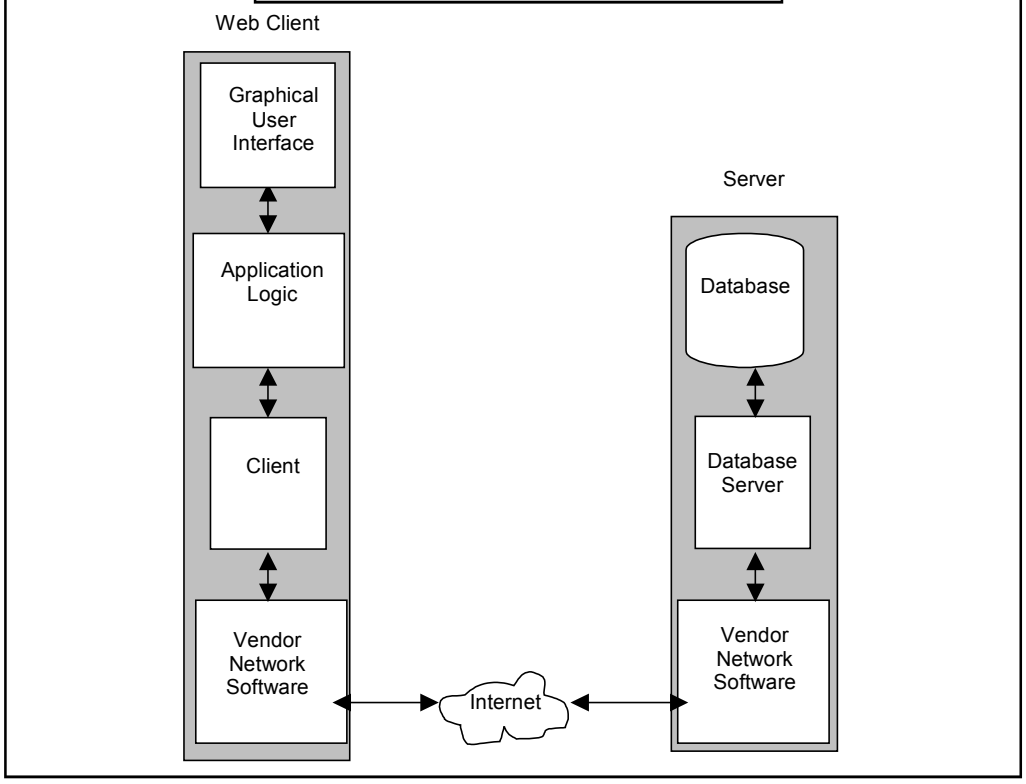
A web client makes a resource request to the Web server, processes the returned resource, and presents the result through a graphical user interface.

The difference between a server, in client/server technology, and a Web server seems to be one accepts requests for data, and the other accepts requests for a resource. The differences are dramatic, as we look closer.

The server, in client/server technology, is typically a specialized database server. The Microsoft SQL server product is a good example. A database server receives requests for data from a client through vendor proprietary network software. It locates the data and returns it. The client applies application logic to the data, and presents the result through a graphical user interface. This is a "fat client" because the application logic is in the client. Figure 1.8 illustrates client/server components.

Notes

Figure 1.8: Client/Server concept: A Fat Client



Not just any client in client/server technology can request data from the database server. All clients must run the correct vendor proprietary network software.

Notes Likewise, a database server can only locate the return data form the vendor proprietary database, without using a database gateway to another proprietary database.

The network connection between the client and the database server remains until one or the other closes it, or until the network fails. The database server retains state information about the client for the entire lifetime for the connection. This saves the database server time in completing requests for data.

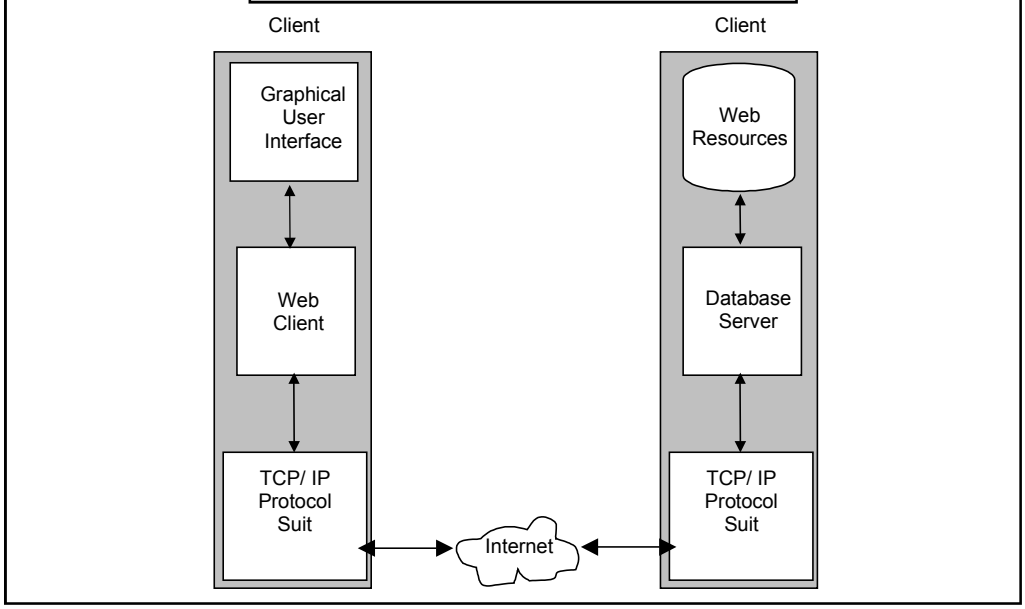
Web servers receive requests for a resource from Web client through the standard TCP/IP protocol Suite. The resource can be a file, or data returned by another process. The Web server locates the file and returns it, or executes another process, supplies it with input, and returns the output. The Web client does not apply application logic to the resource. It presents the resource through a graphical user interface. This is a "thin client" because it does not contain application logic. Figure 1.8 illustrates Web client/server components.

Any web client can request a resource from any Web server, and any Web server can request a resource from any other web server. This is possible, because they use the standard TCP/IP Protocol Suite.

The network connection between the Web client and Web server remains only until the Web server has returned the resource. The Web server does not retain any state information about the Web client.

Notes

Figure 1.9: Client/Server concept: A "Thin Client"



1.12 URL

A URL contains the following information:

- ***Protocol:*** Specifies the Internet protocol to access a resource. The abstract encompasses FTP, Gopher, and HTTP Internet Protocols.
- ***Network Endpoint:*** Internet Addresses of Internet Information Server and Protocol Port Number.
- ***Resource Location:*** Path information to locate a resource on Internet Information Server URL syntax is {service}: ://{host} [: port]/[path /… /] [file name]

Required parameters are surrounded by { }. Optional parameters are surrounded by [ ]. Other characters are mandatory separators.

Notes *Service* is a required field. Web server support FTP, Gopher and HTTP services.

Host is a required field. This field is the host name or IP address of the Internet Information Server.

Self Assessment

Fill in the blanks:

17. A prompt database is an ……. repository of prerecorded sound files used by the speech output engines of Speech Engine Services.
18. A database server receives requests for data from a client through ……. proprietary network software.

Notes

Caselet

'Internet will Take time to Evolve as Business Model'

Since it took a century for the industrial revolution to become a reality, it would be unrealistic to expect the Internet to become a business model overnight, said Mr Ajit Balakrishnan, founder and CEO of rediff.com.

Speaking at the IIM-Kozhikode on the topic 'Entrepreneurship in e-business: Current challenges and future perspectives', he recounted the struggles and setbacks faced by rediff.com during the early days of its listing on Nasdaq and the subsequent collapse of all Internet stocks.

IIM-Kozhikode had organised the two-day Annual IT Management Seminar with the theme 'IT India: Crossing the Chasm.' The event was organised by AbaKus, the Systems interest group at IIMK. TCS and Satyam were the sponsors of the event, which was supported by Nasscom.

Addressing the session Ms Jessie Paul, Chief Marketing Officer, Wipro Technologies, talked about how branding is both an art and a science. She remarked that a brand is not about advertising and public relations alone, but must deliver its promise to attain the affection and respect of customers.

Mr Sriniketh Raman Chakravarthi, Senior Manager, Corporate Development, Cognizant talked about global outsourcing landscape and the evolution of IT services.

Mr K. Ganesh, non-Executive Chairman of Marketics; Col K.P.M. Das, VP, Mobile Computing Products Group, Encore Technologies; Ms Aruna Jayanthi, VP & Head of Outsourcing, Capgemini; and Mr Viswanathan Krishnamurthi, CIO, Yahoo! Bangalore, also shared their experiences and insights on the occasion.

1.13 Summary

- The beginning of the Internet dates back to the late 1960s, when the Advanced Research Projects Agency (ARPA) of the Department of Defense (DOD) formed ARPANET.
- The World Wide Web is also known as the 'www' or the 'web'. It is an architectural framework of information system fully implemented in 1994 on the Internet.
- It contains millions of electronic documents called Web Pages.
- A web page contains text and graphics (drawings) which are linked to related information.
- The Web is not internet.
- The Internet means a network infrastructure that is built on certain standards, which are followed by all participants to connect to each other.
- The Internet Protocol (IP) defines how the flow of information is organised.
- World Wide Web lays down the specifications of information and services to be exchanged with its Hyper-Text Transfer Protocol (HTTP).
- Thus, the web offers the exchange of documents via HTTP.
- The World Wide Web provides a simple-to-use web interface that allows people with very little knowledge in computing to access web services all over the Internet.

Notes

- The World Wide Web that has become de facto standards for any professional belonging to any discipline finds its extensive use and utility in providing information stored in computer system attached to the Internet or www.
- Different web designing techniques are used to make the information in a presentable form to the users. Some of the web designing techniques such as HTML has become a standard for web pages.
- In addition to the above, In HTML webpage design, the limited use of colors often makes the appearance of the colors more powerful.
- It is also possible to add an image or a plain color as background with the help of its specifications in the <body> tag. Form also gives navigability to a website.
- Forms are objects that enable to enter information in the form of text boxes, drop-down menus or radio buttons. Front Page provided by Microsoft, however, provides an excellent tool for designing WebPages with very minimal knowledge of HTML has been replaced by more advanced tools.
- FrontPage enables to work in a WYSI–A–WYG (What You See Is –Almost– What You Get) environment that facilitates a great deal like the Microsoft word processing application, Word.
- Web browser is software to run the hypertext transmission protocols to retrieve a webpage from its destination.
- The browser program residing in the end user computer facilitates the advantages of text formatting, hypertext links, images, sounds, motion, and other features.
- Internet Explorer and Netscape are some of the widely used browsers. Browsers have sub programs called plug-ins to handle the documents you find on the Web.
- It may also other plug-ins stored elsewhere in your computer.
- Search engines with different approaches provide an excellent method of searching WebPages on www in minimal possible time. Contribution of search engines is enormous to the growth of www and Internet.

1.14 Keywords

Frames: It is for displaying more than one HTML document in the same browser window or to divide the screen into separate windows.

Home Page: It provides a point of entry to a Website with help. It also contains all relevant links of that particular website.

HTML: Hypertext Markup Language defines the rules for formatting a web page so that a web browser displays the page properly.

Hyper Text Transfer Protocol (HTTP): Hypertext Transfer Protocols are the rules that enable the transmission of web documents from one computer to another via the Internet.

Hypertext: It defines the documents containing embedded links (hyperlinks) to other documents or other parts of the same document.

Internet: It means a network infrastructure that is built on certain standards, which are followed by all participants to connect to each other.

Internet Protocol (IP): It defines how the flow of information is organized.

Search Engines: They are software that enables searching of the content available on Internet.

Notes

URL: It denotes Uniform Resource Locator. It is the address of a document on the World Wide Web.

Web Browser: It is the client software used to explore and display web pages from a website.

Web Client: It refers to the computer and software used to access a website and web pages.

Web Page: It is a single hypertext document written in Hypertext Markup Language (HTML) and described in HTML basics. This normally contains the basic information and links to navigate in the websites to which it belongs.

Web Server: It is the computer or server which provides a space for hosting a website. The web client access web servers to retrieve information from a website.

Web Site: It is written in HTML and is a collection of linked Web pages on a Web server. Web server is the machine where a website is located or hosted. It may be organization owned or Internet Service Provider (ISP) owned.

World Wide Web (WWW): It is defined as a client-server information system using the Internet to access computers containing millions of hypertext documents.

World Wide Web: It is an architectural framework of information system which offers the exchange of documents via HTTP.

1.15 Review Questions

1. Compare and contrast between internet and www.
2. The Internet is an important tool for practically everybody. Analyze this statement and discuss the evolution of internet.
3. A web browser is the software program used to access the World Wide Web that is the graphical portion of the Internet. Give reasons
4. Explain why the server, in client/server technology, is typically a specialized database server?
5. There are several search engines. Explain the different types of search engines with their characteristics.
6. What is relevance of Front Page in designing web pages when most of the web pages are designed in HTML?
7. How is the HTML document used for making a hyperlink? Explain with example.
8. Provide a link of another site and define links to send an email to different people. Take relevant example.
9. Hypertext is a document containing words that connect to other document. Do you agree with this statement? Why or why not? Give reasons
10. The resource can be a file, or data returned by another process. Discuss

Answers: Self Assessment

Fill in the blanks:

1. network technologies
2. nuclear event
3. Searchable
4. hyperlinks

Notes

5.	private	6.	home pages
7.	NNTP	8.	Web servers
9.	web browser	10.	web pages
11.	hypertext	12.	Hypercard
13.	protocol	14.	Web clients
15.	Uniform Resource Identifiers (URI)	16.	web page
17.	application-specific	18.	vendor

1.16 Further Readings

Books

Rajneesh Agrawal and Bhata Bhushan *Computer Networks and Internet Tiwari*, published by Vikas Publication

Andrew S. Tanenbaum, *Computer Networks,* published by Prentice Hall

Behrouz A. Forouzan, *Sophia Chung Data Communications and Networking*, Fegan published by McGraw-Hill Companies

Dr. Ravi Kalakota, Marcia Robinson, *E-business Road Map for Success*.

G.Winfield Treese Lawrence C. Stewart, Designing Systems for Internet Commerce.

Kalakota and Whinston, *Frontiers of Electronic Commerce*.

Kamelesh K Bajaj, Debjani Nag, *E-commerce the Cutting Edge of Business*.

Burton, Bill, *Remote Access for Cisco Networks,* published by McGraw-Hill Osborne Media

Web Tutorials on HTML and Front Page

Online links

http://en.wikipedia.org/wiki/Internet

http://www.w3.org/

Notes

Unit 2: Creating Static Web Pages

CONTENTS

Objectives

Introduction

2.1 HTML Document Structure

2.1.1 Document Tags

2.1.2 HTML Tag

2.1.3 Text Structures

2.1.4 Headings

2.1.5 Paragraphs and Line Breaks

2.1.6 Logical and Physical Markup

2.1.7 Special Characters

2.1.8 Formatting Text with Attributes

2.1.9 Changing fonts

2.1.10 Colors

2.2 Singular and Paired Tags

2.3 Adding Images

2.3.1 The Alt Attribute

2.3.2 Hyperlinks

2.4 Audio and Video

2.4.1 Using the <video> tag

2.4.2 Using the <audio> tag

2.5 Creating List

2.6 Tables

2.7 Forms

2.8 Frames

2.9 Using Multiple Windows in Web Pages

2.10 Summary

2.11 Keywords

2.12 Review Questions

2.13 Further Readings

Notes

Objectives

After studying this unit, you will be able to:

- Scan the HTML Document Structure
- Describe Singular and Paired Tags
- Demonstrate the Hyperlinks and Adding images
- Recognize Audio and Video tags
- Explain the creation of lists, tables, forms, frames
- Scan the using of multiple windows for web pages

Introduction

HTML (Hyper-Text Markup Language) is composed of tags, i.e., commands in angle-brackets (< >). HTML tags are case-insensitive, that is, it doesn't matter whether you type them in upper or lower case.

2.1 HTML Document Structure

- An HTML element starts with a start tag/opening tag
- An HTML element ends with an end tag/closing tag
- The element content is everything between the start and the end tag
- Some HTML elements have empty content
- Empty elements are closed in the start tag
- Most HTML elements can have attributes

Tags typically occur in begin-end pairs. These pairs are in the form tag> ... </tag> Where the <tag> indicates the beginning of a tag-pair, and the </tag> indicates the end. (The three dots indicate an arbitrary amount of content between the tags.). These pairs define containers. Any content within a container has the rules of that container applied to it. For example, the text within a "boldface container" would be boldfaced. Similarly, paragraphs are defined using a "paragraph container."

Some commands do not consist of a begin and end tag, but just of a single tag. In HTML, this is just a begin tag:<tag>

2.1.1 Document Tags

The first tag in an HTML document is the "!DOCTYPE" tag. It looks like this

(or similar):

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

This says that the document is written in HTML, version 4.01, and that all requirements of HTML 4.01 are strictly adhered to. Don't worry if that line looks a bit complicated—just type it into your web page exactly like you see it above, and it will work for you. The DOCTYPE declaration needs to be written in upper case, just like you see it above—it is an exception to the general rule of writing tags in lower case.

Notes

If you want to use commands from previous versions of HTML, you can use the transitional variant of HTML 4.01 instead:

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

In older web pages, you may note that the DOCTYPE declaration is missing. This is because it was not required in earlier versions of HTML.

Caution You can still write web pages without it and most browsers will display your page correctly. However, in order to write a valid HTML page, it is required nowadays.

2.1.2 HTML Tag

After the DOCTYPE declaration come the "html" tags. These are the tags that tell a web browser where the HTML part in your document begins and ends.

```
<html>
</html>
```

Inside the "html" container, we have the "head" and the "body" container:

```
<html>
<head>
</head>
<body>
</body>
</html>
```

The "body" contains the actual content of your web page.

The "head" contains all of the document's header information like the web document's title and information about the document itself. This is an important point for search engines.

The container "title" is placed within the head structure. Between the title tags, you should have the title of your document. This will appear at the top of the browser's title bar, and also appears in the history list. Finally, the contents of the title container go into your bookmark file, if you create a bookmark to a page.

Also, the head contains meta information about the document, most importantly the character encoding that is used.

The encoding "ISO-8859-1" is used for English, French, Spanish, German and other Western European languages.

Here you see an example of a "head" container with a "title" element and a "meta" element denoting the encoding:

```
<head>
<title>This is my very first HTML document</title>
<meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1">
</head>
```

Again, you can just copy the "meta" element as you see it above into your document.

Notes

The "body" comes after the head structure. Between the body tags, you find all of the stuff that gets displayed in the browser window. All of the text, the graphics, and links, and so on - these things occur between the body tags.

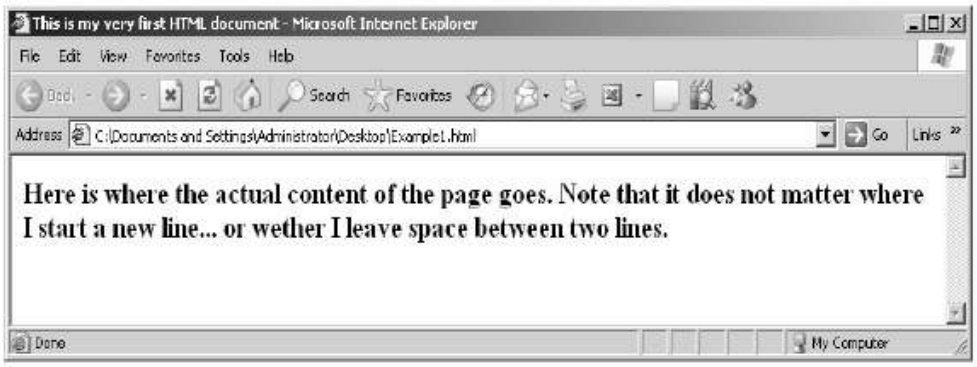
The strict variant of HTML 4.01 requires that any content inside the body is within a further set of tags (if you use the transitional variant of HTML 4.01, this is not necessary). For text, you can use "p" (the paragraph tag). A complete page would then look like this as shown in below example:

Example: Writing a paragraph in HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/ukt.dtd">
<html>
<head>
<title>This is my very first HTML document</title>
<meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1">
</head>
<body>
<p><h1> Here is where the actual content of the page goes.
Note that it does not matter where I start a new line...
or wether I leave space between two lines. </p></h1>
</body>
</html>
```

This will result in the following page being displayed in your browser as shown in Figure 2.1.

Figure 2.1



If you want to leave yourself notes in an HTML document, but don't want those notes to show up in the browser window, you need to use the comment tag. To do that, you would do the following:

```
<!— This is a comment. —>
```

Steps to run Example

1. Save the above example as HTMLExample.html.
2. Open the browser.

Notes

3. Open the file by browsing it
4. Get the output

2.1.3 Text Structures

As we have seen, a line break in your source file does not mean that there will be a line break on the web page displayed by your browser. So how do you structure the text on your page? For example, how do you get headings that are set apart from the text underneath? And how do you start a new paragraph that is set off by a blank line from your previous text?

2.1.4 Headings

Heading structures are most commonly used to set apart document or section titles.

There are six levels of headings, from heading 1 through heading 6. Heading1 (h1) is "most important" and heading 6 (h6) is "least important." By default, browsers will display the six heading levels in the same font, with the point size decreasing as the importance of the heading decreases. Here are two examples:

```
<h1>Heading 1: The largest</h1>
<h6>Heading 6: The smallest</h6>
```

2.1.5 Paragraphs and Line Breaks

To create an empty line between two blocks of text, you need to label those text blocks (or paragraphs) with the paragraph marker "p". So, surround your paragraphs with and like we have done in our example page. Actually, in HTML you do not need the closing tag for "p", you can create an empty line after a block by just writing . Here is an example (note that you do not actually need to leave a blank line between the two paragraphs in your source code).

This is the first paragraph. Isn´t it a nice paragraph? It has lots of words in it. It´s such a nice paragraph that I think it should never end.

This is the second paragraph. It´s also a very nice paragraph

but maybe not as nice as that first one was.

If you just want to move down to the next line, without leaving a blank space, you would use a "br", for line break. For example, like this

This is a line.
 And here is the next line.

Example: Given below is another example for using line break in your web page.

```
<html>
<body>
<p>This is<br />a para<br />graph with line breaks</p>
</body>
</html>
```

Notes

Result: The result for the above html code is given below.

This is a para graph with line breaks

2.1.6 Logical and Physical Markup

These examples demonstrate how you can mark-up text in an HTML document when you want to describe its function and desired appearance.

```
<b>This text is bold</b> <br>
<strong> This text is strong </strong> <br>
<i>This text is italic</i> <br>
<em>This text is emphasized</em>
<p>
An example for subscript: H<sub>2</sub>O <br>
An example for superscript: x<sup>2</sup>
</p>
```

And this is what it will look like:

While "b" (for "bold") and "i" (for "italics") are physical markups (they prescribe the look of the text), the markups "strong" and "em" (for "emphasized") are logical markups, i.e. they rather describe the idea behind the look. As you can see in the above example, the default behavior of a browser is usually to display "strong" passages in bold, and "emphasized" passages in italics. However, this default can be overridden with the help of style sheets (more on that later).

For example, the composer of a web site might decide that emphasized text should be displayed in bold, and strong text should be displayed in bold and in a larger font. Generally, it is better to use logical markup than physical markup, as this describes the meaning of a text passage rather than its look. You are more flexible that way and can reuse the same page later with different style settings.

2.1.7 Special Characters

Certain characters, such as the angle-brackets (< >), the ampersand (&) and others are reserved by HTML to represent special things – for example, the left angle-bracket denotes the start of an HTML tag. If you actually want an angle-bracket to appear in your text, you need to use a special HTML command. In addition there are many ISO-Latin 1 characters that you may wish to include in a document, but which are not trivially available on a standard keyboard.

HTML gives you the possibility of including these special characters through certain commands.

Commands for special characters consist of three parts:

- a leading ampersand character, (&),
- the name of the entity (in ascii characters)
- a terminating semicolon (;)

Notes

Example:

```
< &lt; (for "lower than")
> &gt; (for "greater than")
& &amp; (for "ampersand")
ä &auml; (for "a umlaut")
ö, ü &ouml; &uuml;
Ä,Ö,Ü &Auml; &Ouml; &Uuml;
€ &euro;
© &copy;
```

So, for example the German sentence München ist eine schöne Stadt! will display as: München ist eine schöne Stadt!

2.1.8 Formatting text with Attributes

Attributes can be used to change a tag's properties. For example, a paragraph's property might be its alignment (left, right or center). A text's property might be its font size or color.

To change these properties, tags can be extended by attributes:

```
<p style="text-align:center"> a centered paragraph </p>
<p style="text-align:right"> a paragraph aligned right </p>
```

The paragraph tag "p" here has the attribute "style", and the attribute is assigned a value, namely "text-align:center" (or "text-align:right").

This particular attribute "style" is our first example of CSS (Cascading Style Sheets) technology.

Style sheets allow you to specify the presentation (look) or your text. We will learn more about CSS later.

In older HTML documents, you may also see the attribute "align",

For example:

```
<p align="center"> another centered paragraph </p>.
```

This still works in HTML, but is deprecated, i.e. you are not supposed to use this attribute anymore.

Example:

```
<html>
<body>
<p><b>This text is bold</b></p>
<p><strong>This text is strong</strong></p>
<p><big>This text is big</big></p>
<p><i>This text is italic</i></p>
<p><em>This text is emphasized</em></p>
<p><code>This is computer output</code></p>
```

Notes

```
<p>This is<sub> subscript</sub> and <sup>superscript</sup></p>
</body>
</html>
```

Result: The visual result for the above code is shown below:

This text is bold

This text is strong

This text is big

This text is italic

This text is emphasized

2.1.9 Changing Fonts

You can change the size of text similarly by writing

```
<p style="font-size:250%">Pretty large text</p>
<p style="font-size:50%">Pretty small text</p>
```

Here, the first line is displayed in a large font size, the second line in a small font size. If you don't want to change the size of text for a whole paragraph, but just for a passage (a sequence of words) within a paragraph, for example, you can use the "span" tag:

```
<p> This text is in normal size. Now,
<span style="font-size:250%">this is large</span> and now we go
back to normal size.
</p>
```

Again, you might come across a deprecated older tag for specifying text properties (size, face and color): the tag. For setting size, it was used like this: Here is a size 5 font, and you could set 7 different levels of sizes from 1 (smallest) to 7 (largest), where the default was 3.

Notes However, the recommended method for setting text properties nowadays is via styles, as shown before.

To change the font type, you can use "font-family". Choose a different font face using any font you have installed. Be aware that if the user viewing the page doesn't have the font installed, it will not be displayed. Instead, the default font Times New Roman (which is a default browser setting) will be displayed. An option is to choose a few fonts that are similar in appearance:

```
<span style="font-family:Arial,Verdana,Helvetica">nice font...</span>
```

If you specify various fonts like this, the browser will try the first one first, and if that font is not installed, the second, and so on.

The old way was to use the element "font" with the attribute face:

```
<font face="Arial"> some text </font>.
```

Notes

2.1.10 Colors

The attribute changing the color is color. It's values can be color names like red, green, blue, etc.

```
<span style="color:red">red text</span>
```

However, only 16 color names are supported by the W3C HTML 4.0 standard (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow). For all other colors you must specify how the color is made up from the three base colors red, green and blue (RGB).

Did u know? What you need to specify for all three base colors?

You need to specify for all three base colors an intensity between 0 (lowest) and 255 (highest), giving you the color that results from mixing these base colors with the specified intensities.

For example, assigning 0 to red, 0 to green and 255 to blue gives you "100% blue". There are two ways of defining RGB values – firstly using hexadecimal notation, secondly using decimal numbers.

When defining the RGB values of a color in hexadecimal notation, you write a "#" followed by six digits, of which two represent the red value, two the green value and two the blue value. These are in hexadecimal notation, i.e. the lowest value 0 is written as "00", the highest value 255 is written as "FF", for example, "#0000FF".

When using the decimal numbers, you write "rgb", followed by a triplet of numbers, in parentheses and separated by commas, for example "rgb(0,0,255)".

This table shows the result of combining red, green and blue in various ways:

```
Color Color HEX Color RGB
#000000  rgb(0,0,0)
#FF0000 rgb(255,0,0)
#00FF00 rgb(0,255,0)
#0000FF rgb(0,0,255)
#FFFF00 rgb(255,255,0)
#00FFFF rgb(0,255,255)
#FF00FF rgb(255,0,255)
#C0C0C0 rgb(192,192,192)
#FFFFFF rgb(255,255,255)
<p style="color:#FF00FF">WOW! This is pink</p>
<p style="color:rgb(255,0,255)">This is pink, too!</p>
```

Example:

```
<html>
<body>
<h1 style="font-family:verdana;">A heading</h1>
<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>
</body>
</html>
```

Notes

Result: The visual result for the above code is shown below:

A heading

A paragraph.

Self Assessment

Fill in the blanks:

1. The "……" contains all of the document's header information like the web document's title and information about the document itself.
2. The container "……" is placed within the head structure.
3. Be aware that if the user viewing the page doesn't have the font installed, it will not be ……
4. When using the decimal numbers, you write "……", followed by a triplet of numbers, in parentheses and separated by commas.

2.2 Singular and Paired Tags

HTML Tags

Tags are instructions that are embedded directly into the text of a HTML document. Each HTML tag describes that the browser should do something instead of simply displaying the text. In HTML, the tags begin with (<) and end with (>)

HTML tags can be of two types. They are:

1. Paired Tags
2. Unpaired Tags

Paired Tags: A tag is said to be a paired tag if the text is placed between a tag and its companion tag. In paired tags, the first tag is referred to as Opening Tag and the second tag is referred to as Closing Tag.

Example:

```
<i>This text is in italics. </i>
```

Notes Note: Here is called opening tag. and is called closing tag.

Unpaired Tags: An unpaired tag does not have a companion tag. Unpaired tags are also known as Singular or Stand-Alone Tags.

Example:
 , <hr> etc. These tags does not require any companion tag.

Notes

2.3 Adding Images

Images can be used to make your web pages distinctive, and can greatly help to get your message across. .In HTML, images are defined with the <img> tag.

The <img> tag is empty, which means that it contains attributes only and it has no closing tag.

To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

The syntax of defining an image:

```
<img  src="url"  />
<img  src="background.jpg"  width="200"  height="150">
```

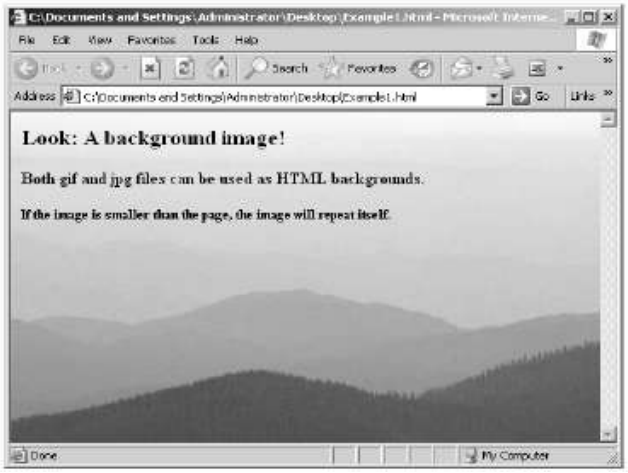
The URL points to the location where the image is stored.

The browser puts the image where the image tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

```
<html>
<body background="C:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Blue hills.jpg">
<h1>Look: A background image!</h1>
<p><h2>Both gif and jpg files can be used as HTML backgrounds.</h2></p>
<p><h3>If the image is smaller than the page, the image will repeat itself.</h3></p>
</body>
</html>
```

The output is shown in Figure 2.2.

Figure 2.2



2.3.1 The Alt Attribute

The alt attribute is used to define an "alternate text" for an image. The value of the alt attribute is an author-defined text.

```
<img  src="background.jpg"  alt="bckgrnd"  />
```

Notes

The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image.

Task It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers. Explain with suitable examples

2.3.2 Hyperlinks

What makes the web so effective is the ability to define links from one page to another, and to follow links at the click of a button. A single click can take you right across the world!

Links are defined with the <a> tag.

Let's create a link to the page defined in the file "peter.html": This link to

```
<a href="Example1.html">upendra's homepage</a>.
```

The text between the <a> and the is used as the caption for the link. It is common for the caption to be in blue underlined text. It is by the way a good idea to not have any blank spaces in the names of your HTML files, as this might create problems with some web servers. You can use an underscore, "_", to separate words in your file names.

To link to a page on another web site you need to give the full web address (the URL). For instance, to link to "Google" you need to write:

```
A link to <a href="http://www.google.com">Google</a>.
```

If you want the user's browser to open a new window for the linked page, (that way the user finds back to your page as soon as he or she closes the new window), use the attribute target:

```
A link to <a href=http://www.google.com target='_blank'>Google</a>.
```

Did u know? What are the two attributes in image tag?

```
The <img> tag has two required attributes: src and alt.
```

Example: Try the given below code to learn how an hyperlink is inserted in a webpage.

```
<html>

<body>

<p>

<a href="default.asp">HTML Tutorial</a> This is a link to a page on this website.

</p>

<p>

<a href="http://www.google.com/">Google</a> This is a link to a website on the World Wide Web.

</p>
```

Notes

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

Result: Your result window for the above code is shown below:

HTML Tutorial This is a link to a page on this website.

Google This is a link to a website on the World Wide Web.

Self Assessment

Fill in the blanks:

5. A tag is said to be a paired tag if the text is placed between a tag and its …… tag.
6. To display an image on a page, you need to use the …… attribute.
7. The …… points to the location where the image is stored.
8. The alt attribute is used to define an "…….. " for an image.

2.4 Audio and Video

2.4.1 Using the <video> tag

Lest you think what we're about to wade through is ultimately an exercise in futility, if there is no agreed upon standard, consider this: Google is chomping at the bit to use the video tag for YouTube.

In fact, there's already a mockup of what YouTube would look like using only HTML 5. While the company hasn't announced a timeline to convert YouTube to use the HTML 5 <video> tag, you can bet that when they do, a sizable chunk of the web will follow suit.

So how does video work? Well, are you ready? Here's the code to embed a video in HTML 5:

```
<video src="/myvideo.mp4"></video>
```

Well, ideally you would do something more like this (which is what the aforementioned YouTube demo does):

```
<video width="640" height="360" src="/demo/google_main.mp4?2" autobuffer>
<div class="fallback">
<p>You must have an HTML5 capable browser.</p>
</div>
</video>
```

There are also a number of useful attributes for the <video> tag, including auto-play controls, a "poster" attribute that points to an image file to display before the video is loaded and a boolean attribute for play/pause controls. The <video> tag also has a whole host of events you can hook into with JavaScript, allowing you to play movies inside movies and set up complex user interactions via mouse and keyboard events.

Notes

Example: Here's an example that uses the video tag in conjunction with the Canvas tag and Web Workers (we'll cover those in the future) to create a motion tracking system for web video.

The unfortunate answer is that you'll need to point to multiple videos. Hardly ideal, but if you want to push the HTML boundaries, you can embed your video using the <video> tag for browsers that support HTML 5 and fallback on Flash for those that don't.

Something like this would do the trick:

```
<video src="video.mp4" controls>
<object data="player.swf" type="application/x-shockwave-flash">
<param value="player.swf" name="movie"/>
...etc...
</object>
</video>
```

Obviously, all we've really done is wrap the same old <object> and <embed> tags with the new<video> tag — hardly a great leap for the web.

How about we get rid of the fallback code, keep our HTML limited to the video tag and use a little JavaScript to handle the Flash embedding behind the scenes?

```
<video controls>
<source src="video.m4v" type="video/mp4" /> <!— MPEG4 for Safari —>
<source src="video.ogg" type="video/ogg" /> <!— Ogg Theora for Firefox
3.1b2 —>
</video>
```

Sjokvist's Flash solution requires a little JavaScript to sniff out the browsers capabilities and then offer Flash if the browser can't understand HTML5 (note that the code uses the swfobject library to handle the actual embed).

Notes We prefer this method since it keeps the actual HTML code cleaner and when video tag support is ubiquitous, all you need to do is drop the JavaScript. There's no requirement to rewrite your actual pages.

Another possible solution would be to simply load the MP4 movie into a Flash container file. As of Flash Player 10, Flash supports dynamically loaded MP4 files, so all you would need is to use Sjokvist's JavaScript detection code, but rather than feeding your player SWF a separate .flv video file, you could just load the same MP4 file.

If you need a refresher course on how to dynamically load videos into a Flash file, check out this Stack Overflow page which has a quick overview and some basic sample code.

Using that scenario, you've got a solution where every visitor can see your video and you only need to offer two actual files: Ogg Theora for Firefox and other browsers which support it, and an MP4 for everyone else.

2.4.2 Using the <audio> tag

The audio tag is more or less a duplicate of the video tag. The same codec limitations apply — Mozilla only supports Ogg Vorbis files, while Safari can handle pretty much anything QuickTime can.

Notes

```
The code looks very similar to <video>:
<audio src="/music/myaudio.ogg" autoplay>
Sorry, your browser does not support the <audio> element.
</audio>
```

And as with the <video> tag, the same Flash-based workarounds would give you near universal support for today's crop of browsers.

2.5 Creating List

HTML offers authors several mechanisms for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

- Unordered information
- Ordered information
- Definitions

The previous list, for example, is an unordered list, created with the UL element.

Unordered List

An unordered list is a list of items. The list items are marked with bullets (typically small black circles).

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
Output
```

- Coffee
- Milk

Notes Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Example:

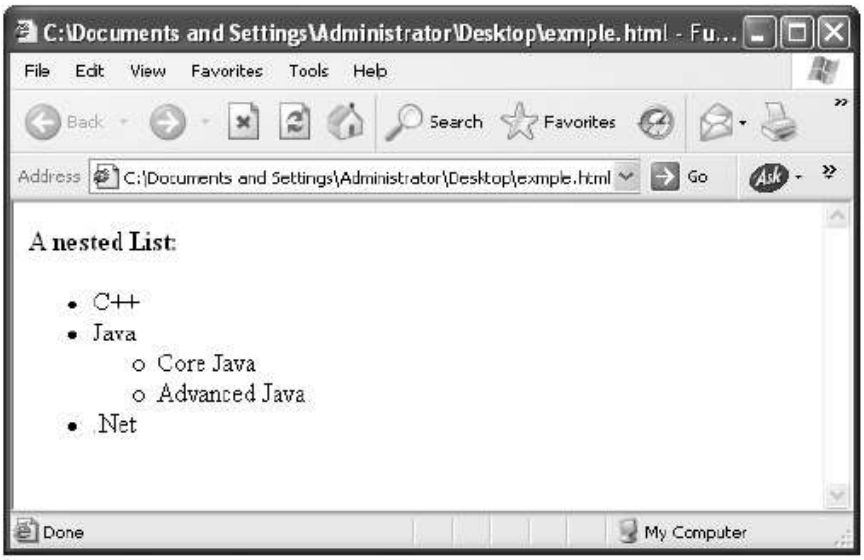
```
<html>
<body>
<h4>A nested List:</h4>
<ul>
   <li>C++</li>
   <li>Java
     <ul>
      <li>Core Java</li>
```

Notes

```
      <li>Advanced  Java</li>
     </ul>
   </li>
   <li>.Net</li>
</ul>
</body>
</html>
```

The output is shown in Figure 2.3.

Figure 2.3



Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers.

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

Here is how it looks in a browser:

1. Coffee
2. Milk

Notes Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Notes

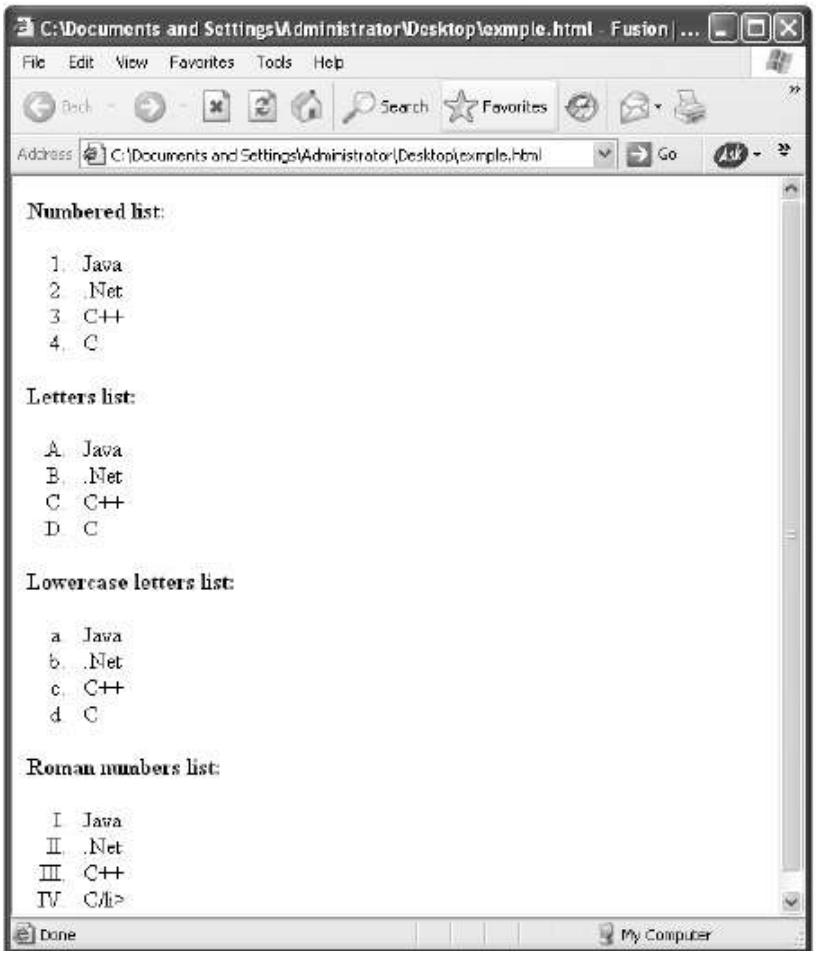
Example:

```
<html>
<body>
<h4>Numbered list:</h4>
<ol>
  <li>Java</li>
  <li>.Net</li>
  <li>C++</li>
  <li>C</li>
</ol>
<h4>Letters list:</h4>
<ol type="A">
  <li>Java</li>
  <li>.Net</li>
  <li>C++</li>
  <li>C</li>
</ol>
<h4>Lowercase letters list:</h4>
<ol type="a">
  <li>Java</li>
  <li>.Net</li>
  <li>C++</li>
  <li>C</li>
</ol>
<h4>Roman numbers list:</h4>
<ol type="I">
  <li>Java</li>
  <li>.Net</li>
  <li>C++</li>
  <li>C/li>
</ol>
<h4>Lowercase Roman numbers list:</h4>
<ol type="i">
  <li>Java</li>
  <li>.Net</li>
  <li>C++</li>
  <li>C/li>
</ol>
</body>
</html>
```

Notes

The output will be shown in Figure 2.4.

Figure 2.4



Definition Lists

A definition list is not a list of single items. It is a list of items (terms), with a description of each item (term).

A definition list starts with a <dl> tag (definition list).

Each term starts with a <dt> tag (definition term).

Each description starts with a <dd> tag (definition description).

Example:

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

Notes

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

The output is shown in Figure 2.5:

Figure 2.5

Notes Inside the <dd> tag you can put paragraphs, line breaks, images, links, other lists, etc.

Self Assessment

Fill in the blanks:

9. The <video> tag also has a whole host of events you can hook into with ……
10. An unordered list starts with the ……. tag
11. An ordered list starts with the <ol> tag. Each list item starts with the …… tag.
12. A definition list is not a list of ……. items.

2.6 Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

Example: `Tables in html.`

```
<html>
```

Notes

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
</html>
```

Output

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Tables and the Border Attribute

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show.

To display a table with borders, you will have to use the border attribute.

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

Headings in a Table

Headings in a table are defined with the <th> tag.

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
```

Notes

```
</tr>
</table>
```

Output

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
```

Output

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Table Tags

Table 2.1 is showing the table tags:

Table 2.1

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

Notes

Example:

```
<html>
<body>
<table border="1">
<tr>
  <td>
    <p>This is a paragraph</p>
    <p>This is another paragraph</p>
  </td>
  <td>This cell contains a table:
    <table border="1">
   <tr>
      <td>A</td>
      <td>B</td>
   </tr>
   <tr>
      <td>C</td>
      <td>D</td>
   </tr>
   </table>
  </td>
</tr>
<tr>
  <td>This cell contains a list
   <ul>
     <li>apples</li>
     <li>bananas</li>
     <li>pineapples</li>
   </ul>
  </td>
  <td>HELLO</td>
</tr>
</table>
</body>
</html>
```

Notes

Result: The result for the above html code is shown below:

This is a paragraph This is another paragraph	This cell contains a table: A B C D
This cell contains a list • apples • bananas • pineapples	HELLO

2.7 Forms

HTML fill-out forms can be used for questionnaires, hotel reservations, order forms, data entry and a wide variety of other applications. The form is specified as part of an HTML document. The user fills in the form and then submits it. The user agent then sends the form's contents as designated by the FORM element.

Notes Typically, this is to an HTTP server, but you can also e-mail form contents for asynchronous processing.

Forms are created by placing input fields within paragraphs, preformatted text, lists and tables. This gives considerable flexibility in designing the layout of forms.

HTML 3.0 supports the following kinds of fields:

- Simple text fields
- Multi-line text fields
- Radio buttons
- Checkboxes
- Range controls (sliders, or knobs)
- Single/multiple choice menus
- Scribble on image
- File widgets for attaching files to forms.
- Submit buttons for sending form contents
- Reset buttons for resetting fields to their initial values
- Hidden fields for book keeping information

It is expected that future revisions to HTML will add support for audio fields, multi-row entry of database tables, and extending multi-line text fields to support a range of other data types, in

Notes

addition to plain text. Client-side scripts will provide the means to constrain field values and to add new field types.

Exampel of a form

This fictitious example is a questionnaire. It uses the INPUT element for simple text fields, radio buttons, checkboxes, and the submit and reset buttons. The TEXTAREA field is used for a multi-line text entry field. The form fields are laid out with several paragraph elements and an unordered list. Notice the use of the NAME attribute to name each field:

```
<html>
<TITLE>Sample Questionaire</TITLE>
<H1>Sample Questionaire</H1>
<P>Please fill out this questionnaire:
<FORM METHOD=post ACTION="http://www.hal.com/sample">
<P>Your name: <input name="name" size="48">
<P><input name="male" type=radio> Male
<P><input name="female" type=radio>Female
   Number in family: <input name="family" type=int>
   <P>Cities in which you maintain a residence:
<UL PLAIN>
<LI><input name="city" type=checkbox value="delhi"> delhi
<LI><input name="city" type=checkbox value="ghaziabad"> ghaziabad
<LI>Others <textarea name="other" cols=48 rows=4></textarea>
</UL>
<P>Nickname: <INPUT NAME="nickname" size ="42">
<P>Thank you for responding to this questionaire.
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</FORM>
</html>
```

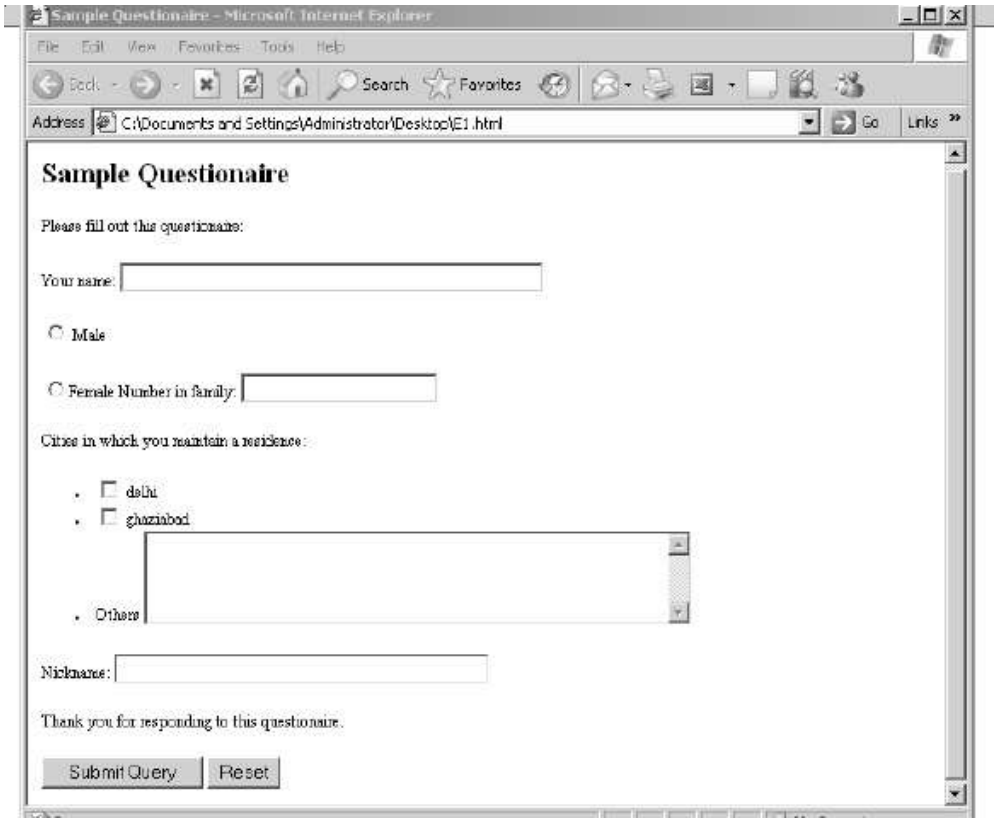
The output is shown in Figure 2.6.

Every form must be enclosed within a FORM element. There can be several forms in a single document, but the FORM element can't be nested. The browser is responsible for handling the input focus, i.e., which field will currently get keyboard input. Many platforms have existing conventions for forms, for example, using Tab and Shift-Tab to move the keyboard focus forwards and backwards between fields, and using the Enter (aka Return) key to submit the form.

This standard defines and requires support for the HTTP access protocol only. Under any protocol, the submitted contents of the form logically consist of a list of name/value pairs where the names are given by the NAME attributes of the various fields in the FORM. Each field will normally be given a distinct name. Several radio buttons can share the same name, as this is how you specify that they belong to the same control group - at any time, only one button in the group can be selected.

Notes

Figure 2.6



Notes The contents list of name/value pairs excludes unselected radio buttons and checkboxes. In general, any field with a null value can be omitted from the contents list.

Client-side Scripts and fill-out Forms

HTML 3.0 doesn't provide direct support for constraining the values entered into text fields, or for derived fields whose values are calculated from the values of other fields. Rather than extending the markup to support these features, HTML 3.0 provides a means for associating the form with a script. Support for scripts is not required, however, and the HTML 3.0 specification doesn't cover the scripting languages or the details of their interface with the user agent.

The SCRIPT attribute of the FORM element specifies the script via a URI. The user agent downloads the script and interprets it locally. Scripts handle a variety of messages for individual fields and the form as a whole. These messages correspond to events such as:

- Enter/Leave Form (for initialization and clean up)
- When a field gains or loses the input focus
- Mouse clicks and drags over a field
- Keyboard events

Notes

Scripts can examine and set properties of fields. They can also examine a small set of standard properties of the user agent, for instance the user's name, the time of day, the type of user agent, and so on.

Scripts can't do anything that might jeopardize the user or the host machine. Scripts can't send messages over the network, or read or write files. The library calls that are allowed are restricted to a very small and well defined set. These precautions are necessary for untrusted scripts.

Task It is envisaged that script interpreters will offer a much wider application programming interface to trusted scripts, as determined on the basis of a digital signature by a trusted third party. Analyze in group of four.

Permitted Attributes for Form

- ACTION

  The ACTION attribute is a URL specifying the location to which the contents of the form is submitted to elicit a response. If the ACTION is missing, the URL for the document itself is assumed. The way data is submitted varies with the access protocol of the URL, and with the values of the METHOD and ENCTYPE attributes.

- METHOD

  This specifies variations in the protocol used to send the form contents. It is currently restricted to GET (the default) or POST. The attribute was introduced to inform user agents which HTTP methods the server supports.

- ENCTYPE

  This attribute specifies the MIME content type to be used to encode the form contents. It defaults to the string: "application/x-www-form-urlencoded"

- SCRIPT

  This can be used to give a URI for a script. The scripting language and the interface with the user agent is not part of the HTML 3.0 specification.

Self Assessment

Fill in the blanks:

13. If you do not specify a border attribute the table will be displayed without any……...

14. Scripts can examine and set properties of……..

2.8 Frames

Frames divide a browser window into two or more document windows, each displaying a different document, or a different part of the same document. Frames in an HTML document can cause a web page to appear to be divided into multiple, scrollable regions. For each frame, you can assign a name, a source document locator, dimensions, border alignment and decorations, scroll and resize behaviors, loading and unloading behavior, file and topic maps, and style sheets.

Notes

- ***Names:*** You can place an anchor in any frame, link to any addressable object, and place the object into any named frame.
- ***Source Document Locator:*** You can use whatever addressing schemes your user agent supports, including URLs and filenames.
- ***Dimensions:*** You can rigidly or flexibly layout a two-dimensional grid of rectangular blocks.
- ***Border Alignment and decoration:*** You can adjust the position of the left and right margins, the top and bottom margins, and the alignment of the frame. You can also make the borders of a frame invisible.
- ***Scrolling:*** Frames can have scrollbars, no scrollbars, or you can let the browser turn them on if the document is larger than the current horizontal or vertical size of the frame.
- ***Resizable:*** Frames are normally resizable in the browser, but that can be disabled so the frame may not be resized at the user agent.
- ***Loading and Unloading:*** You can provide a script to be run when the user agent finishes loading all frames or when the user exit the document.
- ***File and Topic Maps: You*** can place a file or topic navigator into a frame.

Notes The navigator might be a collapsible listing of file system, a listing of document headings, thumbnails of images in a document, or an index of any element type.

These properties make possible:

- Static frames: Elements that a user should always see, such as button bars, logos, copyright notices, and title graphics, can be placed in individual frames that are locked into place on the user agent window.
- Live frames: Documents, icons, interactive forms, videos, multimedia, topic maps, and anything else that can react to user input or programmed activity.
- As a user navigates a site in "live" frames, the contents of static frames remain fixed, even though adjoining frames redraw.
- Functional tables of contents: A frame can contain interactive tables of contents (TOCs) with links that, when clicked, display results in an adjoining frame. These TOCs can be static or interactive with collapsible lists, graphical maps of document structure, or displays of file and link architectures.
- Single-page query and answer displays: Frames designed side-by-side permit queries to be posed and answered on the same page, with one frame holding the query form, and the other presenting the results.

Task A frame can contain interactive tables of contents (TOCs) with links that, when clicked, display results in an adjoining frame. Explain with suitable example

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

Notes

```
</HEAD>
        <FRAMESET rows="33%,33%,33%">
                <FRAMESET cols="50%,50%">
                        <FRAME name="frame1">
                       <FRAME name="frame2">
        </FRAMESET>
        <FRAME name="frame3">
<FRAME name="frame4">
</FRAMESET>
<BODY>
...contents to display in non-frame-capable user agent...
</BODY>
```

</HTML>

Try to remember the following basic frameset concepts:

- The opening FRAMESET tag defines the actual size and layout of the frames.
- Using the ROWS and COLS attributes of the FRAMESET tag, you define frames, then fill them with the FRAME element, or subdivide them with nested FRAMESET elements.
- The number of entries in ROWS or COLS attribute values determine how many frames are displayed and what size they are. If you have both ROWS and COLS, multiply the number of entries in each attribute value to calculate the total number of frames (i.e., three entries in the ROWS attribute and four entries in the COLS attribute create a grid of twelve frames).
- The FRAME elements fill the frames defined by FRAMESET, and are applied in the frame set in a left-to-right, top-to-bottom order.
- FRAME elements which include the SRC attribute, define the contents of each frame you create.
- Frames are always rectangular.

The FRAMESET Element

A frameset divides the browser window into rectangular regions. Each such region can be:

- A frame, which displays one document. A frame is represented by a FRAME element.
- Another frame set, which is itself further divided into frames.

Example: A frame set can contain a frame, plus another frame set containing two frames, resulting in three frames in all.

An HTML document that contains frames basically replaces the BODY element with the FRAMESET element.

- ***The Rows and Cols Attributes***

  The attributes of the FRAMESET element are COLS (columns) and ROWS. They determine how many frames the frame set is divided into. These attributes may be blank, or may consist of a list of one or more values separated by commas or spaces. Each such value

Notes

determines the width (for columns) and height (for rows) of the regions; the number of width and height values supplied determines how many rows and columns, respectively, are created. The default for each is one. For example, if you have:

```
<FRAMESET  cols="20%,30%,50%">
```

In which there is no ROWS value, the frame set is divided vertically into three regions: the first region's width is 20% of the current frame set (or browser window if this frame set is at the top level), the second region's width is 30%, and the third region's width is 50%. When there is only one frame set in the document, these widths apply to the entire browser window. Similarly, when there is a ROWS value but no COLS value, the frame set is divided horizontally into regions. When values are supplied for both attributes, the frame set is divided into a grid of rows and columns.

The ROWS and COLS attributes take comma-separateds lists of values. These values can be absolute pixel values, percentage values between 1 and 100, or relative scaling values. The number of rows and columns is implicit in the number of values in the respective list. Since the total height of all the rows must equal the height of the window, row heights might be normalized to achieve this.

Notes If the rows (or cols) attribute values are unspecified, then the number of rows (or columns) is assumed to be one, and it may be arbitrarily sized to fit.

- ***Syntax of Value List***
  - *value*

    A simple numeric value is assumed to be a fixed size in pixels. The result of this value varies with the size of a viewer's window. Fixed pixel values are usually used with one or more of the relative size values described below. You might use a fixed value if you want a graphic, such as an image map, to fill an entire frame and you want to ensure that the frame is big enough to display the entire image. User agents can be expected to over ride a specified pixel value to ensure that the total proportions of a frame are 100% of the width and height of a user's window.
  - *value%*

    This is a simple percentage value between 1 and 100. If the total is greater than 100, all percentages are scaled down. If the total is less than 100, and relative-sized frames exist, extra space is given to them. If there are no relative-sized frames, all percentages are scaled up to match a total of 100%. For example, suppose you assign the ROWS attribute a value of "50%, 50%, 50%". Each entry is 50%, which is one third of the sum of all the entries (150%), so the browser assigns the frame sizes proportionately, giving each frame one third of the browser height.
  - *value**

    The value on this field is optional. A single '*' character is a "relative-sized" frame and is interpreted as a request to give the frame all remaining space. If multiple relative-sized frames are specified, the remaining space is divided evenly among them. If there is a value in front of the '*', that frame gets that much more relative space. "2*,*" would give 2/3 of the space to the first frame, and 1/3 to the second.
- ***Example:*** Setting Frame width and height

  There are three ways to specify the width or height of a frame
  - As a percentage of the area allotted to the parent frame set

Notes

- As an absolute (specific) number of screen pixels (e.g., 250)
- As a 'relative size'

Example for 3 rows, the first and the last being smaller than the center row:

```
<FRAMESET rows="20%,60%,20%">
```

Example for 3 rows, the first and the last being fixed height, with the remaining space assigned to the middle row:

```
<FRAMESET rows="100,*,100">
```

A 'relative size' is specified with an asterisk, e.g., '1*', '2*', '3*' ('1*' can also be written simply as '*'). This is interpreted as follows: after all widths (or heights) specified as percentages or absolute amounts have been allocated to the corresponding frames, the remaining space is allocated to frames whose widths (or heights) have been specified as a relative size. The amount of space allocated to a frame is proportional to the number in front of the asterisk. For example:

```
<FRAMESET rows="30%,400,*,2*">
        <FRAME ... >
        <FRAME ... >
        <FRAME ... >
        <FRAME ... >
</FRAMESET>
```

Suppose the browser window is currently 1000 pixels high. The first frame gets 30% of the total height, that is, 300 pixels; the second frame gets 400 pixels, since an absolute amount was specified. This leaves 300 pixels to be divided between the other two frames. The fourth frame's height is specified as '2*', so it is twice as high as the third frame, whose height is only '*' (1*). Therefore the third frame is 100 pixels high and the fourth is 200 pixels high.

Task Explain the web development in India?

The FRAME Element

The FRAME element defines a single frame in a frameset. It has 7 possible attributes: SRC, NAME, FRAMEBORDER, MARGINWIDTH, MARGINHEIGHT, SCROLLING, and NORESIZE. The FRAME tag is not a container so it has no matching end tag.

The attributes of the FRAME element type are as follows:

- src="address"

  Specifies the address of the document to be displayed in the frame. When omitted the frame is displayed as a blank space.

- name="window_name"

  The NAME attribute is used to assign a name to a frame so it can be targeted by links in other documents (These are usually from other frames in the same document.) The NAME attribute is optional; by default all windows are unnamed. Names must begin with an alphabetic character. Named frames can have their window contents targeted with the TARGET attribute.

Notes

- frameborder="1|0"

  FRAMEBORDER is a boolean which triggers the display of frame separators around the frame. When set to "1" a separator is drawn on every side next to another frame. When set to "0" the decision to draw separators is left to surrounding frames. Indeed in this case separators might still be drawn because FRAMEBORDER is set "1" on adjacent frames. Default is "1".

- marginwidth="value"

  The MARGINWIDTH attribute is used when the document author wants some control of the margins for this frame. If specified, the value for MARGINWIDTH is in pixels. Margins can not be less than one; this insures that frame objects won't touch frame edges, and can't be specified in a way that leaves no space for the document contents. The MARGINWIDTH attribute is optional; by default, all frames default to letting the browser decide on an appropriate margin width.

- marginheight="value"

  The MARGINHEIGHT attribute is just like MARGINWIDTH above, except it controls the upper and lower margins instead of the left and right margins.

- noresize

  The NORESIZE attribute doesn't require a value. It can simply be used as a flag to indicates that the frame is not resizable by the user. Users typically resize frames by dragging a frame edge to a new position.

> *Notes* Note that if any frame adjacent to an edge is not resizable, that entire edge is restricted from moving.

  This affects the resizability of other frames. The NORESIZE attribute is optional; by default all frames are resizable.

- scrolling="yes|no|auto"

  The SCROLLING attribute indicates whether the frame should have scrollbars or not. "yes" results in scrollbars always being visible on that frame. "no" results in scrollbars never being visible. "auto" instructs the browser to decide whether scrollbars are needed, and to place them where necessary. That is, with "auto", the frame has scrollbars only if the document is larger than the current size of the frame. The SCROLLING attribute is optional; the default value is "auto."

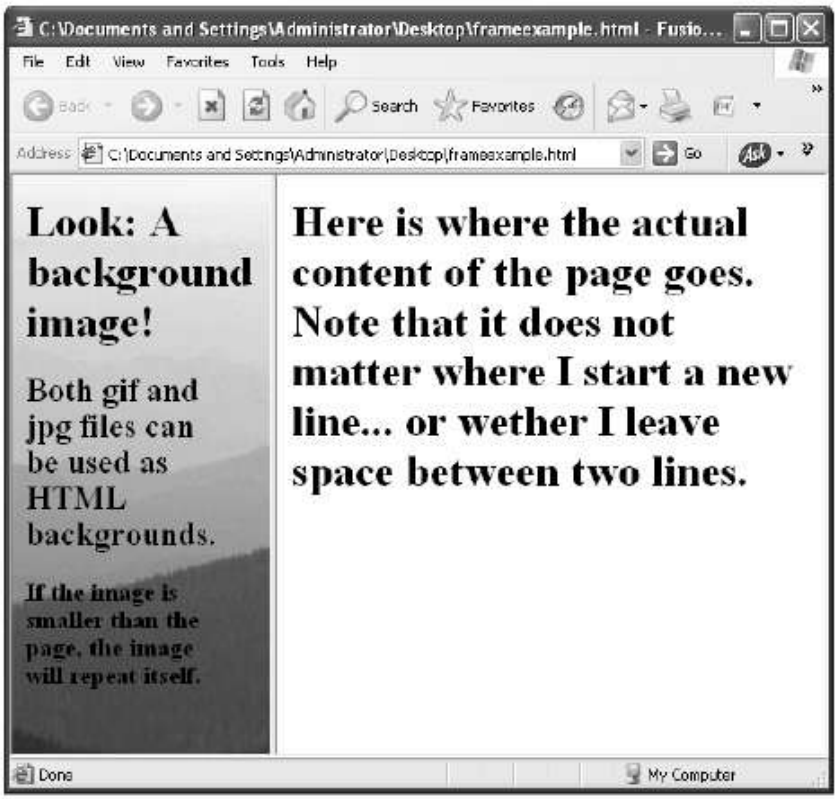
Example:

```
<frameset cols="25%,75%">
        <frame src="frame_a.htm">
        <frame src="frame_b.htm">
</frameset>
```

The output will be shown in Figure 2.7.

Notes

Figure 2.7



Example:

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

Result:

Frame A	Frame B	Frame C

Notes

Self Assessment

Fill in the blanks:

15. Frames divide a browser window into two or more ……

16. Forms are created by placing input fields within paragraphs, ……, lists and tables.

2.9 Using Multiple Windows in Web Pages

You can multiply your surfing fun by browsing more than one web page at the same time.

If you are an Internet power surfer, then you might find it useful to open more than one web page at once so you can perform different actions on different pages and switch between them as you wish. For example, you can search for different things in different pages, search in one page and surf in another, explore one site on one page and a related site on another page, track the news in one page and type in a chat room in another, or simply switch to a more responsive page while waiting for a slow page to load.

There are two effective methods of managing multiple pages: creating *new windows* which works with any browser; and tabbed browsing that provides significantly improved browser usability for those browsers that support it.

New window. With almost any browser you can open a new window with the command <ctrl>-n or the menu item "File / New Window". You can then quickly stagger new pages for easy access as you open them in two steps:

- Top left. Line up the top left corner of the new window with the previous windows so they more or less line up at the same spot.
- Bottom right. Pull the bottom right corner of the new window so that it is staggered above the previous windows as shown in the picture below.

This arrangement ensures that no window can ever get completely covered by the other windows, and you will always be able to select any window at any time in one action by simply clicking on the bottom right corner of the window you wish to switch to.

With several windows open you can perform advanced web navigation, such as searching for different things in different windows, search in one window and surf in another, use one window to explore one site and another site in a second window, or use one window to track the news while you type in a chat room in another window.

Notes

Caselet

PDF, HTML Files

What do terms such as pdf file, html, etc., mean? How does one convert these into text files? Please explain.

A PDF file is a Portable Document Format file. It can be viewed from any PDF reader application such as Adobe Acrobat Reader.

Any document that has to be distributed across multiple platforms will be made a PDF file and distributed. The advantage of this is that any PDF reader is sufficient to view and print the document and it is freely available.

If the document is sent in any other format other than the relevant application, you should have the fonts installed. Most manuals are sent as PDF files.

An HTML file is a Hyper Text Markup Language file. It can be viewed from any Web browser such as Internet Explorer, Netscape, etc. HTML document is used in the Internet.

You can open the HTML file in a browser and from the browser it can be saved as Text file.

For some pages, the file of text format is not the same as it appears in the browser. Similarly a PDF file can be saved as Text Format from a PDF application such as Adobe Acrobat. Acrobat Reader 7.0.7 full version can be downloaded from this URL: http://www.softwarepatch.com

/graphics/acrobatreader7.html

Only a PDF reader is freely available on the Internet.

If you want to create a PDF document, you will need to buy a licensed application such as Adobe Acrobat or any third party applications.

2.10 Summary

- HTML (Hyper-Text Markup Language) is composed of tags.
- Attributes can be used to change a tag's properties.
- Tables are defined with the <table> tag.
- Images can be used to make your web pages distinctive, and can greatly help to get your message across.
- HTML fill-out forms can be used for questionnaires, hotel reservations, order forms, data entry and a wide variety of other applications.
- HTML 3.0 doesn't provide direct support for constraining the values entered into text fields, or for derived fields whose values are calculated from the values of other fields.
- Frames divide a browser window into two or more document windows, each displaying a different document, or a different part of the same document.

2.11 Keywords

CSS: Cascading Style Sheets

HTML: Hyper-Text Markup Language

Notes

2.12 Review Questions

1. What is HTML? How many types of tags are used in HTML? Discuss in detail.
2. Write the HTML code for the following:

Name		Marks	
	CO	IWT	DAA
AAA	60	70	78
BBB	55	66	60

3. After the DOCTYPE declaration come the "html" tags. Explain with example.
4. Explain why the attributes can be used to change a tag's properties.
5. Tags are instructions that are embedded directly into the text of a HTML document. Discuss.
6. The <img> tag is empty, which means that it contains attributes only and it has no closing tag. Give reasons and examples.
7. Is audio tag is more or less a duplicate of the video tag? Yes or no? Explain briefly.
8. HTML offers authors several mechanisms for specifying lists of information. What do you mean by this statement? Give examples to support your answer.
9. The form is specified as part of an HTML document. Discuss the relevance of Forms in HTML.
10. Frames in an HTML document can cause a web page to appear to be divided into multiple, scrollable regions. Explain all this with suitable examples.

Answers: Self Assessment

1.	head	2.	title
3.	displayed	4.	rgb
5.	companion	6.	src
7.	URL	8.	alternate text
9.	JavaScript	10.	
11.		12.	single
13.	borders	14.	fields
15.	document windows	16.	preformatted text

2.13 Further Readings

Books

David Mercer, *HTML,* McGraw-Hill Professional

Willard, *HTML: A Beginner'S Guide,* Tata McGraw-Hill Education

Chuck Musciano, Bill Kennedy, *HTML & XHTML: the definitive guide,* O'Reilly Media, Inc.

Notes

Gary B. Shelly, Denise M. Woods, William J. Dorin, *HTML, XHTML, and CSS: Comprehensive,* Cengage Learning

Online links

http://www.w3schools.com/html/

http://www.w3schools.com/html/html_intro.asp

Notes

Unit 3: Text Formatting

CONTENTS

Objectives

Introduction

3.1 Character Elements

3.2 Look Elements

3.3 Summary

3.4 Keywords

3.5 Review Questions

3.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Recognize the character elements
- Describe the looking elements

Introduction

Generally, text is unformatted in your Web documents. However, to emphasize specific letters, words, or phrases which reflects the characteristics about the text. Also, you can use some text formatting options including boldfacing and italicizing. Depending on your browser, you may also be able to add underlining to emphasize characters. There are two major classes of text markup:

(a) Character Elements (*Meaning*)

(b) Look Elements (*Formatting*)

3.1 Character Elements

The character elements describe the type of the text formatting. It is also the treat as *Logical codes* which shows the meaning of the text. For example, if you want to display some programming code, with the help of <CODE> tag can be display. The same font also used with <PRE>; however, <PRE> is for formatted text while the browser will format the text associated with the <CODE>. These elements also called as *Phrase elements*. These elements require start and end tags. There are no required HTML attributes for these elements.

<EM> Element

The <EM> element is an HTML tag that is used to specify emphasized text within an HTML document. The <EM> basic emphasis typically rendered, in an italic font.

Notes

Example:

```
<BODY>
The witness was <EM>not even there</EM> at the time.
</BODY>
…..is rendered similar to…..
```

<STRONG> Element

The <STRONG> element indicates text, which should be more forceful than surrounding text. The <STRONG> element emphasizes, usually bold.

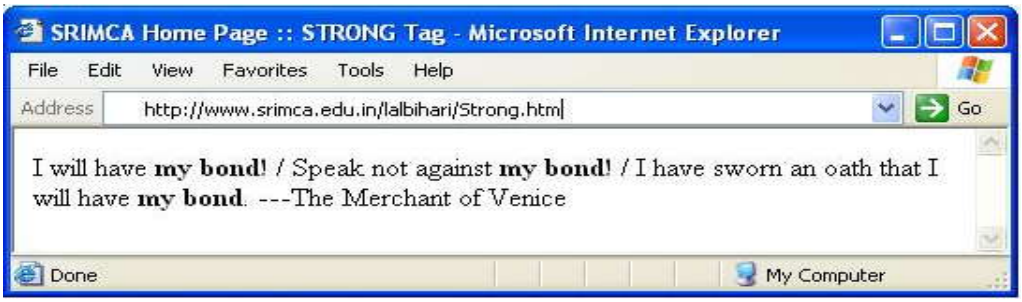
Example:

```
<BODY>
```

We will have <STRONG> my bond! </STRONG>/ Speak not against <STRONG> my bond! </STRONG>/ I have sworn an oath that I will have <STRONG> my bond</STRONG>. —The Merchant of Venice

```
</BODY>
```

…..is rendered similar to…..



Notes What's the difference between <EM> (emphasis) and <STRONG>? It's a matter of degree: <STRONG> is like <EM> only more so.

<SAMP> Element

The <SAMP> element indicates text, which is an example of something. It is also commonly used to represent some key piece of text that has special meaning.

Notes

Example:

```
<BODY>
```

The process of writing Java file names must be the file extension is <SAMP> .java <SAMP>.

```
</BODY>
```

…..is rendered similar to…..

Notes The <SAMP> element is rendered in a fixed width font. The <SAMP> element has a similar purpose to <CODE> element.

<CODE> Element

The <CODE> element indicates text that is the code for a program. <CODE> is rendered in a fixed width font.

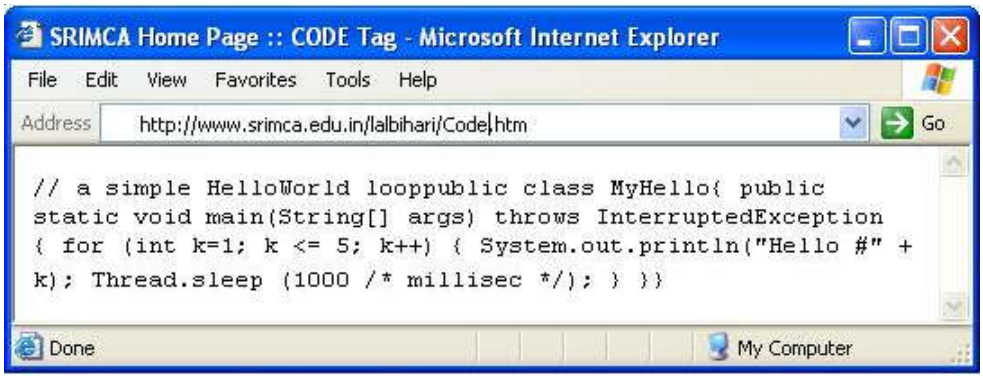
Example:

```
<BODY>
<code>
// a simple HelloWorld loop
public class MyHello
{
 public static void main(String[] args) throws InterruptedException
  {
     for (int k=1; k <= 5; k++)
    {
        System.out.println("Hello #" + k);
        Thread.sleep (1000 /* millisec */);
    }
  }
}
</code>
```

Notes

```
</BODY>
```

…..is rendered similar to…..



Notes For lengthy code examples, most programmers prefer <PRE ...>, which is much better for showing spacing and indentation.

<ADDRESS> Element

The <ADDRESS> element denotes contact information for the author or an organization of the website.

Example:

```
<BODY>
```

Site is developed at

```
<ADDRESS>
```

Shrimad Rajchandra Institute of Management & Computer Application

Gopal Vidyanagar

Bardoli-Mahuva Road, Tarsadi- 394 350

Surat, Gujarat, INDIA

```
</ADDRESS>
</BODY>
```

…..is rendered similar to…..

Notes

<CITE> Element

The <CITE> element indicates a reference to work, such as a book, report or web site. A citation (typically rendered in italics).

Example:

```
<BODY>
```

Source:

integral - verb controlling "<CITE> ... Hicks and Streeten (1979, p. 568) identify and review four different approaches... </CITE>"

```
</BODY>
```

…..is rendered similar to…..

Notes The <CITE> element is also commonly used to indicate ownership and authorship of a web page, usually at the bottom of the page under a <HR ...>

<INS>, <DEL> Element

The <DEL> element indicates that the text has been deleted from the current revision of the document. The <INS> element indicates that the text has been inserted into the document since the last revision. Both of these tags are currently only supported by MSIE.

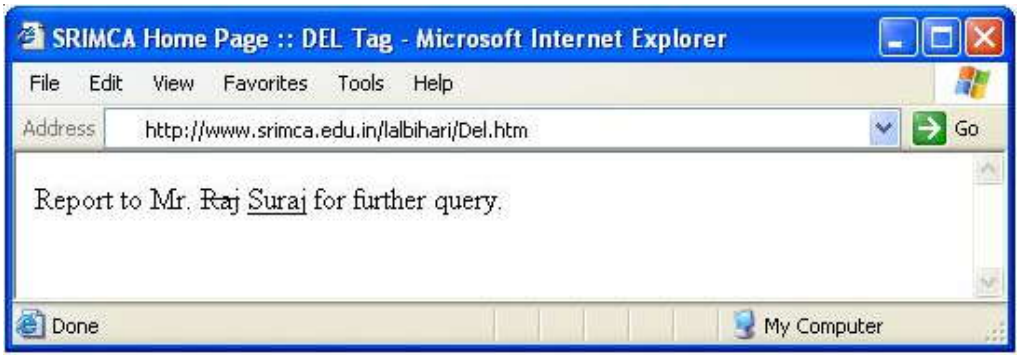
Caution The problem with <DEL> and <INS> elements are that they don't degrade properly. Browsers that don't know these tags will display the contents of both of them as normal, giving an incorrect representation of the intended meaning.

Example:

```
<BODY>
Report to Mr.
<DEL>Raj</DEL>
<INS>Suraj</INS>
for further query.
</BODY>
```

…..is rendered similar to…..

Notes



Notes Don't use these tags unless you are sure your audience's browsers support them.

<KBD> Element

The <KBD> element indicates characters that should be typed in at a keyboard. <KBD> is rendered in a fixed width font. Typically this is used in instruction manuals.

Example:

```
<BODY>
```

Finally, type <KBD>logout</KBD> and press the return key.

```
</BODY>
```

…..is rendered similar to…..

Notes Use the <KBD> element for fixed strings only. To indicate input which varies from one case to another, use the <VAR> element.

<VAR> Element

To indicate that a piece of text (typically, a word) is a variable, a "placeholder", i.e., a generic notation to be replaced by different actual expressions.

Notes

Example:

```
<BODY>
```

In the simplest case, the command for deleting a file in Unix is

```
<KBD>rm</KBD>  <VAR>filename</VAR>
</BODY>
```

…..is rendered similar to…..

<DFN> Element

The <DFN> element can be regarded as a special kind of emphasis, too, but logically it indicates that a term is used in a context where it is defined.

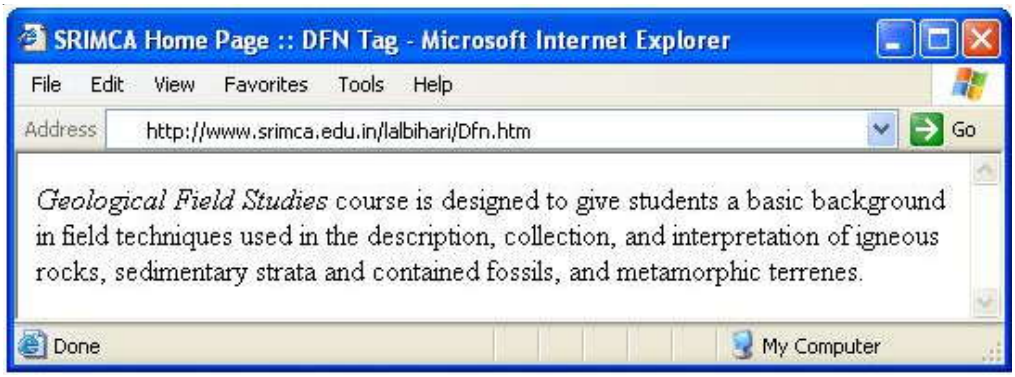
Example:

```
<BODY>
```

<DFN> Geological Field Studies </DFN> course is designed to give students a basic background in field techniques used in the description, collection, and interpretation of igneous rocks, sedimentary strata and contained fossils, and metamorphic terrenes.

```
</BODY>
```

…..is rendered similar to…..



Notes This is a very useful element in principle, but unfortunately, many browsers, including Netscape, do not effectively support it.

Notes

<PERSON> Element

The <PERSON> element is used around names of people mentioned in a document. It is typically displayed, just like normal text, but is used by automatic indexers.

Example:

```
<BODY>
```

<PERSON>Jane Doe</PERSON>, former President of United Corporation, was born in Hot Springs, Arizona.

```
</BODY>
```

…..is rendered similar to…..

<ABBREV> Element

The <ABBREV> element defines an abbreviation. It is typically displayed just like normal text, but is used by automatic indexers.

Example:

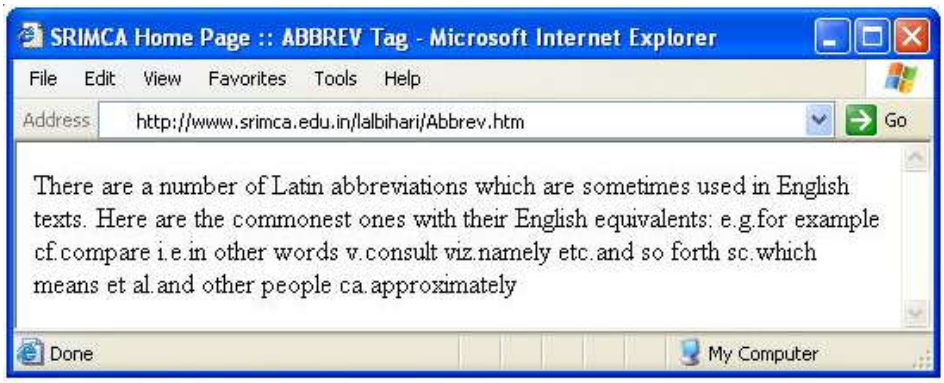
```
<BODY>
```

There are a number of Latin abbreviations, which are sometimes used, in English texts. Here are the commonest ones with their English equivalents:

```
<ABBREV>e.g.</ABBREV>for example
<ABBREV>cf.</ABBREV>compare
<ABBREV>i.e.</ABBREV>in other words
<ABBREV>v.</ABBREV>consult
<ABBREV>viz.</ABBREV>namely
<ABBREV>etc.</ABBREV>and so forth
<ABBREV>sc.</ABBREV>which means
<ABBREV>et al.</ABBREV>and other people
<ABBREV>ca.</ABBREV>approximately
</BODY>
```

…..is rendered similar to…..

Notes



<ACRONYM> Element

The <ACRONMY> element defines an acronym. It is typically displayed just like normal text, but is used by automatic indexers.

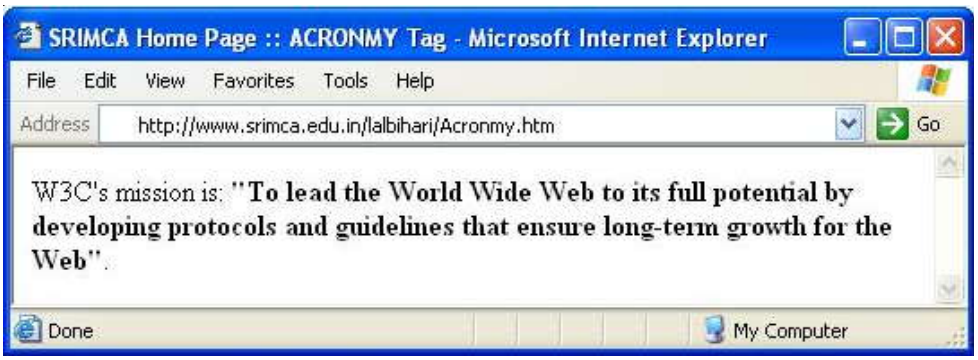
Example:

```
<BODY>
```

<ACRONMY>W3C</ACRONMY>'s mission is:<STRONG> "To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web"</STRONG>.

```
</BODY>
```

…..is rendered similar to…..



<AUTHOR> | <AU> Element

The <AUTHOR> | <AU> element defines text that names the author of a document. It is typically displayed just like normal text, but is used by automatic indexers.

Example:

```
<BODY>
```

E=MC<SUP>2: A Biography of the World's Famous Education, By <AUTHOR>David Bodanis</AUTHOR>

```
</BODY>
```

…..is rendered similar to…..

Notes

Task Analyze the advantages of author tag.

Self Assessment

Fill in the blanks:

1. The …….. element is an HTML tag that is used to specify emphasized text within an HTML document.
2. The ……. element indicates text, which should be more forceful than surrounding text.
3. The …….. element indicates text, which is an example of something.
4. The …….. element denotes contact information for the author or an organization of the website.
5. The …….. element indicates a reference to work, such as a book, report or website.
6. The ……. element indicates that the text has been deleted from the current revision of the document.
7. The …… element can be regarded as a special kind of emphasis, too, but logically it indicates that a term is used in a context where it is defined.
8. The …… element is used around names of people mentioned in a document.
9. The …… element defines an acronym.

3.2 Look Elements

The look elements deal with the specific text formatting. It is also the treat as Physical codes, which use for how the text is to be displayed and not it's meaning. These tags are useful when bold or idealize a phrase or text.

Note: The logical styles may not be distinct (i.e. different logical styles may be rendered in the same way). Also, some browsers do not support all physical styles. For example, Mosaic does NOT support underlined text.

<B> Element

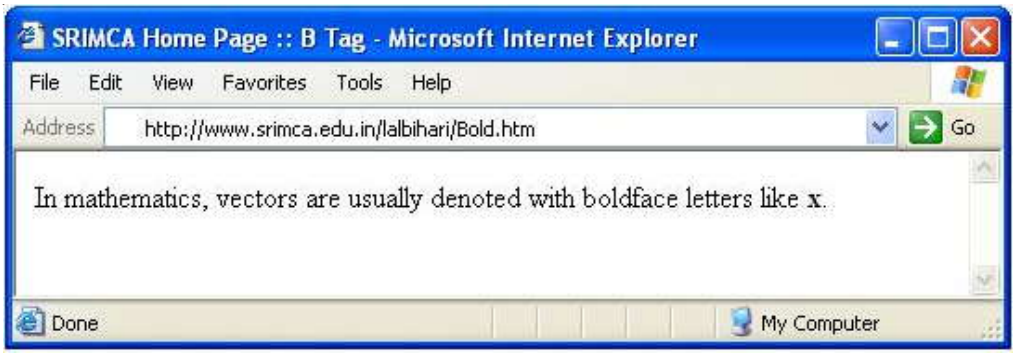
The <B> element defines text that should be shown in boldface. It can be nested with other logical or physical elements but some browsers will respect only the innermost tag.

Example:

```
<BODY>

<P>In mathematics, vectors are usually denoted with boldface letters like
<B>x</B>.

</P>

</BODY>
```

…..is rendered similar to…..

Notes



Notes Avoid using <B> element; use logical elements instead. In particular, for emphasis uses <EM> or <STRONG> elements.

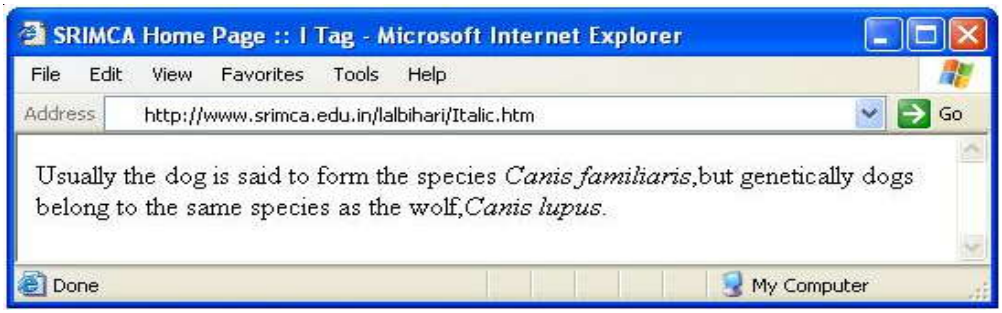
<I> Element

The <I> element defines text that should be shown in italics. It can be nested with other logical or physical elements but some browsers will respect only the innermost tag.

Example:

```
<BODY>
Usually the dog is said to form the species
<I>Canis familiaris</I>,
but genetically dogs belong to the same species as the wolf,
<I>Canis lupus</I>.
</BODY>
```

…..is rendered similar to…..



Notes However, don't overuse the <I> element. In particular, for emphasis use <EM> or <STRONG>, and for variables (placeholders) use <VAR>.

Notes

<U> Element

The <U> element defines text that should be shown with a line underneath it. It can be nested with other logical or physical elements but some browsers will respect only the innermost tag.

Example:

```
<BODY>
        Compare <U>underlined text</U> with normal text.
</BODY>
```

…..is rendered similar to…..

Notes Avoid using <U> element; use logical elements. For example, to emphasize use EM or STRONG. In HTML 4.0, the U element is deprecated .

<TT> Element

The <TT> element defines text that should be shown in a fixed width font. It can be nested with other logical or physical elements, but some browsers will respect only the innermost tag. (Typically rendered in monospaced font)

Example:

```
<BODY>
Compare <TT>monospaced font</TT> with normal font.
</BODY>
…..is rendered similar to…..
```

Notes

Notes Avoid using <TT> element; use logical elements instead, e.g. <CODE> or <SAMP>.

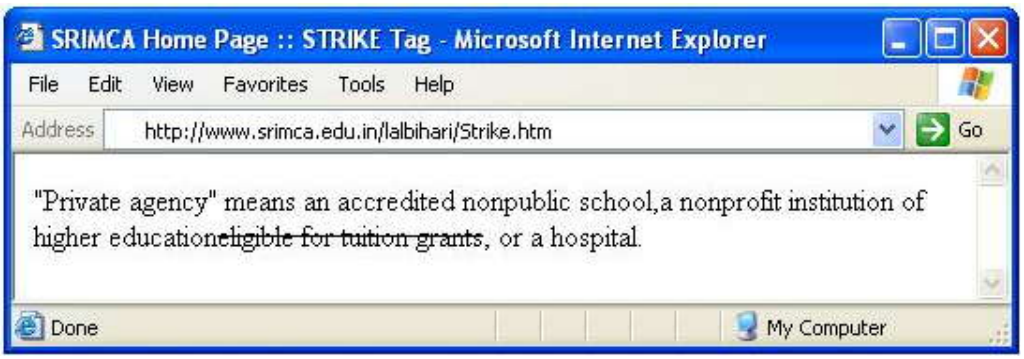
<STRIKE> Element

The <STRIKE> element defines text that should be shown with a horizontal line through it. It is defined as "font style element". Typically text is striked out to indicate that a text segment belongs to the original version of a text but has been deleted later.

Example:

```
<BODY>
"Private agency" means an accredited non-public school,
a nonprofit institution of higher education
<STRIKE>eligible for tuition grants</STRIKE>, or a hospital.
</BODY>
```

…..is rendered similar to…..



Notes If you use STRIKE in your document, it is advisable to include a note about its meaning.

<BIG> Element

The <BIG> text element defines text that should be displayed in a larger font than usual.

Example:

```
<BODY>
That was a <BIG>big</BIG> mistake!
</BODY>
```

…..is rendered similar to…..

Notes

Notes Avoid using <BIG> element; use logical elements instead. In particular, for emphasis use <EM> or <STRONG>.

<SMALL> Element

The <SMALL> text element defines text that should be displayed in a smaller font than usual.

Example:

```
<BODY>
<P>
This is normal text.
</P>
<P>
<SMALL>
This text will be presented in a smaller font, if possible.
</SMALL>
</P>
</BODY>
```

…..is rendered similar to…..

Notes There are no logical elements in current HTML standard; instead, the <FONT> element may provide more alternatives for specifying different font sizes.

Notes

<SUB> Element

The <SUB> element defines text that should be displayed slightly below the normal text level, often so that it the text is vertically centered with respect to normal text baseline, and possibly in smaller font.

Example:

```
<BODY>
Let us form the sum of all x<SUB>i</SUB>'s, ie
x<SUB>1</SUB> + x<SUB>2</SUB> + ... + x<SUB>n</SUB>.
</BODY>
```

…..is rendered similar to…..

Notes Since this element is new, support for it is not universal. Some browsers simply ignore it, displaying e.g., a₁ as a1. And naturally, text-only browsers cannot truly support <SUB>. Internet Explorer ignores <SUB> elements after nesting level of two.

<SUP> Element

The <SUP> element defines text that should be displayed slightly above the normal text level and possibly in smaller font.

Example:

```
<BODY>
The expression:
a<SUP>b<SUP>c</SUP></SUP>
means a<SUP>(b<SUP>c</SUP>)</SUP>
</BODY>
```

…..is rendered similar to…..

Notes

Notes Since this element is new, support for it is not universal. Some browsers simply ignore it, displaying e.g., a^b as ab. And naturally, text-only browsers cannot truly support <SUP>. Internet Explorer ignores <SUP> elements after nesting level of two.

Did u know? What is the difference between <SUB> element and <SUP> element?

This text contains _{subscript} text.

This text contains ^{superscript} text.

Self Assessment

Fill in the blanks:

10. The …… element defines text that should be shown in boldface.
11. The …… element defines text that should be shown in italics.
12. The …… element defines text that should be shown with a line underneath it.
13. The …… element defines text that should be shown in a fixed width font.
14. The …… element defines text that should be shown with a horizontal line through it.
15. The …… element defines text that should be displayed slightly above the normal text level and possibly in smaller font.

Caselet

CSS Group Sets up Arm to Service Indian Clients

The CSS Group has set up an information technology subsidiary, CSS Software India, for servicing Indian clients. The new company will operate out of 12 centres over the next six months.

The new company will offer specific software development and testing facilities and will target verticals such as retail, telecom, banking, financial services and insurance and manufacturing, said Mr Shiva Ramani, co-founder, CSS Group. The company is also upbeat on infrastructure management services for the Indian market.

Contd.....

Notes

The company has planned three centres in Maharashtra (Mumbai, Pune and Nagpur) expecting greater demand from North and Western India, said Mr Mani Sam, Head-India Operations, CSS Group. Other centres include Bangalore, Hyderabad, New Delhi, Kolkata and Ahmedabad.

He added that over time, the company should be able to maintain operating profit margins ranging between 20 and 25 per cent.

Commenting on the fact that large clients typically went only with large vendors, he said, "That's where our alliance partners help, in terms of penetrating large clients." The CSS Group has an alliance with IBM. It is learnt that two more alliances are to be finalised shortly.

To hire more

CSS Software will hire over 500 people in the next 12 months, quadrupling headcount from the present 150 employees to 650 employees.

It plans to add 10 Indian more customers this year to its existing 10. "We are close to signing five new customers," said Mr Ramani.

3.3 Summary

- Generally, text is unformatted in your Web documents.
- However, to emphasize specific letters, words, or phrases which reflects the characteristics about the text.
- Also, you can use some text formatting options including boldfacing and italicizing.
- The character elements describe the type of the text formatting.
- It is also the treat as *Logical codes* which shows the meaning of the text.
- The look elements deal with the specific text formatting.
- It is also the treat as Physical codes, which use for how the text is to be displayed and not it's meaning.

3.4 Keywords

Character Elements: The character elements describe the type of the text formatting.

Look Elements: The look elements deal with the specific text formatting.

3.5 Review Questions

1. Explain why the character elements describe the type of the text formatting.
2. The <BIG> text element defines text that should be displayed in a larger font than usual. Explain this with an suitable example.
3. Write a program which uses a text-align property to set the horizontal alignment of a text.
4. Write a program which uses the text-decoration property to set or remove decorations from text.
5. Write a program which uses the text-indentation property to specify the indentation of the first line of a text.

Notes

6. The <INS> element indicates that the text has been inserted into the document since the last revision. Explain with the suitable example.
7. Using CSS, you can set your text in any point size, the same way that all other print and graphic designers can already. Explain with a program.
8. Explain how <ABBREV> element defines an abbreviation.
9. CSS gives you full control over the size of your text. Analyze
10. Write a program which uses the text-transform property to specify uppercase and lowercase letters in a text.

Answers: Self Assessment

1.	<EM>	2.	<STRONG>
3.	<SAMP>	4.	<ADDRESS>
5.	<CITE>	6.	<DEL>
7.	<DFN>	8.	<PERSON>
9.	<ACRONMY>	10.	<B>
11.	<I>	12.	<U>
13.	<TT>	14.	<STRIKE>
15.	<SUP>		

3.6 Further Readings

Books

Chuck Musciano, Bill Kennedy, *HTML & XHTML: The definitive guide,* O'Reilly Media, Inc.

David Mercer, *HTML,* McGraw-Hill Professional

Gary B. Shelly, Denise M. Woods, William J. Dorin, *HTML, XHTML, and CSS: Comprehensive,* Cengage Learning

Willard, *HTML: A Beginner'S Guide,* Tata McGraw-Hill Education

Online links

http://www.yourhtmlsource.com/stylesheets/csstext.html

http://www.w3schools.com/css/css_text.asp

Notes

Unit 4: Cascading Style Sheets

CONTENTS

Objectives

Introduction

4.1 CSS Web Template

4.2 How does CSS Work?

4.3 CSS Declaration

4.4 CSS Properties or Applying Style

4.4.1 Font Properties

4.4.2 Text Properties

4.4.3 Outline Properties

4.4.4 List Properties

4.4.5 Generated Content Properties

4.4.6 Margin Properties

4.4.7 Border Properties

4.4.8 Padding Properties

4.4.9 Positioning Properties

4.4.10 Classification Properties

4.4.11 Dimension Properties

4.4.12 Color/Background Properties

4.4.13 Units of Measure

4.5 Different Ways to Implement CSS

4.5.1 Inline CSS

4.5.2 Embedded CSS or Internal Styles

4.5.3 External Style Sheets/Linked CSS

4.5.4 Importing Style Sheets

4.6 Limitations of CSS

4.7 Advantages of CSS

4.8 Style Tag

4.9 DIV and SPAN

4.10 Creating and Using CSS Classes

4.11 Summary

4.12 Keywords

4.13 Review Questions

4.14 Further Readings

Notes

Objectives

After studying this unit, you will be able to:

- Scan CSS (Cascading Style Sheets)
- Recognize the CSS properties
- Describe Style, DIC, SPAN tags
- Demonstrate the internal and external stylesheets
- Explain the creation and use of classes
- Discuss the applying of style and images on text.

Introduction

CSS stands for Cascading Style Sheets, created by Hakon Wium Lie of MIT in 1994. It is a markup language used to directly effect the "look" of web pages. This includes overall layout, text size, style and formatting, table formatting, link properties, and more. CSS style blocks are also commonly referred to as rules. These rules can be embedded into an individual HTML page or placed in an external file that will control many individual pages on your website. Thus changing a property in one place in the linked style sheet will immediately make that change on every web page that is linked to it. When used correctly, CSS allows one to define the look of an entire site in one single document. It provides a means for authors to specify how they wish documents written in a markup language such as XML or HTML to be formatted.

In short Cascading Style Sheets:

- Styles define how to display HTML elements; styles are normally stored in Style Sheets.
- "Cascading" refers to the fact that each different style declaration can be "cascaded" under the one above it, forming a parent-child relationship between the styles.

4.1 CSS Web Template

"CSS Web Template" is a website design created using Cascading Style Sheets (CSS) technology. Cascading style sheets provide web developers an easy way to format and to style web pages. CSS will be used even more because it is seen the same way by all browsers, making it the best option during the browser wars.

CSS templates allow enhanced browser and platform compatibility (CSS supporting browsers are used by 99.98% of existent web surfers). Your website will look perfect in Windows, UNIX and MAC browsers. The template is primarily tested on multiple platforms to ensure better requirements compliance.

Style Sheets are the easiest way to provide a default font styling for HTML. Therefore, you can modify the whole text and link colors on every page by editing just a single CSS file! CSS also makes your coding much easier because you don't have to repeat the many formatting tags. This streamlined code equals faster download time and reduced bandwidth usage.

CSS templates show 100% compatibility with Macromedia Dream weaver and MS Front Page. Other web editors also handle this technology easier than the usual HTML code.

CSS technology is a great step forward in web development. Separated content and presentation adds more flexibility to your website. And, you'll have no problems with future modification of your website.

Notes

Notes All templates are based on w3c.org technologies and standards that make your website much more user-friendly.

4.2 How does CSS Work?

In this lesson, you will learn how to make your first style sheet. You will get to know about the basic CSS model and which codes are necessary to use CSS in an HTML document. Many of the properties used in Cascading Style Sheets (CSS) are similar to those of HTML. Thus, if you are used to use HTML for the layout, you will most likely recognize many of the codes. Let us look at a concrete example.

The basic CSS syntax:

Let's say we want a nice red color as the background of a webpage:

Using HTML, we could have done it like this:

```
<body bgcolor="#FF0000">
```

With CSS the same result can be achieved like this:

```
body {background-color: #FF0000;}
```

Notes As you will note, the codes are more or less identical for HTML and CSS. The above example also shows you the fundamental CSS model.

Self Assessment

Fill in the blanks:

1. The …….... is primarily tested on multiple platforms to ensure better requirements compliance.
2. ……..... are the easiest way to provide a default font styling for HTML.
3. Many of the properties used in ……..... are similar to those of HTML.

4.3 CSS Declaration

A CSS declaration represents the effect to be applied to the element(s) and it consists of two more parts — a property and a value. The property is that aspect of an element's presentation that is being modified, such as its color, its width, or its placement on the page. Dozens of properties are available in the CSS language, and you'll become familiar with many of them in the pages of this book.

The declaration block, denoted by the curly braces ("{","'}"). Within this block are zero or more declarations. While defining multiple declarations, it must be separated by a semicolon (";").

Each declaration consists of a property and a value, separated by a colon (":"). Properties are identifiers, such as font-size, background-color, etc. The values permitted depend on the individual property.

Did u know? What is the use of whitespace in CSS declarations?

Whitespace may be used around any of these tokens, so you can format them for easy reading. You can also insert comments anywhere whitespace is permitted.

Notes

4.4 CSS Properties or Applying Style

The property value delivers the specific style that should be applied to the selected element. The values accepted depend on the particular property, and some properties accept multiple values, separated by spaces.

A Property is an identifier. Every property has its own syntax and constraints on the values it accepts. Property values also often indicate measurement units to aid in rendering to the specified media.

4.4.1 Font Properties

A font is a set of glyphs, which map to known characters in one or more languages. The glyphs within a font usually all share common characteristics such as size, style, and design. The Font properties allow control over many different font criteria such as boldness, italics, size, and specific or generic font name.

In CSS1, fonts were identified only by their name. If none of the fonts an author specified were available, the browser had to resort to using default fonts. This is clearly not the best solution when documents are available in an environment as heterogeneous as the web.

CSS2 font selection greatly expands the capabilities of browsers with respect to matching fonts when the intended font is not available. Dozens of "font descriptors" have been introduced which break down a desired font's characteristics. When a desired font is not present, all of these descriptors can be used to make an intelligent "best-choice" for an alternate font match. In the event a best match cannot be generated, fonts may even be generated on the fly or downloaded as needed over the web.

FONT	
font	font-style font-variant font-weight font-size/line-height font-family caption | icon | menu| message-box | small- caption | status-bar
font-family	family-name generic-family
font-size	xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger length %
font-size-adjust	None Number
font-stretch	normal | wider | narrower | ultra-condensed | extra- condensed | condensed | semi-condensed | semi- expanded | expanded | extra-expanded | ultra-expanded
font-style	normal | italic | oblique
font-variant	normal | small-caps
font-weight	Normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

Notes

Example:

```
P {font: 12pt/14pt sans-serif}
P {font: 80% sans-serif}
P {font: x-large/110% "new century schoolbook", serif}
P {font: bold italic large Palatino, serif}
P {font: normal small-caps 120%/120% fantasy}
```

4.4.2 Text Properties

These properties control the visual appearance and presentation of text content (characters, words, blocks and the spacing between them.) Some of the special physical formatting effects from HTML have found their home in the Text properties.

TEXT	
color	Color
direction	ltr | rtl
letter-spacing	normal length
text-align	left | right | center | justify
text-decoration	none | underline | overline | line-through | blink
text-indent	length %
text-shadow	none color length
text-transform	none | capitalize | uppercase | lowercase
unicode-bidi	normal | embed | bidi-override
white-space	normal | pre | nowrap
word-spacing	normal length

Example:

```
H1 {word-spacing: 1em}
BLOCKQUOTE {letter-spacing: 0.1em}
BLOCKQUOTE {letter-spacing: 0}
BLOCKQUOTE {letter-spacing: 0cm}
A:link, A:visited, A:active {text-decoration: underline}
H1 {text-transform: uppercase}
DIV.center {text-align: center}
P {text-indent: 3em}
DIV {line-height: 1.2; font-size: 10pt}       /* number */
DIV {line-height: 1.2em; font-size: 10pt}     /* length */
DIV {line-height: 120%; font-size: 10pt}      /* percentage */
```

Notes

4.4.3 Outline Properties

New properties in CSS2 allow outlines to be created for an element to help it stand out from other elements. CSS Outlines are very similar to borders except for several subtle differences:

- Border sides can be independently controlled, Outline sides cannot. The visual effects for an outline are uniform for all sides.
- Outlines may be non-rectangular.
- Outlines do not take up space in the box-rendering model; they are rendered on "top" of the element's box. An outline does not affect the position or size of the rendering box.

OUTLINE	
Outline	Outline-color outline-style outline-width
outline-color	color invert
outline-style	none | dotted | dashed | solid | double | groove | ridge | inset | outset
outli ne-width	thin | medium | thick length

Example:

```
BUTTON {outline-width: thick}
focus  {outline: thick solid black}
:active {outline: thick solid red}
```

4.4.4 List Properties

Lists are used to organize and render one or more data items with some relationship to each other. With a list structure, extra information is used to give some indication of the degree of relationship between the items.

However, this extra information is not actually stored in the document itself, it is generated automatically when it is rendered. List structures generate two blocks for each item of rendered data - a box for each data item, along with a separate marker box for the visual list iteration object (such as a bullet, number, or image) that is created by the system. The list properties control the type and placement of the marker box in list contexts.

LIST & MARKERS	
list-style	list-style-type list-style-position list-style-image
list-style-image	none url
list-style-position	inside | outside
list-style-type	none | disc | circle | square | decimal | decimal-leading- zero | lower-roman | upper- roman | lower-alpha | up- per-alpha | lower-greek | lower-latln | upper-latin | hebrew | armenian | geor- gian | cjk-ideographic | hiragana | katakana | hira- gana-iroha | katakaa-iroha
ma rker-offset	auto length

Notes

Example:

```
UL {list-style: upper-roman inside}
UL UL {list-style: circle outside}
LI.square {list-style: square}
OL.alpha  {list-style: lower-alpha}
UL  {list-style: disc}
UL {list-style: url (http://png.com/ellipse.png) disc}
```

4.4.5 Generated Content Properties

The basic aim of CSS is to offer hints and aids for the direct translation of a document and its elements to differing physical media. There are times when the information contained in a document's structure is not enough to display the intended content. In HTML, the various list structures are a perfect example - the bullets and iterative/hierarchical numbers that are rendered before the items in the list are not explicitly present in the source document. This content is automatically generated by the browser when rendering an element from the source document.

CSS2 extends this concept of automatic content generation to allow content (strings, quotes, list generators or objects) to be rendered before or after any element. Generated content includes aural rendering as well; auditory icons can offer powerful usability enhancements for a listener.

GENERATED CONTENT	
content	String url counter (name) counter (name, list-style- type) counters (name, string) counters (name, string, list- style-type) attr (X) open-quote | close-quote | no-open-quote | no-close- quote
counter-increment	None identifier number
counter-reset	None identifier number
quotes	None string string string string

Example:

```
BLOCKQUOTE P:before       {content: open-quote}
BLOCKQUOTE P:after        {content: no-close-quote}
BLOCKQUOTE P.last:after {content: close-quote}
[LANG|=fr] > *   {quotes: "«" "»" "\2039" "\203A"}
[LANG|=en] > *   {quotes: "\201C" "\201D" "\2018" "\2019"}
[LANG|=fr] > * {quotes: "«" "»" "‹" "›"}
[LANG|=en] > * {quotes: "“" "”" "‘" "’"}
H1:before {
```

Notes

```
      content: "Chapter " counter (chapter) ". ";
      counter-increment: chapter;   /* Add 1 to chapter */
      counter-reset: section;       /* Set section to 0 */
}
H2:before {
      content: counter (chapter) "." counter (section) " ";
      counter-increment: section;
}
H1 {counter-reset: section -1}
H1 {counter-reset: imagenum 99}
OL {counter-reset: item}
LI {display: block}
LI:before {content: counter (item) ". "; counter-increment: item}
```

4.4.6 Margin Properties

In CSS, the fundamental visual rendering model places all components of the document tree in physical and virtual rectangular boxes, each having a specific height and width. An element's rendering box consists primarily of an element's content at the center (text, images, etc.) Surrounding the element's content (moving outward in rectangular layers/strips) are optional padding, surrounded by any optional border effects, surrounded in turn by any optional margin values that may be specified.

The margin properties allow the author to specify how much space will be inserted between other exterior elements and the current element border. Negative values may be used with margin properties to create overlapping content.

Each side of the margin dimensions (top, right, bottom and left) can be addressed and controlled, independently using separate properties, or a convenient shorthand notation may be used that controls multiple sides at once.

MARGIN	
margin	margin-top margin-right margin-bottom margin-left
margin-bottom	auto length %
margin-left	auto length %
margin-right	auto length %
margin-top	auto length %

Example:

```
BODY {margin: 2em} /* all margins set to 2em */
BODY {margin: 1em 2em} /* top & bottom = 1em, right & left = 2em */
```

Notes

```
BODY {margin: 1em 2em 3em} /* top=1em, right=2em, bottom=3em, left=2em */
BODY {
   margin-top: 1em;
   margin-right: 2em;
   margin-bottom: 3em;
   margin-left: 2em;          /* copied from opposite side (right) */
}
H1 {margin-top: 2em}
H1 {margin-right: 12.3%}
H1 {margin-bottom: 3px}
```

4.4.7 Border Properties

In CSS, the fundamental visual rendering model places all components of the document tree in physical and virtual rectangular boxes, each having a specific height and width. An element's rendering box consists of an element's content at the center (text, images, etc.) Surrounding the element's content (moving outward in rectangular layers/strips) are optional padding, surrounded by any optional border effects, surrounded in turn by any optional margin values that may be specified.

The border properties allow an author to specify the width, color, and style of the border area (between any specified padding and margins) of an element's rendering box. While the capability to create simple line effects has been available in HTML via tables, the CSS border properties give authors much more power in creating such effects and allow them to be applied to any element.

Each side of the border dimensions (top, right, bottom and left) can be addressed and controlled, independently using separate properties, or a convenient shorthand notation may be used that controls multiple sides at once.

Border Style

The border-style property specifies what kind of border to display.

Notes None of the border properties will have ANY effect unless the border-style property is set!

Border-style values: There can be following types of border values:

1. none: Defines no border
2. dotted: Defines a dotted border
3. dashed: Defines a dashed border
4. solid: Defines a solid border
5. double: Defines two borders. The width of the two borders are the same as the border-width value
6. groove: Defines a 3D grooved border. The effect depends on the border-color value
7. ridge: Defines a 3D ridged border. The effect depends on the border-color value
8. inset: Defines a 3D inset border. The effect depends on the border-color value
9. outset: Defines a 3D outset border. The effect depends on the border-color value

Notes

Border Width

The border-width property is used to set the width of the border.

The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

> *Notes* The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Border Color

The border-color property is used to set the color of the border. The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"

You can also set the border color to "transparent".

Border - Individual Sides

In CSS it is possible to specify different borders for different sides:

The border-style property can have from one to four values.

- border-style:dotted solid double dashed;
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed
- border-style:dotted solid double;
  - top border is dotted
  - right and left borders are solid
  - bottom border is double
- border-style:dotted solid;
  - top and bottom borders are dotted
  - right and left borders are solid
- border-style:dotted;
  - all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

Notes

To shorten the code, it is also possible to specify all the border properties in one property. This is called a shorthand property.

The shorthand property for the border properties is "border":

When using the border property, the order of the values are:

- border-width
- border-style
- border-color

It does not matter if one of the values above are missing (although, border-style is required), as long as the rest are in the specified order.

Example: How to set all the border properties in one declarartion

```
<html>
<head>
<style type="text/css">
p{border:5px solid red;}
</style>
</head>
<body>
<p>This is some text in a paragraph.</p>
</body>
</html>
```

Result:

There is some text in a paragraph

BORDER	
border	border-width border-stye border-color
border-bottom	border-bottom-width border-style border-color
border-bottom-color	border-color
border-bottom-style	border-style
border-bottom-width	thin | medium | thick length
border-color	color
border-left	border-left-width border-style border-color
border-left-color	border-color
border-left-style	border-style
border-left-width	thin | medium | thick length

Notes

Example:

```
P {border: solid red}
P {
   border-top: solid red;
   border-right: solid red;
   border-bottom: solid red;
   border-left: solid red
}
H1 {border-bottom: thick solid}
H1 {border-bottom: thick solid red}
H1 {border-width: thin}                    /* thin thin thin thin */
H1 {border-width: thin thick}              /* thin thick thin thick */
H1 {border-width: thin thick medium}       /* thin thick medium thin */
H1 {border-width: thin thick medium thin} /* thin thick medium thin */
```

4.4.8 Padding Properties

In CSS, the fundamental visual rendering model places all components of the document tree in physical and virtual rectangular boxes, each having a specific height and width. An element's rendering box consists of an element's content at the center (text, images, etc.) Surrounding the element's content (moving outward in rectangular layers/strips) are optional padding, surrounded by any optional border effects, surrounded in turn by any optional margin values that may be specified.

The padding properties allow the author to specify how much space will be inserted between the element border and the element content. Negative values are not allowed.

Each side of the padding dimensions (top, right, bottom and left) can be addressed and controlled, independently using separate properties, or a convenient shorthand notation may be used that controls multiple sides at once.

Padding — Individual Sides

In CSS, it is possible to specify different padding for different sides such as:

padding-top:25px;

padding-bottom:25px;

padding-right:50px;

padding-left:50px;

Padding — Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property. The shorthand property for all the padding properties is "padding":

Example: padding:25px 50px;

The padding property can have from one to four values.

- padding:25px 50px 75px 100px;
  - top padding is 25px
  - right padding is 50px

Notes

- bottom padding is 75px
- left padding is 100px

- padding:25px 50px 75px;
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px
- padding:25px 50px;
  - top and bottom paddings are 25px
  - right and left paddings are 50px
- padding:25px;
  - all four paddings are 25px

All CSS Padding Properties

Property	Description
padding	A shorthand property for setting all the padding properties in one declaration
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element
padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element

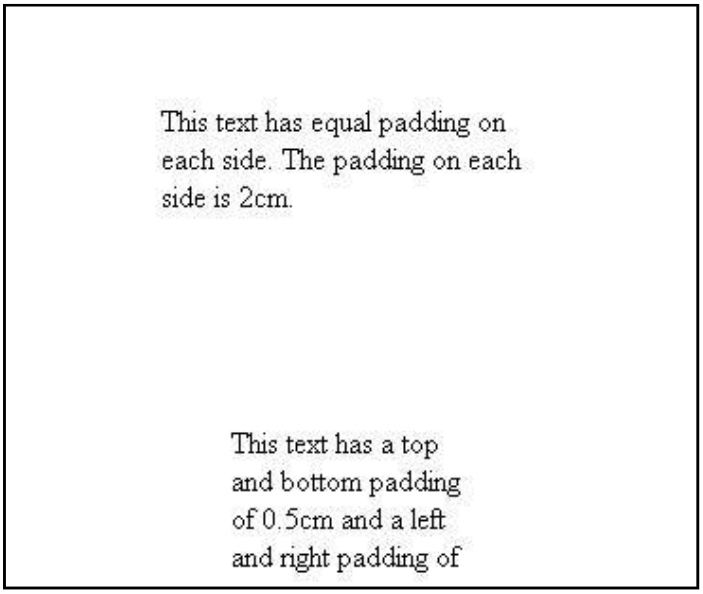
Example: How to set all the padding properties in one declaration

Source code:

```
<html>
<head>
<style type="text/css">
p.ex1 {padding:2cm;}
p.ex2 {padding:0.5cm 3cm;}
</style>
</head>
<body>
<p class="ex1">This text has equal padding on each side. The padding on each side is 2cm.</p>
<p class="ex2">This text has a top and bottom padding of 0.5cm and a left and right padding of 3cm.</p>
</body>
</html>
```

Result:

Notes



PADDING	
Padding	padding-top padding-right padding-bottom padding-left
padding-bottom	length %
padding-left	length %
padding-right	length %
padding-top	length %

Example:

```
BLOCKQUOTE {padding-top: 0.3em}
BLOCKQUOTE {padding-left: 20%}
BLOCKQUOTE {padding-bottom: 2em}
H1 {background: white;
   padding: 1em 2em;
}
```

4.4.9 Positioning Properties

When an element is called 'positioned' in the visual formatting model, it may or may not be rendered immediately after the previous element in the document tree. An element that is positioned ('position' property) has its rendering offset from the canvas origin or from its normal flow position via the 'top', 'right', 'bottom', and 'left' properties.

Notes

In addition to determining position offsets, positioning properties also specify the visible display area of an element's rendering box ('clip') and directives indicating what should happen when an element's content falls outside the rendering box ('overflow'.)

But, positioning in CSS is not just about rendering in 2-dimensions. It also allows for placement and overlap of elements in an imaginary third dimension perpendicular to the screen ('z-axis'.)

The final property included in this category, 'vertical-align', is not involved with "CSS positioning" per se, but it does involve vertical positioning of content within a rendered line box.

The number in the "CSS" column indicates in which CSS version the property is defined.

1. bottom: Sets the bottom margin edge for a positioned box
2. clip: Clips an absolutely positioned element
3. cursor: Specifies the type of cursor to be displayed
4. left: Sets the left margin edge for a positioned box
5. overflow: Specifies what happens if content overflows an element's box
6. position: Specifies the type of positioning for an element
7. right: Sets the right margin edge for a positioned box
8. top: Sets the top margin edge for a positioned box
9. z-index: Sets the stack order of an element

Property	Description
height	Sets the height of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
width	Sets the width of an element

POSITIONING	
bottom	auto % length
clip	shape auto
left	auto % length
overflow	visible | hidden | scroll | auto
position	static | relative | absolute | fixed
right	auto % length
top	auto % length
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom length %
z-index	auto number

Notes

Example:

```
BODY {height: 8.5in} /* Required for percentage heights below */
       #header {
          position: fixed;
          width: 100%;
          height: 15%;
         top: 0;
         right: 0;
         bottom: auto;
         left: 0;
       }
       #sidebar {
          position: fixed;
          width: 10em;
          height: auto;
         top: 15%;
         right: auto;
         bottom: 100px;
         left: 0;
       }
       #main {
          position: fixed;
          width: auto;
          height: auto;
         top: 15%;
         right: 0;
         bottom: 100px;
         left: 10em;
       }
       #footer {
          position: fixed;
          width: 100%;
          height: 100px;
   top: auto;
         right: 0;
         bottom: 0;
         left: 0;
       }
```

Notes

4.4.10 Classification Properties

The classification properties serve to classify elements and select which layout models will be used in their display. In the visual media-formatting model, an element's rendering is based heavily upon, among other things, the positioning method specified (floating, positioned or normal flow positioning) and the classification type of the element's box (the 'display' property).

CLASSIFICATION	
clear	left | right | both | none
cursor	url auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize| s-resize | w-resize | text | wait | help
display	none | inline | block | list- item | run-in | compact | marker | tabie | in line-table| table-row-group | table- header-group | table-footer- group | table-row | table-column-group | table- column | table-cell | table caption
float	left | right | none
position	static | relative | absolute |
visibility	visible | hidden | collapse

Example:

```
P {display: block}
EM {display: inline}
LI {display: list-item}
IMG {display: none}
More Practical Example
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<STYLE type="text/css">
<!—
    #container1 {position: absolute;
                top: 2in; left: 2in; width: 2in}
    #container2 {position: absolute;
                top: 2in; left: 2in; width: 2in;
                visibility: hidden;}
—>
</STYLE>
</HEAD>
<BODY>
<P>Choose a suspect:</P>
<DIV id="container1">
```

Notes

```
    <IMG alt="Al Capone"
          width="100" height="100"
          src="suspect1.jpg">
    <P>Name: Al Capone</P>
    <P>Residence: Chicago</P>
</DIV>
<DIV id="container2">
    <IMG alt="Lucky Luciano"
          width="100" height="100"
          src="suspect2.jpg">
    <P>Name: Lucky Luciano</P>
    <P>Residence: New York</P>
</DIV>
<FORM method="post"
        action="http://www.bpkm.org.in/process-bums">
   <P>
    <INPUT name="Capone" type="button"
           value="Capone"
             onclick='show ("container1"); hide ("container2")'>
    <INPUT name="Luciano" type="button"
           value="Luciano"
             onclick='show ("container2"); hide ("container1")'>
</FORM>
</BODY>
</HTML>
```

4.4.11 Dimension Properties

In CSS, the fundamental visual rendering model places all components of the document tree in physical and virtual rectangular boxes, each having a specific height and width. The dimension properties describe the details influencing the calculation and visual rendering of an element's line, in-line and block box dimensions. The actual values for an element's visually rendered height and width are dependent on many criteria, including the intrinsic type of the current element and the dimensions of the element's containing block.

All CSS Dimension Properties

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	Description	Values	CSS
height	Sets the height of an element	Auto *length* % inherit	1
max-height	Sets the maximum height of an element	none *length* % inherit	2
max-width	Sets the maximum width of an element	none *length* % inherit	2
min-height	Sets the minimum height of an element	*length* % inherit	2
min-width	Sets the minimum width of an element	*length* % inherit	2
width	Sets the width of an element	auto *length* % inherit	1

Notes

Example: How to set the height of the image using pixel.

Source Code ***Result***

```
<html>
<head>
<style type="text/css">
img.normal {height:auto;}
img.big {height:50%;}
img.small {height:10%;}
</style>
</head>

<body>
<img class="normal" src="logocss.gif"
width="95" height="84" /><br />
<img class="big" src="logocss.gif" width="95"
height="84" /><br />
<img class="small" src="logocss.gif"
width="95" height="84" />
</body>
</html>
```

4.4.12 Color/Background Properties

Color and Background properties allow color control of both the foreground and background of an element's rendering box. The background properties also allow sophisticated placement and control of images to create background textures. The color and background capabilities behaviors originally available in HTML have been absorbed in to these CSS properties and greatly expanded.

Setting Colors

Basically you have three color options with CSS:

1. Setting the foreground color for contents
2. Setting the background color for an area
3. Setting a background image to fill out an area.

In plain HTML, colors can either be entered by name (red, blue etc.) or by a hexadecimal color code (for example: #FF9900).

With CSS you have these options:

1. ***Common name:*** You can define colors with the use of common names, by simply enter the name of the desired color.

   For example: .myclass {color:red; background-color:blue;}

2. ***Hexadecimal value:*** You can define colors with the use of hexadecimal values, similar to how it's done in plain HTML.

   For example: .myclass {color:#000000; background-color:#FFCC00;}

Notes

3. ***RGB value:*** You can define colors with the use of RGB values, by simply entering the values for amounts of Red, Green and Blue.

   For example: .myclass {color:rgb(255,255,204); background-color:rgb(51,51,102);}

   You can also define RGB colors using percentage values for the amounts of Red, Green and Blue:

   For example: .myclass {color:rgb(100%,100%,81%); background-color:rgb(81%,18%,100%);}

Setting Background Colors

Background colors are defined similar to the colors mentioned above. For example you can set the background color of the entire page using the BODY selector:

BODY {background-color:#FF6666;}

1. Setting a background image: CSS lets you set a background image for both the page and single elements on the page.

   In addition, CSS offers several positioning methods for background images.

   You can define the background image for the page like this:

   BODY {background-image:url(myimage.gif);}

2. You can control the repetition of the image with the background-repeat property.

   a. background-repeat:repeat: Tiles the image until the entire page is filled, just like an ordinary background image in plain HTML.

   b. background-repeat:repeat-x: Repeats the image horizontally - but not vertically.

   c. background-repeat:repeat-y: Repeats the image vertically - but not horizontally.

   d. background-repeat:no-repeat: Does not tile the image at all.

3. Positioning a background: Background positioning is done by entering a value for the left position and top position separated by a space.

   In this example the image is positioned 75 pixels from the upper left corner of the page:

   BODY {background-image:url(myimage.gif); background-position: 75px 75px;}

Notes Background positioning is not supported by Netscape 4 browsers.

4. Fixing a background: You can fixate an image at a certain position so that it doesn't move when scrolling occurs.

   BODY {background-image:url(myimage.gif); background-attachment: fixed;}

Notes Background fixation is not supported by Netscape 4 browsers.

5. Setting multiple background values: Rather than defining each background property with its own property you can assign them all with the use of the background property.

Look at this example:

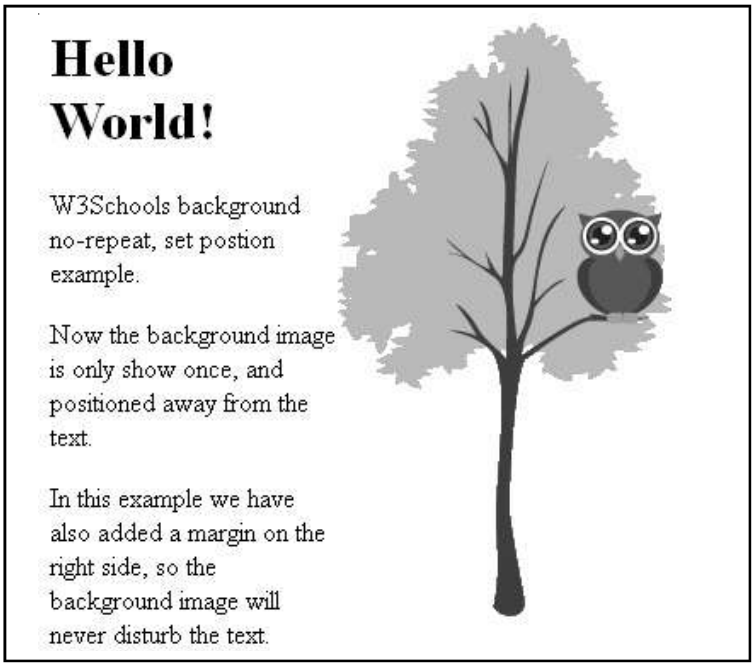
BODY {background:green url(myimage.gif) repeat-y fixed 75px 75px;}

Notes

Example: How to position a background image

```
<html>
<head>
<style type="text/css">
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
background-position:right top;
margin-right:200px;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>W3Schools background no-repeat, set postion example.</p>
<p>Now the background image is only show once, and positioned away from the text.</p>
<p>In this example we have also added a margin on the right side, so the background image will never disturb the text.</p>
</body>
</html>
```

Result:



Notes

background	background-color background-image background-repeat background-attachment background-position
background-attachment	scroll | fixed
background-color	color-rgb color-hex color-name Transparent
background-image	url None
background-position	top left | top center | top right | center left | center center | center right | bottom left | bottom center | bottom right x-%y-% x-pos y-pos
background-repeat	repeat | repeat-x | re- peat-y | no-repeat

Example:

```
EM {color: red}                  /* natural language */
EM {color: rgb (255,0,0)}       /* RGB range 0-255    */
H1 {background-color: #F00}
BODY {background-image: url (marble.gif)}
P {background-image: none}
BODY {background: red url (pendant.gif);
   background-repeat: repeat-y;
}
BODY {background: red url (pendant.gif);
   background-repeat: repeat-y;
   background-attachment: fixed;
}
BODY {background: url (banner.jpeg) right top}      /* 100% 0% */
BODY {background: url (banner.jpeg) top center}     /* 50% 0% */
BODY {background: url (banner.jpeg) center}         /* 50% 50% */
BODY {background: url (banner.jpeg) bottom}         /* 50% 100% */
BODY {background-image: url (logo.png);
   background-attachment: fixed;
   background-position: 100% 100%;
}
BODY {background: red}
P {background: url (chess.png) gray 50% repeat fixed}
```

Notes

4.4.13 Units of Measure

CSS is used to render information to a device. These devices exist in the real world and have dimensions and capabilities that are measurable. Therefore, CSS provides a way for document content to explicitly specify how it will be mapped to a physical device. If a property value contains an illegal unit measure, the entire style rule should be ignored.

UNITS of MEASUREMENT	
%	Percentage
cm	Centimeter
em	1em = current font size of current element
ex	lex = 1/2 current size of current element
in	Inch
mm	Millimeter
pc	pica (= 12 points)
pt	point (= 1/72 inch)
Px	Pixel
COLORS	
Color name	Red, blue, green, dark green
rgb(x,y,z)	Red = rgb (255,0,0)
rgb(x%,y%,z%)	Red = rgb(100%,0,0)
#rrggbb	Red = #00000 (or shorthand =#f00)

Self Assessment

Fill in the blanks:

4. Every property has its own syntax and …… on the values it accepts.
5. The …… within a font usually all share common characteristics such as size, style, and design.
6. An outline does not affect the position or size of the …… box.
7. Generated …… includes aural rendering as well; auditory icons can offer powerful usability enhancements for a listener.
8. An element's rendering box consists primarily of an element's content at the ……
9. CSS is used to render information to a ……

4.5 Different Ways to Implement CSS

There are four (4) different ways to use CSS in your Web pages: Inline CSS, Embedded CSS, Linked CSS and Importing CSS. They all provide advantages over normal HTML.

4.5.1 Inline CSS

This is the HTML specific style inclusion method, in which style information is directly attached to the HTML elements, they affect. A universal element attribute, called STYLE, may be specified for an HTML element, taking as a value one or more style Declarations.

Notes

Many of the powerful CSS syntax mechanisms (like Selectors) are not applicable in Inline CSS. This style specification method is most useful, when serving as an override on a case-by-case basis for style rules specified in an External or Embedded style sheet. Inline styles have higher cascade precedence than the other specification methods.

The simplest and most direct way of applying CSS to an element is to write it into the tag itself as a style attribute style="...". CSS styles applied this way have the highest priority over other style definitions, i.e., they will overwrite existing styles.

```
Syntax:
<[element name] STYLE="[CSS property]:[property value]">
```

Example: `<p style="border:2px solid #8dc919">`

Using HTML Code:

Boy do I love Saturdays. Nothing like sleeping 'til 10 and not waking up for work.

Using inline CSS: (this above same paragraph could be coded like)

 Boy do I love Saturdays. Nothing like sleeping 'til 10 and not waking up for work.

Usage Guidelines:

1. ***Use as little as possible:*** Remember that we're trying to separate content from presentation. Inline styles are the exact opposite. They're inefficient because they'll indulge up your code and are difficult to maintain. Only use them for some quick testing of CSS styles or for exceptions, where a style is only used at a particular place and nowhere else on the homepage.
2. ***Don't use double quotes:*** Because double quotes are already used for marking the start and end of the style attribute, having another double quote in-between would abruptly terminate the CSS instructions. Use single quotes instead.

Example:

```
<p style="font-family:'trebuchet ms'">...</p> is ok but not <p style= "font
family:"trebuchet ms"">...</p>.
```

CSS offers, Web designers more precise control over the look of their pages. With inline CSS, we can specify the exact pixel size of a font. In comparison, <font> tags limit us to numbers (-7,-6, -5, -4-3, -2, -1, 1, 2, 3, 4, 5, 6…). These numbers correspond to the font settings in the user's browser, and thus eliminate the control you have over the final size of a font.

Apart from this small improvement in the control of our font size, inline CSS does not really reduce the amount of code needed to format text. It just accomplishes a similar result in a different way. This leads us to a slightly better method of CSS implementation.

Task Analyze in a group of four that why inline styles have higher cascade precedence than the other specification methods.

[See complex HTML example using Inline Style Sheets]

Notes

```
<html>
       <head>
                <title>Document Title</title>
       </head>
 <body LINK="#ff8080" VLINK="#ff0000" ALINK="a05050" STYLE="background: #000000; color: #80c0c0">
 <h1 ALIGN="center" STYLE="background: #000080; font: 36pt/40pt courier; font-variant: small-caps; border: thick dashed blue">Welcome to my home page!</h1>
       <br><br>
 <p><span STYLE="margin-left: 25px">Hi there! If you are reading this then you have found my home page! Congratulations! </span>
We know it can be hard to find my pages, but I bet that you feel lucky now. Now that you are here, please take a look at my page of links to <a HREF="http://www.mysite.com/coolsites.html">cool sites</a> or sign my <a HREF="http://www.mysite.com/guestbook.html">guest book</a></p>
 <div><div STYLE="font-weight: bold; margin-left: 30px">
<span STYLE="font-size: x-large; color: #ffffff">M</span>y wonderful poetry</div> is available if you are REALLY bored. Why not give it a spin? </div>
 <h2 STYLE="background: #000080; color: green; line-height: 50px; font-size: 40px">
 The Best Poetry I <em STYLE="font-weight: 900">NEVER</em> Wrote</h2>
       <ul>
           <li>'There Once Was A Man From Nantucket' -
                  <span STYLE="font-family: 'comic sans ms', fantasy; color: rgb (100%, 100%, 0%)">Anonymous</span></li>
           <li>'Cool In Fur' -
                  <span STYLE="font-family: 'comic sans ms', fantasy; color: rgb (100%, 100%, 0%)">Harry B. Foot</span></li>
           <li>And My All Time Fave:
           <ul>
             <li STYLE="font-size: x-large; font-style: italic"> 'A Toast To My Toaster' -
                  <span STYLE="font-family: 'comic sans ms', fantasy; color: rgb (100%, 100%, 0%)">Me!</span></li>
           </ul>
       </li></ul>
 <blockquote STYLE="background: #000080; color: #00ff00; margin-left: 2cm">Brought to you by the letter
 <span STYLE="font-family: 'comic sans ms', fantasy; color: rgb(100%, 100%, 0%)">&quot;H&quot;</span>
 and <span STYLE="Padding: 20px; border: 20px groove #ffffff">Joe Shmoe</span> </blockquote>
 <div STYLE="background: #000080; font-weight: bold; margin-left: 30px">
 <span STYLE="font-size: x-large; color: #ffffff"> W</span>hen you are done looking through these masterpieces, I encourage you to visit my proud sponsor!! </div>
```

Notes

```
<p STYLE="font: 12pt/14pt sans-serif; margin: 5px 0px 2px 25px; border: medium dashed #ff0000; background: white url (http://www.foo.com/image.gif) repeat-x fixed top right}">

<span STYLE="font: 36pt/40pt courier; font-variant: small-caps; border: thick dashed blue}"> ADVERTISEMENT:</span>

Buy Navel Lint Contemplator! It's a browser and it's a sandwich spread! Go to our <a HREF="http://www.navellint.com">home page</a> without delay! Why? Because shelf life is only 8 hours unless refrigerated. We know that makes it hard to browse, but it promotes frequent upgrading to the latest and greatest version. </p>

<h6 STYLE="font-size: xx-small! important; color: red! important">All standard disclaimers apply. Your life depends on NOT copying this document in any way. This tape will

<a HREF="http://www.mysite.com/selfdestruct.html">self destruct</a> in 10 seconds...</h6>

    </body>

    </html>
```

Inline Styles are not Best Practice

Inline styles, while they have a purpose, are not the best way to maintain your Web site. They go against every one of the above benefits:

- ***Inline styles don't separate content from design:*** In fact, inline styles are exactly the same as embedded font tags and other design tags that we're trying to stop using. The styles only affect the exact tag they are applied to, and while that might give you more control, it makes other aspects of your design and development more difficult.
- ***Inline styles cause more maintenance headaches:*** When you're working with style sheets, it can sometimes be very difficult to figure out where a style is being set. When you add a mixture of inline, embedded, Linked CSS and Importing CSS and you have even more locations to look. Again, if you work on a Web design team or have to redesign or maintain a site built by someone else, then you're going to have even more trouble. Then, once you find the style and get rid of it, you'll have to get rid of it on every element on every page where it's been placed, which can increase your maintenance work astronomically.
- ***Inline styles are not as accessible:*** While a screen reader or other assistive device might be able to handle the attributes and tags effectively, some of the older devices don't and can result in some strange Web pages.

Caution In addition, the extra characters and text can affect how your page is viewed by a robot such as a search engine, so your page optimization would not do as well as a page with external style sheets.

- ***Inline styles make your pages bigger:*** If you set a style on every paragraph on your site, you can do it once with like 6 lines of code and an external style sheet. However, if you do it with an inline style, you'll have to add those styles to every paragraph of your site. If you have, 5 lines of CSS that's 5 lines are multiplied by every paragraph on your site. That bandwidth can add up in a hurry.

Notes

4.5.2 Embedded CSS or Internal Styles

When applying internal CSS you put all your CSS rules in the <head>...</head> part of your HTML document and enclose it with <style type="text/css">...</style> tags. This CSS style specification method is only used with HTML. An entire style sheet can be embedded in an HTML document using the <STYLE> element contained within the <HEAD> block. The complete range of CSS syntax is not explicitly tied directly to the document's elements, so Selector syntax is used to specify what styles attach to which portions of the document tree (the same as with External Style Sheets, but in this case the style sheet is contained within the document itself).

Unlike external style sheets, this method can only specify style information for the current document. If equivalent style rules are specified in an external and embedded style sheet, the embedded style sheet rule will have higher precedence.

```
Syntax:
      <style TYPE="text/css">
<!—
[CSS Style Sheet]
—>
</style>
```

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <head>
      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
      <title>Internal CSS styles</title>
      <style type="text/css">
        p {border:2px solid #8dc919}
     </style>
   </head>
   <body>
      <p>1st paragraph ...</p>
      <p>2nd paragraph ...</p>
   </body>
</html>
```

Usage Guidelines:

1. Avoid if possible: Really, there's not much point using internal CSS styles. You might as well take all those CSS rules and put them into an external .css file so it can be re-used in other pages.

   [See complex HTML example using Internal Style Sheets]

```
<html>
       <head>
                <title>Document Title</title>
         <style TYPE="text/css">
```

Notes

```
<!—
body {
            background: black;
            color: #80c0c0}
 a:link {color: #ff8080}
 a:visited {color: #ff0000}
 a:active {color: #a05050}
  a.case1:link {background: green}
 p:first-line { margin-left: 25px }
 div.foo:first-line {
            font-weight: bold;
            margin-left: 30px}
 div:first-letter {
            font-size: x-large;
            color: #ffffff}
ul ul li {
            font-size: x-large;
             font-decoration: italic}
 h2 em {font-weight: 900}
h2.ex1 {
            color: green;        /* This type of heading MUST be Green
and Large! */
            line-height: 50px;
            font-size: 40px}
 .funkyclass {
             font: 36pt/40pt courier;
             font-variant: small-caps;
             border: thick dashed blue}
#tagid1 {
            padding: 20px;
             border: 20px groove #ffffff}
  .class1, #tagid2 {font-family: 'comic sans ms', fantasy; color: rgb
(100%,100%,0%)}
   h1, h2, div.class5, blockquote {background: #000080}
p.special {
             font: 12pt/14pt sans-serif;
            margin: 5px 0px 2px 25px;
             border: medium dashed #ff0000;
    background: white url (http://www.foo.com/image.gif) repeat-x fixed top
right}
```

Notes

```
blockquote {
margin-left: 2cm;
color: #00ff00}
.part1 {
font-size: xx-large;
color: #808000}
h6 {
font-size: xx-small ! important;
color: red ! important}
—>
</style>
</head>
<body>
<h1 CLASS="funkyclass" ALIGN="center">Welcome to my home page!</h1>
<br><br>
```

Hi there! If you are reading this then you have found my home page! Congratulations! I know it can be hard to find my pages, but I bet that you feel lucky now. Now that you are here, please take a look at my page of links to <a HREF="http://www.mysite.com/coolsites.html">cool sites or sign my <a HREF="http://www.mysite.com/guestbook.html">guest book

```
<div CLASS="foo"> My wonderful poetry <br> is available if you are REALLY bored. Why not give it a spin?</div>
<h2 CLASS="ex1"> The Best Poetry I <em>NEVER</em> Wrote</h2>
<ul>
<li>'There Once Was A Man From Nantucket' - <span CLASS="class1"> Anonymous</span></li>
<li>'Cool In Fur' - <span CLASS="class1">Harry B. Foot</span></li>
<li>And My All Time Fave:
<ul>
<li> 'A Toast To My Toaster' - <span CLASS="class1">Me!</span></li>
</ul>
</li></ul>
<blockquote>Brought to you by the letter <span ID="tagid2">&quot;H&quot;</span> and <span ID="tagid1">Joe Shmoe</span> </blockquote>
<div CLASS="class5"> When you are done looking through these masterpieces, I encourage you to visit my proud sponsor!! </div>
```

ADVERTISEMENT: Buy Navel Lint Contemplator! It's a browser and it's a sandwich spread! Go to our <a HREF="http://www.navellint.com">home page without delay! Why? Because shelf life is only 8 hours unless refrigerated. We know that makes it hard to browse, but it promotes frequent upgrading to the latest and greatest version.

Notes

<h6>All standard disclaimers apply. Your life depends on NOT copying this document in any way. This tape will <a HREF="http://www.mysite.com/selfdestruct.html" CLASS="case1">self destruct in 10 seconds...</h6>

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

4.5.3 External Style Sheets/Linked CSS

The true power of this CSS specification mechanism is separating presentation from content. External Style Sheets are the tool for doing exactly this by putting all your CSS rules into a separate .css file (a simple text file) you can reuse them for all your pages without rewriting.

Notes All your styles are in a centralized place making it easy to maintain and change the look of your website.

This is the most powerful of all the CSS attachment methods, as it allows a single style sheet to control the rendering of multiple documents. These results in a timesaving for the author, a savings of space for the web server, and less download time for the user. In addition, this method can be used in both HTML and XML.

An External Style Sheet is a file containing only CSS syntax (no document content or elements) and should carry a MIME type of "text/css". The style information is not explicitly tied directly to the document's elements, so Selector syntax is used to specify what styles attach to which portions of the document tree. The full range of CSS syntax is allowed in this method.

```
Syntax:
    <link REL="STYLESHEET" TYPE="text/css" HREF="[Style sheet URL]">
```

LINK Attribute	Description
type:	This is the MIME content-type and should always be set to text/css for stylesheets.
rel:	This attribute describes the relationship between the actual document and the one you're linking to. In this case, it's always stylesheet.
media:	You might want to provide more than one stylesheet for different presentations of your website. For example a normal one, a high-contrast one for people with reading difficulties, a printer-friendly one for printing and so on. If the client supports this, it will use the appropriate stylesheet. If you just have a single stylesheet for everything, you can omit this attribute or use all.
href:	This is the link to your stylesheet.

Usage Guidelines:

1. ***Use as much as possible:*** Try to put all your CSS styles into stylesheets and minimize your inline CSS and internal CSS.
2. ***Use a logical structure:*** As a CSS beginner you probably won't care much about the order in which you write your CSS rules, because your stylesheets will be relatively small. As they get bigger, though, it can help having a logical structure in your stylesheet. For example you could group general selectors such as <BODY>, <A>, <P>, <IMG>, <H1>, ... at the beginning and then follow with contextual selector rules. Or group CSS styles together, which belong to a certain layout part of the page such as the header, sidebar, footer and so on.

Notes

3. ***Naming your stylesheet:*** It's an unofficial naming convention that the main stylesheet is called style.css. It can't hurt adopting this yourself.

4. ***Use F5 to display recent changes:*** When you're making changes to your stylesheets you have to press the F5 key to force the web browser to load the newest version instead of using the cached one.

Creation of .css File

Let us get started by making that external CSS file. Open up notepad.exe or any other plain text editor and type the following CSS code.

```
CSS Code: (test.css)
body{background-color: gray}
p {color: blue;}
h3{color: white;}
```

Now save the file as a CSS (.css) file. Make sure that you are not saving it as a text (.txt) file, as notepad likes to do by default. Name the file "test.css" (without the quotes). Now create a new HTML file and fill it with the following code.

```
HTML Code: (index.htm)
<html>
<head>
<link rel="stylesheet" media="all" type="text/css" href="test.css" />
</head>
<body>
<h3> A White Header </h3>
<p> This paragraph has a blue font.
The background color of this page is gray because
we changed it with CSS! </p>
</body>
</html>
```

Task What is the major difference between the Inline CSS and Embedded CSS?

Save this file as "index.htm" (without the quotes) in the same directory as your CSS file. Now open your HTML file in your web browser and it should look something like…

Display:

[See complex HTML example using External Style Sheets]

```
External File: 'example.css'
        body {
                    background: black;
                    color: #80c0c0}
         a:link {color: #ff8080}
```

Notes

```
a:visited {color: #ff0000}
a:active {color: #a05050}
a.case1:link {background: green}
p:first-line {margin-left: 25px }
div.foo:first-line {
            font-weight: bold;
            margin-left: 30px}
div:first-letter {
            font-size: x-large;
            color: #ffffff}
ul ul li {
            font-size: x-large;
            font-decoration: italic}
h2 em {font-weight: 900}
h2.ex1 {
    color: green; /* This type of heading MUST be Green and Large! */
            line-height: 50px;
            font-size: 40px}
.funkyclass {
            font: 36pt/40pt courier;
            font-variant: small-caps;
            border: thick dashed blue}
#tagid1 {
            padding: 20px;
            border: 20px groove #ffffff}
.class1, #tagid2 {font-family: 'comic sans ms', fantasy; color: rgb
(100%,100%,0%)}
h1, h2, div.class5, blockquote {background: #000080}
p.special {
            font: 12pt/14pt sans-serif;
            margin: 5px 0px 2px 25px;
            border: medium dashed #ff0000;
background: white url (http://www.foo.com/image.gif) repeat-
x fixed top right}
blockquote {
            margin-left: 2cm;
            color: #00ff00}
.part1 {
            font-size: xx-large;
            color: #808000}
```

Notes

```
h6 {
font-size: xx-small ! important;
color: red ! important}
```

Main Document: 'example.htm'

```
<html>
<head>
<title>Document Title</title>
<link REL="StyleSheet" TYPE="text/css" HREF="example.css">
</head>
<body>
<h1 CLASS="funkyclass" ALIGN="center">Welcome to my home page!</h1>
<br><br>
<p>Hi there! If you are reading this then you have found my home page! Congratulations! We know it can be hard to find my pages, but I bet that you feel lucky now. Now that you are here, please take a look at my page of links to <a HREF="http://www.mysite.com/coolsites.html">cool sites</a> or sign my <a HREF="http://www.mysite.com/guestbook.html">guest book</a></p>
<div CLASS="foo"> My wonderful poetry <br> is available if you are REALLY bored. Why not give it a spin?</div>
<h2 CLASS="ex1"> The Best Poetry I <em>NEVER</em> Wrote</h2>
<ul>
<li>'There Once Was A Man From Nantucket' - <span CLASS="class1">Anonymous</span></li>
<li>'Cool In Fur' - <span CLASS="class1">Harry B. Foot</span></li>
<li>And My All Time Fave:
<ul>
<li> 'A Toast To My Toaster' - <span CLASS="class1">Me!</span></li>
</ul>
</li></ul>
<blockquote>Brought to you by the letter <span ID="tagid2">&quot;H&quot;</span> and <span ID="tagid1">Joe Shmoe</span> </blockquote>
<div CLASS="class5"> When you are done looking through these masterpieces, I encourage you to visit my proud sponsor!! </div>
```

ADVERTISEMENT: Buy Navel Lint Contemplator! It's a browser and it's a sandwich spread! Go to our <a HREF="http://www.navellint.com">home page without delay! Why? Because shelf life is only 8 hours unless refrigerated. We know that makes it hard to browse, but it promotes frequent upgrading to the latest and greatest version.

<h6>All standard disclaimers apply. Your life depends on NOT copying this document in any way. This tape will <a HREF="http://www.mysite.com/selfdestruct.html" CLASS="case1">self destruct in 10 seconds...</h6>

Notes

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

Benefits of External CSS

- It keeps your website design and content separate.
- It's much easier to reuse your CSS code if you have it in a separate file. Instead of typing the same CSS code on every web page you have, simply have many pages refer to a single CSS file with the "link" tag.
- You can make drastic changes to your web pages with just a few changes in a single CSS file.
- Help to achieve the consistency in the styles through out the pages.

Did u know? What is the limitation of External CSS?

If the user downloads the page and if he forgot to download the Linked CSS then he/she will not be able to view the page with all the styles.

4.5.4 Importing Style Sheets

The @import rule thus allows you to keep some things the same while having others different.

Syntax:

```
@import url (nameoffile.css)
```

It must come at the start of the style sheet, before any rulesets (a ruleset is something like P {color: red}). Alternatively, it can be specified as:

Syntax:

```
@import "nameoffile.css"
as @import url ("nameoffile.css")
or as @import 'nameoffile.css'
```

However, Internet Explorer only supports the url() formats, not the " and ' formats.

Self Assessment

Fill in the blanks:

10. The simplest and most direct way of applying CSS to an element is to write it into the tag itself as a style attribute ……...
11. An entire style sheet can be embedded in an HTML document using the <STYLE> element contained within the ……... block.
12. If you set a style on every paragraph on your site, you can do it once with like 6 lines of code and an ……...

Notes

4.6 Limitations of CSS

- Limited transcription capabilities
  - Limited transposition of elements (float:left/right)
  - Calls of parameterized formatting tasks the major transcription type supported
- In CSS1 context specification limited:
  - No sibling or parent/child relationships
  - Limited use of attributes (CSS1: only class)
  - CSS2 more powerful,
    - But no access to element's children or content
    - Unable to access targets of cross references (?)
- Non-programmable
  - No decision structures
  - Unable to store calculated quantities
  - Non-extensible
  - Relatively simple
  - Western-language orientation (left-to-right)
  - XSL allows unrestricted transformations of the document to precede a CSS-like formatting

4.7 Advantages of CSS

- Workflow
  - Faster downloads
  - Streamlined site maintenance
  - Global control of design attributes
  - Precise control (Advanced)
    - Positioning
    - Fluid layouts
- Cost Savings
  - Reduced Bandwidth Costs
    - One style sheet called and cached
  - Higher Search Engine Rankings
    - Cleaner code is easier for search engines to index
    - Greater density of indexable content
- Faster download = better usability
  - Web usability redesign can

Notes

- CSS requires less code
- Tables require spacer images
- Entire table has to render before content
- CSS can control the order that elements download (content before images).

- Increased Reach
  - CSS website is compatible with many different devices
  - In 2008 an est. 58 Million PDA's will be sold (Source: eTForecast.com) 1/3 of the world's population will own a wireless device by 2010.

Cascading Style allows obtaining the full control under HTML tagging. CSS allows easily redefining the all default properties of any HTML tag. Using CSS, you will open new unique opportunities missing in common HTML.

Task Analyze the non programmable issues in CSS

Self Assessment

Fill in the blanks:

13. …….. allows unrestricted transformations of the document to precede a CSS-like formatting.
14. ……..can control the order that elements download (content before images).

4.8 Style Tag

The <style> tag is used to define style information for an HTML document.

Inside the style element you specify how HTML elements should render in a browser.

The required type attribute defines the content of the style element. The only possible value is "text/css".

The style element always goes inside the head section.

```
<html>
<head>
<style type="text/css">
h1 {color:red;}
p {color:blue;}
</style>
</head>
<body>
<h1>Header 1</h1>
<p>A paragraph.</p>
</body>
</html>
```

Notes

4.9 DIV and SPAN

The tag defines logical divisions (defined) in your Web page. It acts a lot like a paragraph tag, but it divides the page up into larger sections.

 also gives you the chance to define the style of whole sections of HTML. You could define a section of your page as a call out and give that section a different style from the surrounding text.

But that's not all it does! The tag gives you the ability to name certain sections of your documents so that you can affect them with style sheets or Dynamic HTML.

One thing to keep in mind when using the tag is that it breaks paragraphs. It acts as a paragraph end/beginning, and while you can have paragraphs within a you can't have a inside a paragraph.

The primary attributes of the tag are:

- style
- class
- id

Even if you don't use style sheets or DHTML, you should get into the habit of using the tag. This will give you more flexibility when more XML parsers become available. Also, you can use the id and name attributes to name your sections so that your Web pages are well formed (always use the name attribute with the id attribute and give them the same contents).

Because the tag has been deprecated in HTML 4.0, it is a good idea to start using

```
<div style="text-align: center;">
```

to center the content inside your div.

```
<span>
```

The tag has very similar properties to the tag, in that it changes the style of the text it encloses. But without any style attributes, the tag won't change the enclosed items at all.

The primary difference between the and tags is that doesn't do any formatting of it's own. The tag acts includes a paragraph break, because it is defining a logical division in the document. The tag simply tells the browser to apply the style rules to whatever is within the .

The tag has no required attributes, but the three that are the most useful are:

- style
- class
- id

Use when you want to change the style of elements without placing them in a new block-level element in the document. For example, if you had a Level 3 Heading (<h3>) that you wanted the second word to be red, you could surround that word with

```
<span style="color : #f00;">2ndWord</span>
```

and it would still be a part of the <h3> tag, just red.

Notes

Task Substantiate the primary difference between the and tags.

Self Assessment

Fill in the blanks:

15. The …….. tag is used to define style information for an HTML document.
16. Use when you want to change the style of elements without placing them in a new …….. element in the document.

4.10 Creating and Using CSS Classes

Using classes with your CSS can make customizing your pages ten times easier, because you can give different occurances of the same element different styles.

Say you wanted every cell of a table to have a different color and font, like this:

pink	green
yellow	blue

Instead of defining everything with HTML, we use CSS classes. Here is the code:

```
<table border="1"><tr>
<td class="pink">pink</td>
<td class="green">green</td>
</tr><tr>
<td class="yellow">yellow</td>
<td class="blue">blue</td>
</tr></table>
```

Then, in your header, you'd define what happens with each different class, like so:

```
<style type="text/css"><!—
.pink {FONT-FAMILY: Times New Roman, Times; BACKGROUND: #FFCCFF}
.green {FONT-FAMILY: Courier New, Courier; BACKGROUND: #33FFCC}
.blue {FONT-FAMILY: Verdana; BACKGROUND: #00CCFF}
.yellow {FONT-FAMILY: Arial, Helvetica; BACKGROUND: #FFFF66}
—></style>
```

Notes Now everything you add class="pink" to will have a pink background color and Times New Roman font.

This can also be useful if you want different links to look different ways. For example, links in the menu to look one way, and links in the body/text to be different. In the style sheet of your page you'd usually have this:

```
a:active {color:pink}
a:visited {color:pink}
```

Notes

```
a:link {color:pink}
a:hover {color:blue}
```

If you had a few links you wanted to be a bit different, you'd add this:

```
a.menu:active {color:blue}
a.menu:visited {color:blue}
a.menu:link {color:blue}
a.menu:hover {color:pink}
```

Then add class="menu" to the links you'd like different, like this:

```
<a href="page.html" class="menu">link</a>
```

Using classes is simple. You just need to add an extension to the typical CSS code and make sure you specify this extension in your HTML. Let's try this with an example of making two paragraphs that behave differently. First, we begin with the CSS code, note the red text.

CSS Code:

```
p.first{ color: blue; }
p.second{ color: red; }
HTML Code:
<html>
<body>
<p>This is a normal paragraph.</p>

<p class="first">This is a paragraph that uses the p.first CSS code!</p>
<p class="second">This is a paragraph that uses the p.second CSS code!</p>
...
```

Display:

This is a normal paragraph.

This is a paragraph that uses the p.first CSS code!

This is a paragraph that uses the p.second CSS code!

You can use CSS classes with any HTML element! However, what happens if we had already defined a value for the default <p> tag, would this cause any problems for classes of the paragraph tag?

Well, when this happens the CSS class for any <p> tag will override the default <p> CSS. If the CSS class uses a CSS attribute already defined by the default CSS, then the formatting defined by the class will be the value that is used.

It may be easier to imagine that the CSS for a generic HTML element is the starting point and the only way to change that look is to overwrite the attributes using CSS classes. Please see the example below for a visual of this tricky topic.

Example:

CSS Code:

```
p{ color: red; font-size: 20px; }
```

Notes

```
p.test1{ color: blue; }
p.test2{ font-size: 12px; }
HTML Code:
<html>
<body>
<p>This is a normal paragraph.</p>
<p class="test1">This is a paragraph that uses the p.test1 CSS code!</p>
<p class="test2">This is a paragraph that uses the p.test2 CSS code!</p>
...
```

Display:

This is a normal paragraph.

This is a paragraph that uses the p.test1 CSS code! The p.test1 paragraph remained the same size, but it's color changed.

This is a paragraph that uses the p.test2 CSS code! The p.test2 paragraph remained the same color, but it's size changed.

Self Assessment

Fill in the blanks:

17. If the CSS class uses a CSS …… already defined by the default CSS, then the formatting defined by the class will be the value that is used.

Caselet

CSS will Come out with IPO in India

CSS Corp, the US-based IT company, will consider coming out with an initial public offer (IPO) in India but has not fixed any time frame for it. "We can list anywhere, including the US, London and India. But at this stage it will be in India," according to Mr Nick Sharma, CEO, CSS Corp.

Backed by financial institutions, including SAIF Partners, Goldman Sachs and Sierra Ventures, CSS Corp has revenue of around $135 million. Anything over $100 million is enough to go for IPO, Mr Sharma told newspersons while announcing the company's new brand identity and inaugurating a new centre in Chennai. "We have not set any target for the IPO but it can be any time," he said.

Financial investors together hold over 50 per cent stake in the company, he said.

Revenue Growth

According to Mr Sharma, CSS hopes to add nearly $100 million in revenue in the next two years to become a $250-million company by 2012-end. Nearly 25 per cent of the revenue has come from acquisitions and this inorganic strategy will continue, he said.

"We are confident of achieving revenue of $250 million with the global economy picking up," he said. At present nearly 85 per cent of the company's revenue comes from the US.

Contd...

Notes

"We want to reduce our dependence on the US market to around 65 per cent and grow in other markets such as Europe and Asia Pacific," he said.

Consolidation

CSS that has around 5,000 employees across 20 locations globally inaugurated its new facility at Ambit Park in Chennai.

The company has invested nearly $5 million in the new facility taken on lease.

It will consolidate all its operations in Chennai and move people to the new facility in which CSS has taken two floors with options to have more space in the building. It will move all of its 2,500 employees in the city to the new premise, he said.

"We are starting the next decade of aggressive international expansion that will see the company double in size by 2012," he said.

The brand was evolved from its Cybernet, Slash-Support and Ready Test Go to the three-letter acronym of CSS. With new acquisitions (three in the last one year), there was a need to unite the offerings from various brands in to a single brand to deliver an end-to-end information and communication technology services company, he said.

4.11 Summary

- This unit has taught you how to create style sheets to control the style.
- "CSS Web Template" is a website design created using Cascading Style Sheets (CSS) technology. Cascading style sheets provide web developers an easy way to format and to style web pages.
- In this unit, you will learn how to make your first style sheet.
- You have learned how to use CSS to add backgrounds, format text, add and format borders, and specify padding and margins of elements.
- You have also learned how to position an element, control the visibility and size of an element, set the shape of an element, place an element behind another, and to add special effects to some selectors, like links.
- A CSS declaration represents the effect to be applied to the element(s) and it consists of two more parts: a property and a value.
- Every property has its own syntax and constraints on the values it accepts. Property values also often indicate measurement units to aid in rendering to the specified media.
- The tag has very similar properties to the tag, in that it changes the style of the text it encloses.
- Using classes is simple. You just need to add an extension to the typical CSS code and make sure you specify this extension in your HTML.

4.12 Keywords

CSS: Cascading Style Sheets

HTML: Hypertext Markup Language

XSLT: Extensible Stylesheet Language Transformations

Notes

4.13 Review Questions

1. Explain why CSS technology is a great step forward in web development?
2. Style Sheets are the easiest way to provide a default font styling for HTML. Do you agree with this statement? Explain with examples to support your answer.
3. Cascading style sheets provide web developers an easy way to format and to style web pages. Analyze
4. A Property is an identifier. Give reasons
5. Substantiate why inline Styles Are Not Best Practice?
6. gives you the chance to define the style of whole sections of HTML. Explain and give example
7. Using classes with your CSS can make customizing your pages ten times easier. Explain why?
8. Can Style Sheets and HTML stylistic elements be used in the same document?
9. What do you mean by CSS declaration and CSS properties? Give examples.
10. List out the different ways of implementing CSS.

Answers: Self Assessment

1.	template	2.	Style Sheets
3.	Cascading Style Sheets (CSS)	4.	constraints
5.	glyphs	6.	rendering
7.	content	8.	center
9.	device	10.	style="..."
11.	<HEAD>	12.	external style sheet
13.	XSL	14.	CSS
15.	<style>	16.	block-level
17.	attribute		

4.14 Further Readings

Books

Core CSS: cascading style sheets,

Hakon Wium Lie, Håkon Wium Lie, Bert Bos, *Cascading style sheets: designing for the Web,* Addison-Wesley Professional

Ian Pouncey, Richard York, Ian Pouncey, *Beginning CSS: Cascading Style Sheets for Web Design,* Richard York

Notes

Online links

http://www.w3.org/Style/CSS/Overview.en.html

http://htmlhelp.com/reference/css/

Unit 5: Scripting Language

Notes

CONTENTS

Objectives

Introduction

5.1 Java Script Programming

5.1.1 Advantages of JavaScript

5.1.2 Disadvantages of JavaScript

5.1.3 Creating Our First Program in JavaScript

5.2 Java Script Data Types

5.2.1 Primitive Data Types

5.2.2 Composite Data Types

5.3 Variables

5.3.1 Declaring vs. Initializing Variables and Null

5.3.2 JavaScript Variable Naming Conventions

5.3.3 Variable Scope

5.4 Arrays

5.5 Operators

5.6 JavaScript Loops

5.6.1 The for Loop

5.6.2 The while Loop

5.6.3 The do...while Loop

5.6.4 Break

5.6.5 Continue

5.7 JavaScript Objects

5.7.1 String Object

5.7.2 Date Object

5.7.3 Boolean Object

5.7.4 Math Object

5.7.5 JavaScript Functions

5.8 Dialog Boxes

5.9 String Manipulation Functions

5.10 Using Timer in Web Page

5.11 Summary

5.12 Keywords

Contd...

Notes

5.13 Review Questions

5.14 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain the Java Script programming
- Scan the data types
- Describe Variables
- Demonstrate Arrays and Operators
- Recognize Loops and Functions
- Explain Dialog Boxes and String Manipulation Function
- Scan the use of timer in web pages
- Discuss setting and getting date in a web page

Introduction

Java script is a browser-interpreted language that was created to access all elements of HTML and the browser. The processing is done entirely by the client-side browser which makes it very useful tool to handle processing which would have otherwise been checked server-side, thereby reducing overhead. Java script is also used to increase user interaction, animate objects, create drop down navigation, grab data from databases, and more!

JavaScript is most commonly used as a client side scripting language. This means that JavaScript 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 my 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.

JavaScript can be used in other contexts than a Web browser. Netscape created server-side JavaScript as a CGI-language that can do roughly the same as Perl or ASP.

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 full 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. A large part of this site is devoted to exploring and explaining these browser differences and finding ways to cope with them.

Notes

Notes Basic JavaScript is easy to learn, but when you start writing advanced scripts browser differences (and occasionally syntactic problems) will creep up.

5.1 JavaScript Programming

JavaScript is *not* the same as Java. Although the names are much alike, JavaScript 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.

Did u know? **Who developed JavaScript?**

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 (and slightly hypey). 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.

5.1.1 Advantages of JavaScript

- JavaScript gives HTML designers a programming tool – HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages.
- JavaScript can put dynamic text into an HTML page – A JavaScript statement like this: document. write ("<h1>" + name + "</h1>") can write a variable text into an HTML page.
- 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 saves the server from extra processing.
- JavaScript can be used to detect the visitor's browser – A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser.
- JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer.

Notes

5.1.2 Disadvantages of JavaScript

- If you lack experience in programming, JavaScript will look daunting to you. Often, it is not clear how the code works when you examine the HTML source. Long, complicated Java scripts can add quite a bit of download time to your HTML page.
- Although JavaScript is supported on the two major web browsers, there are a few differences that may cause problems. Many of the graphic HTML creation tools do not handle JavaScript very well.
- Because the code executes on the users' computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable JavaScript.

Task "JavaScript validates data". Is this a advantage or disadvantage? Analyze

5.1.3 Creating Our First Program in JavaScript

Save the Example 4.1 as MyExample.html and run it in same manner as HTML.

```
<html>
<body>
<script type="text/javascript">
document.write("this is my first program!");
</script>
</body>
</html>
Output
this is my first program!
```

Self Assessment

Fill in the blanks:

1. JavaScript is primarily a scripting language for use within …… pages.
2. A JavaScript can be used to validate form data before it is submitted to a ……
3. Many of the graphic HTML creation tools do not handle …… very well.

5.2 JavaScript Data Types

A program can do many things, including calculations, sorting names, preparing phone lists, displaying images, validating forms, ad infinitum. But in order to do anything, the program works with the data that is given to it.

Data types specify what kind of data, such as numbers and characters, can be stored and manipulated within a program. JavaScript supports a number of fundamental data types. These types can be broken down into two categories, primitive data types and composite data types.

Notes

Did u know? **What is the basic use of data type?**

Undeclared variables and variables declared without a data type are assigned the Object data type. This makes it easy to write programs quickly, but it can cause them to execute more slowly.

5.2.1 Primitive Data Types

Primitive data types are the simplest building blocks of a program. They are types that can be assigned a single literal value such as the number 5.7, or a string of characters such as "hello". JavaScript supports three core or basic data types:

- numeric
- string
- Boolean

In addition to the three core data types, there are two other special types that consist of a single value:

- null
- undefined
- Numeric Literals

  JavaScript supports both integers and floating-point numbers. Integers are whole numbers and do not contain a decimal point; e.g., 123 and –6. Integers can be expressed in decimal (base 10), octal (base 8), and hexadecimal (base 16), and are either positive or negative values.

  Floating-point numbers are fractional numbers such as 123.56 or –2.5. They must contain a decimal point or an exponent specifier, such as 1.3e–2. The letter "e" for exponent notation can be either uppercase or lowercase.

  JavaScript numbers can be very large (e.g., 10308 or 10–308).

Table 5.1

12345	Integer
23.45	float
.234E–2	scientific notation
.234e+3	scientific notation
0x456fff	hexadecimal
0x456FFF	hexadecimal
0777	Octal

- String Literals and Quoting

  String literals are rows of characters enclosed in either double or single quotes. The quotes must be matched. If the string starts with a single quote, it must end with a matching single quote, and likewise if it starts with a double quote, it must end with a double quote. Single quotes can hide double quotes, and double quotes can hide single quotes:

Notes

```
"This is a string"
'This is another string'
"This is also 'a string' "
This is "a string"'
```

An empty set of quotes is called the null string. If a number is enclosed in quotes, it is considered a string; e.g., *"5"* is a string, whereas *5* is a number.

Strings are called constants or literals. The string value "hello" is called a string constant or literal.

> *Notes* To change a string requires replacing it with another string.

Strings can contain escape sequences (a single character preceded with a backslash), as shown in Table 5.2. Escape sequences are a mechanism for quoting a single character.

Table 5.2: Escape Sequences

Escape Sequence	What It Represents
\'	Single quotation mark
\"	Double quotation mark
t	Tab
n	Newline
r	Return
f	Form feed
b	Backspace
e	Escape
\\	Backslash

Special Escape Sequences

XXX	The character with the Latin-1 encoding specified by up to three octal digits XXX between 0 and 377. \251 is the octal sequence for the copyright symbol.
xXX	The character with the Latin-1 encoding specified by the two hexadecimal digits XX between 00 and FF. \xA9 is the hexadecimal sequence for the copyright symbol.
uXXXX	The Unicode character specified by the four hexadecimal digits XXXX. \u00A9 is the Unicode sequence for the copyright symbol.

Example: Illustrating strings in java script.

```
<html>
<head>
<body>
```

Notes

```
<pre>
  <font size="+2">
     <script language="JavaScript">
  <!— Hide script from old browsers.
     document.write("\t\tHello\nworld!\n");
     document.writeln("\"Nice day, Mate.\"\n");
     document.writeln('Smiley face:<font size="+3"> \u263a\n');
  //End hiding here. —>
  </script>
  </pre>
  </body>
  </html>
output
   Hello
world!
"Nice day, Mate."
Smiley face: :&
```

Explanation

1. The escape sequences will work only if in a <pre> tag or an alert dialog box.
2. The JavaScript program starts here.
3. The write() method sends to the browser a string containing two tabs (\t\t), Hello, a newline (\n), world!, and another newline (\n).
4. The writeln() method sends to the browser a string containing a double quote (\"), Nice day, Mate., another double quote (\"), and a newline (\n). Since the writeln() method automatically creates a newline, the output will display two newlines: the default value and the \n in the string.
5. This string contains a backslash sequence that will be translated into Unicode. The Unicode hexidecimal character 233a is preceded by a \u.

The process of joining strings together is called concatenation. The string concatenation operator is a plus sign (+). Its operands are two strings. If one string is a number and the other is a string, JavaScript will still concatenate them as strings. If both operands are numbers, the + will be the addition operator.

The following examples output "popcorn" and "Route 66", respectively.

```
document.write("pop" + "corn:");
document.write("Route" + 66);
```

The expression *5 + 100* results in *105*, whereas *"5" + 100* results in *"5100"*.

Boolean Literals

Boolean literals are logical values that have only one of two values, true or false. You can think of the values as yes or no, on or off, or 1 or 0. They are used to test whether a condition is true or

Notes

false. When using numeric comparison and equality operators, the value true evaluates to 1 and false evaluates to 0.

```
answer1 = true;
        or
if (answer2 == false) { do something; }
```

5.2.2 Composite Data Types

We mentioned that there are two types of data: primitive and composite. The primitive types: numbers, strings and Booleans—each storing a single value. Composite data types, also called complex types, consist of more than one component. Objects, arrays, and functions, covered later in this book, all contain a collection of components.

- Objects contain properties and methods
- Arrays contain a sequential list of elements
- Functions contain a collection of statements

Self Assessment

Fill in the blanks:

4. …….. specify what kind of data, such as numbers and characters, can be stored and manipulated within a program.
5. String literals are rows of characters enclosed in either …….. quotes.
6. The process of joining strings together is called ……..
7. Composite data types, also called …….. types, consist of more than one component.

5.3 Variables

A variable's purpose is to store information so that it can be used later. A variable is a symbolic name that represents some data that you set. To think of a variable name in real world terms, picture that the name is a grocery bag and the data it represents are the groceries. The name wraps up the data so you can move it around a lot easier, but the name is not the data!

When using a variable for the first time it is not necessary to use "var" before the variable name, but it is a good programming practice to make it crystal clear when a variable is being used for the first time in the program. Here in the example VarExample.html we are showing how the same variable can take on different values throughout a script.

Example: Illustrating variables in java script

```
<html>
<body>
<script type="text/JavaScript">
<!—
var linebreak = "<br />"
var my_var = "Hello World!"
document.write(my_var)
```

Notes

```
document.write(linebreak)
my_var = "I am learning JavaScript!"
document.write(my_var)
document.write(linebreak)
my_var = "Script is Finishing up..."
document.write(my_var)
//—>
</script>
</body>
</html>
Output
Hello World!
I am learning JavaScript!
Script is Finishing up...
```

5.3.1 Declaring vs. Initializing Variables and Null

To create a variable, the var keyword precedes the variable name, and is used only once for declaration. All future references to the variable are made without the var keyword.

Variables can be assigned values later on, or immediately by following the name with an equals sign, then the value they represent. Assigning a value to a variable immediately is known as initializing the variable.

Caution If a variable is not initialized, it has a value of null and has only been declared.

var myVarName *// declare variable, has null value*

myVarName = 162 *// assign a value, null value is replaced*

//OR

var myVarName = 162 *// declare AND assign value (initialize)*

myVarName = "two hundred" *// the value can be changed later on, even to a different type of value*

var time,dog,baby *//multiple variables may be declared simultaneously*

var time = 1, dog = "hairy", baby = true *//multiple variables may be initialized simultaneously*

5.3.2 JavaScript Variable Naming Conventions

When choosing a variable name, you must first be sure that you do not use any of the JavaScript reserved names found here. Another good practice is choosing variable names that are descriptive of what the variable holds. If you have a variable that holds the name of a month, then name it "month_name" to make your JavaScript more readable.

Finally, JavaScript variable names may not start with a numeral (0-9). These variable names would be illegal: 7lucky, 99bottlesofbeer, and 3zacharm.

Notes

A good rule of thumb is to have your variable names start with a lowercase letter (a-z) and use underscores to separate a name with multiple words (i.e. my_var, strong_man, happy_coder, etc).

5.3.3 Variable Scope

Variable scope has to do with where a variable can legally be used, and is determined by where it was originally declared or initialized.

A variable declared or initialized outside a function body has a global scope, making it accessible to all other statements within the same document.

A variable declared or initialized within a function body has a local scope, making it accessible only to statements within the same function body.

Self Assessment

Fill in the blanks:

8. A …….. is a symbolic name that represents some data that you set.
9. To create a variable, the …….. keyword precedes the variable name, and is used only once for declaration.

5.4 Arrays

An array is a special variable, which can hold more than one value, at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

cars1=”mecedes”;

cars2=”ferari”;

cars3=”BMW”;

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Did u know? What are the advantages of Arrays in all the programming languages?

Advantages

1. You can use one name for similar objects and save then with the same name but different indexes.
2. Arrays are very useful when you are working with sequences of the same kind of data (similar to the first point but has a different meaning).
3. Arrays use reference type and so.

Notes

Each element in the array has its own ID so that it can be easily accessed.

- Create an Array

  An array can be defined in three ways.

  The following code creates an Array object called myCars:

  1:

  var myCars=new Array(); // regular array (add an optional integer

  myCars[0]="Mercedes"; // argument to control array's size)

  myCars[1]="Ferari";

  myCars[2]="BMW";

  2:

  var myCars=new Array("Mercedes","Ferari","BMW"); // condensed array

  3:

  var myCars=["Mercedes","Ferari","BMW"]; // literal array

> *Notes* Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.

- Access an Array

  You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

  The following code line:

  document.write(myCars[0]);

  Will result in the following output:

  Mercedes

- Modify Values in an Array

  To modify a value in an existing array, just add a new value to the array with a specified Index number:

  myCars[0]="Opel";

  Now, the following code line:

  document.write(myCars[0]);

  will result in the following output:

Opel

Table 5.3: Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string

Contd...

Notes

pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

Example: Create an array, assign values to it, and write the values to the output.

```
<html>
<body>
<script type="text/javascript">
var mycars = new Array();
mycars[0] = "mercedes";
mycars[1] = "ferari";
mycars[2] = "BMW";
mycars[3] = "opel";
mycars[4] = "honda city";
for (i=0;i<mycars.length;i++)
{
document.write(mycars[i] + "<br/>");
}
</script>
</body>
</html>
```

Output

```
mercedes
ferari
BMW
opel
honda city
```

Self Assessment

Fill in the blanks:

10. To modify a value in an existing array, just add a new value to the array with a specified ……..
11. An array can hold all your variable values under a …….. name.

5.5 Operators

Notes

Assignment Operator

The most common assignment operator is the equal sign. It sets one item equal to another.

Operator	What does it do?	Example/Explanation
=	Sets one value equal to another	counter=0 Sets the counter to equal the number 0
+=	Shortcut for adding to the current value.	clicks += 2 Sets the variable named counter to equal the current value plus two.
-=	Shortcut for subracting from the current value.	clicks -= 2 Sets the variable named counter to equal the current value minus two.
*=	Shortcut for multiplying the current value.	clicks *= 2 Sets the variable named counter to equal the current value multiplied by two.
/=	Shortcut for dividing the current value.	clicks /= 2 Sets the variable named counter to equal the current value divided by two.

Comparison Operator

The comparison operators compare two items and return a value of "true" if the conditions are true.

Operator	What does it do?	Example/Explanation
==	Returns a true value if the items are the same	counter==10 > Returns the value "true" if the counter's value is currently equal to the number 10
	Returns a true value if the items arenot the same	counter!=10 Returns the value "true" if the counter's value is any value except the number 10
>	Returns a true value if the item on the left is greater than the item on the right	counter>10 Returns the value "true" if the counter's value is larger than the number 10
>=	Returns a true value if the item on the left is equal to or greater than the item on the right	counter>=10 Returns the value "true" if the counter's value is equal to or larger than the number 10
<	Returns a true value if the item on the left is less than the item on the right	counter<10 Returns the value "true" if the counter's value is smaller than the number 10
<=	Returns a true value if the item on the left is equal to or less than the item on the right	counter<=10 Returns the value "true" if the counter's value is equal to or less than the number 10

Notes

Logical Operator

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

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

Computational Operator

The computational operators perform a mathematical function on a value or values, and return a single value.

Operator	What does it do?	Example/Explanation
+	Adds two values together	counter+2 Returns the sum of the counter plus 2
-	Subtracts one value from another	counter-2 Returns the sum of the counter minus 2
*	Multiplies two values	counter*10 Returns the result of the variable times 10
/	Divides the value on the left by the one on the right and returns the result	counter/2 Divides the current value of the counter by 2 and returns the result
++X	Increments the value, and then returns the result	++counter Looks at the current value of the counter, increments it by one, and then returns the result. If the counter has a value of 3, this equation returns the value of 4.
X++	Returns the value, and then increments the value	counter++ Returns the value of the counter, then increments the counter. If the counter has a value of 3, this equation returns the value of 3, then sets the counter value to 4.
--X	Decreases the value, and then returns the result	--counter Looks at the current value of the counter, decreases it by one, and then returns the result. If the counter has a value of 7, this equation returns the value of 6.
X--	Returns the value, and then decreases the value	counter-- Returns the value of the counter, then decrease the counter value. If the counter has a value of 7, this equation returns the value of 7, then sets the counter value to 6.

Notes

5.6 JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kinds of loops:

- for - loops through a block of code a specified number of times
- while - loops through a block of code while a specified condition is true

5.6.1 The for Loop

The for loop is used when you know in advance how many times the script should run.

Some important points to note are:

- The initialization statements are executed once; only when the for loop is encountered.
- After execution of initialization statements, the condition is evaluated.
- After every iteration, the updation statements are executed and then the condition is checked.

Syntax:

```
for (var=startvalue;var<=endvalue;var=var+increment)
      {
code to be executed
      }
```

Example: Loop example.

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Notes The increment parameter could also be negative, and the <= could be any comparing statement.

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is" + i);
document.write("<br />");
}
</script>
```

Contd...

Notes

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

Output

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

Explanation

This for loop starts with i=0.

As long as i is less than, or equal to 5, the loop will continue to run.

i will increase by 1 each time the loop runs.

5.6.2 The while Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (var<=endvalue)
        {
 code to be executed
 }
```

Note: The <= could be any comparing statement.

Example : while loop example.

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. we will increase by 1 each time the loop runs

Example:

```
<html>
        <body>
        <script  type="text/javascript">
                var  i=0;
        while  (i<=5)
                 {
          document.write("The  number  is"  +  i);
          document.write("<br  />");
                  i++;
                  }
```

Notes

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

Output

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

Explanation

i is equal to 0.

While i is less than, or equal to, 5, the loop will continue to run.

i will increase by 1 each time the loop runs.

5.6.3 The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

```
Syntax
do
  {
  code to be executed
  }
while (var<=endvalue);
```

Example: do while example.

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

```
<html>
        <body>
        <script type="text/javascript">
        var i=0;
        do
         {
        document.write("The number is" + i);
                document.write("<br />");
          i++;
          }
```

Notes

```
                while  (i<=5);
                       </script>
                </body>
        </html>
```

Output

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

Explanation:

i equal to 0.

The loop will run

i will increase by 1 each time the loop runs.

While i is less than, or equal to, 5, the loop will continue to run.

5.6.4 Break

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example: Break example.

```
<html>
<body>
<script  type="text/javascript">
var  i=0;
for  (i=0;i<=10;i++)
 {
 if  (i==3)
 {
break;
 }
document.write("The  number  is"  +  i);
document.write("<br  />");
}
</script>
</body>
</html>
```

Notes

Output

The number is 0

The number is 1

The number is 2

Explanation: The loop will break when i=3.

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

Output

The number is 0

The number is 1

The number is 2

Explanation: The loop will break when i=3.

5.6.5 Continue

The continue statement will break the current loop and continue with the next value.

Example: It is showing an example of continue.

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
continue;
   }
   document.write("The number is" + i);
   document.write("<br />");
  }
</script>
</body>
</html>
```

Output

The number is 0

The number is 1

Notes

The number is 2

The number is 4

The number is 5

The number is 6

The number is 7

The number is 8

The number is 9

The number is 10

Explanation: The loop will break the current loop and continue with the next value when i=3.

Self Assessment

Fill in the blanks:

12. The initialization statements are executed once; only when the …… is encountered.
13. The do...while loop is a variant of the ……

5.7 JavaScript Objects

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.

Notes An object is just a special kind of data. An object has properties and methods.

Properties

Properties are the values associated with an object.

In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.length);
</script>
```

Output

12

Methods

Methods are the actions that can be performed on objects.

Notes

In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

Output

HELLO WORLD!

5.7.1 String Object

The String object is used to manipulate a stored piece of text.

Examples of Use

The following example uses the length property of the String object to find the length of a string:

var txt="Hello world!"; document.write(txt.length);

Output

12

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

var txt="Hello world!"; document.write(txt.toUpperCase());

The code above will result in the following output:

HELLO WORLD!

Example: Illustrate how to style strings.

```
<html>
<body>
<script type="text/javascript">
var txt="Hello Upendra!";
document.write("<p>Big:" + txt.big() + "</p>");
document.write("<p>Small:" + txt.small() + "</p>");
document.write("<p>Bold:" + txt.bold() + "</p>");
document.write("<p>Italic:" + txt.italics() + "</p>":);
document.write("<p>Blink:" + txt.blink() + "(does not work in IE,
Chrome, or Safari)</       p>");
document.write("<p>Fixed:" + txt.fixed() + "</p>");
document.write("<p>Strike:" + txt.strike() + "</p>");
document.write("<p>Fontcolor:" + txt.fontcolor("Red") + "</p>");
document.write("<p>Fontsize:" + txt.fontsize(16) + "</p>");
document.write("<p>Subscript:" + txt.sub() + "</p>");
document.write("<p>Superscript:" + txt.sup() + "</p>");
```

Notes

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

Output

Big: Hello Upendra!

Small: Hello Upendra!

Bold: Hello Upendra!

Italic: Hello Upendra!

Blink: Hello Upendra! (does not work in IE, Chrome, or Safari)

Fixed: Hello Upendra!

Strike: ~~Hello Upendra!~~

Fontcolor: Hello Upendra!

Fontsize: Hello Upendra!

Subscript: Hello Upendra!

Superscript: Hello Upendra!

Example: How to use the replace() method to replace some characters with some other characters in a string.

```
<html>
<body>
<script type="text/javascript">
var str="Visit America!";
document.write(str.replace("America","India"));
</script>
</body>
</html>
```

Output

Visit India!

Example: How to use the length property to find the length of a string.

```
<html>
<body>
<script type="text/javascript">
var txt = "welcome to Saraswati Institute of Engineering &
Technology!";
document.write(txt.length);
</script>
</body>
</html>
```

Output

59

Notes

5.7.2 Date Object

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

Date objects are created with the Date () constructor.

There are four ways of instantiating a date:

new Date() // current date and time

new Date(milliseconds) //milliseconds since 1970/01/01

new Date(dateString)

new Date(year, month, day, hours, minutes, seconds, milliseconds)

Most parameters above are optional. Not specifying causes 0 to be passed in.

Once a Date object is created, a number of methods allow you to operate on it. Most methods allow you to get and set the year, month, day, hour, minute, second, and milliseconds of the object, using either local time or UTC (universal, or GMT) time.

All dates are calculated in milliseconds from 01 January 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds.

Example: Some examples of instantiating a date:

```
today = new Date()
d1 = new Date("October 13, 1975 11:13:00")
d2 = new Date(79,5,24)
d3 = new Date(79,5,24,11,33,0)
```

Set Dates

We can easily manipulate the date by using the methods available for the Date object.

In the example below we set a Date object to a specific date (14th January 2010):

var myDate=new Date();

myDate.setFullYear(2010,0,14);

And in the following example we set a Date object to be 5 days into the future:

var myDate=new Date();

myDate.setDate(myDate.getDate()+5);

Notes If adding five days to a date shifts the month or year, the Date object itself handles the changes automatically!

Compare Two Dates

The Date object is also used to compare two dates.

The following example compares today's date with the 14th January 2010:

Notes

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();
if (myDate>today)
{
alert("Today is before 14th January 2010");
}
else
{
alert("Today is after 14th January 2010");
}
```

Table 5.4: Date Object Methods

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
getTimezoneOffset()	Returns the time difference between GMT and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time (four digits)
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the getFullYear() method instead
parse()	Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
setDate()	Sets the day of the month (from 1-31)
setFullYear()	Sets the year (four digits)
setHours()	Sets the hour (from 0-23)

Contd...

Notes

setMilliseconds()	Sets the milliseconds (from 0-999)
setMinutes()	Set the minutes (from 0-59)
setMonth()	Sets the month (from 0-11)
setSeconds()	Sets the seconds (from 0-59)
setTime()	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month, according to universal time (from 1-31)
setUTCFullYear()	Sets the year, according to universal time (four digits)
setUTCHours()	Sets the hour, according to universal time (from 0-23)
setUTCMilliseconds()	Sets the milliseconds, according to universal time (from 0-999)
setUTCMinutes()	Set the minutes, according to universal time (from 0-59)
setUTCMonth()	Sets the month, according to universal time (from 0-11)
setUTCSeconds()	Set the seconds, according to universal time (from 0-59)
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the toUTCString() method instead
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date string since midnight of January 1, 1970, according to universal time
valueOf()	Returns the primitive value of a Date object

Example: How to display a clock on your web page.

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
```

Notes

```
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
}
function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>
<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

Output

15:50:23

5.7.3 Boolean Object

The Boolean object represents two values: "true" or "false".

The following code creates a Boolean object called myBoolean:

```
var myBoolean=new Boolean();
```

Note: If the Boolean object has no initial value or if it is 0, -0, null, "", false, undefined, or NaN, the object is set to false. Otherwise it is true (even with the string "false")!

All the following lines of code create Boolean objects with an initial value of false:

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

Notes

And all the following lines of code create Boolean objects with an initial value of true:

var myBoolean=new Boolean(true);

var myBoolean=new Boolean("true");

var myBoolean=new Boolean("false");

var myBoolean=new Boolean("Richard");

Table 5.5: Boolean Object Methods

Method	Description
toString()	Converts a Boolean value to a string, and returns the result
valueOf()	Returns the primitive value of a Boolean object

Example: Check if a Boolean object is true or false.

```
<html>
<body>
<script type="text/javascript">
var b1=new Boolean(0);
var b2=new Boolean(1);
var b3=new Boolean("");
var b4=new Boolean(null);
var b5=new Boolean(NaN);
var b6=new Boolean("false");
document.write("0 is boolean "+ b1 +"<br />");
document.write("1 is boolean "+ b2 +"<br/>");
document.write("An empty string is boolean "+ b3 + "<br />");
document.write("null is boolean "+ b4+ "<br />");
document.write("NaN is boolean "+ b5 +"<br />");
document.write("The string 'false' is boolean "+ b6 +"<br />");
</script>
</body>
</html>
```

Output

0 is boolean false

1 is boolean true

An empty string is boolean false

null is boolean false

NaN is boolean false

The string 'false' is boolean true

Notes

5.7.4 Math Object

The Math object allows you to perform mathematical tasks.

The Math object includes several mathematical constants and methods.

Syntax for using properties/methods of Math:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Note: Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

- Mathematical Constants

  JavaScript provides eight mathematical constants that can be accessed from the Math object. These are: E, PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, and base-10 log of E.

  You may reference these constants from your JavaScript like this:

  Math.E

  Math.PI

  Math.SQRT2

  Math.SQRT1_2

  Math.LN2

  Math.LN10

  Math.LOG2E

  Math.LOG10E

- Mathematical Methods

  In addition to the mathematical constants that can be accessed from the Math object there are also several methods available.

  The following example uses the round() method of the Math object to round a number to the nearest integer:

  document.write(Math.round(4.7));

  The code above will result in the following output:

  5

  The following example uses the random() method of the Math object to return a random number between 0 and 1:

- Document.write(Math.random());

  The code above can result in the following output:

  011917396456824547

  The following example uses the floor() and random() methods of the Math object to return a random number between 0 and 10:

  document.write(Math.floor(Math.random()*11));

  The code above can result in the following output:

  7

Notes

Table 5.6: Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of Ex
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

5.7.5 JavaScript Functions

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function?

```
Syntax
        function  functionname(var1,var2,...,varX)
        {
        some  code
        }
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Notes

Notes A function with no parameters must include the parentheses () after the function name.

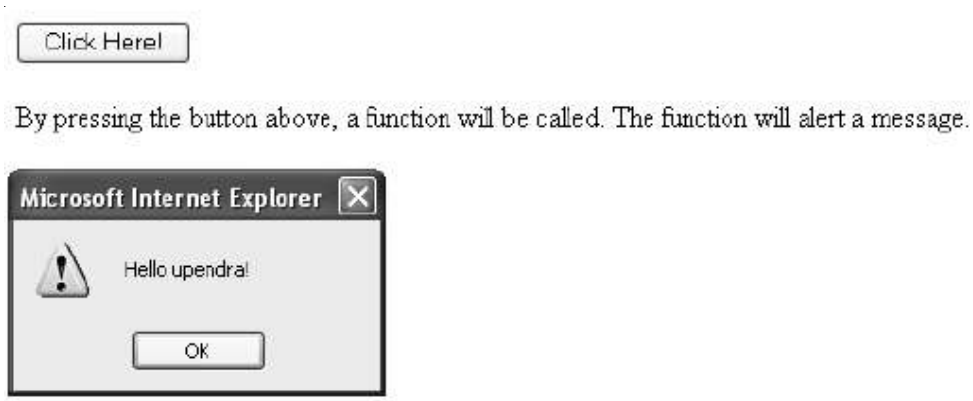
Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

Example:

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello upendra!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()"/>
</form>
<p>By pressing the button above, a function will be called. The
function will alert a message.</ p>
</body>
</html>
```

Output

Figure 5.1



Notes

If the line: alert("Hello upendra!!") in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.

The return Statement

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement.

Example:

```
<html>
<head>
<script type="text/javascript">
function product(x,y)
{
return x*y;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(9,6));
</script>
```

5.8 Dialog Boxes

Three Types of Dialog Boxes in JavaScript

- alert()
- confirm()
- prompt()

alert()

The simplest to direct output to a dialog box is to use the alert() method.

```
alert("Click Ok to continue.");
```

Notes Note that the alert() method doesn't have an object name in front of it. This is because the alert() method is part of the window Object. As the top-level object in the Navigator Object Hierarchy, the window Object is assumed when it isn't specified.

The script alert("Click Ok to continue."); and HTML holding the script will not continue or execute until the user clicks the OK button.

Notes

Generally, the alert() method is used for exactly that – to warn the user or alert him or her to something. Examples of this type of use include:

- Incorrect information in a form
- An invalid result from a calculation
- A warning that a service is not available on a given date

Nonetheless, the alert() method can still be used for friendlier messages.

> *Notes* Note that Netscape alert boxes include the phrase "<url> [JavaScript Application]" in the title bar of the alert while IE has "Microsoft Internet Explorerin the title bar of the alert. Both have yellow triangles to "alert" you. This done in order to distinguish them form those generated by the operating system or the browser. This done for security reasons so that malicious programs cannot trick users into doing things they don't want to do.

```
<INPUT TYPE="button" VALUE="alert" onClick="alert('This is an alert!!')">
```

OR

```
<INPUT TYPE="button" VALUE="alert" onClick="window.alert('This is an alert!!')">
```

The alert dialog box is used to display an alert message to the user.

prompt()

The alert() method still doesn't enable you to interact with the user. The addition of the OK button provides you with some control over the timing of events, but it still cannot be used to generate any dynamic output or customize output based on user input.

The simplest way to interact with the user is with the prompt() method. The user needs to fill in the field and then click OK.

prompt("Enter Your Name:", "Name");

> *Notes*
>
> - You are providing two "arguments" to the method in the parenthesis. The prompt() method "requires two pieces of information". The first is text to be displayed, and the second is the default data in the entry field.
> - In JavaScript, when a method requires more than one argument, they are separated by commas.

```
<INPUT TYPE="button" VALUE="prompt" onClick="respPrompt()">
```

The prompt() method dialog box allows the user the opportunity to enter information. It takes two parameters; a message and a default string for the text entry field.

With function code:

```
<SCRIPT LANGUAGE="JavaScript">
function respPrompt() {
```

Notes

```
      var favorite = prompt('What is your favorite color?', 'RED');
      // OR var favorite = window.prompt('What is your favorite color?', 'RED');
      // if (favorite) equivalent to if (favorite != null && favorite != "");
      if (favorite) alert("Your favorite color is: " + favorite);
      else alert("You pressed Cancel or no value was entered!");
}
</SCRIPT>
```

Notes What happens when you press Cancel? The value null is returned.

confirm()

Confirm displays a dialog box with two buttons: OK and Cancel. If the user clicks on OK the window method confirm() will return true. If the user clicks on the Cancel button confirm() returns false.

```
<INPUT TYPE="button" VALUE="confirm" onClick="respConfirm()">
```

The confirm dialog box returns a Boolean value based on the user's selection.

With function code:

```
<SCRIPT LANGUAGE="JavaScript">
function respConfirm () {
       var response = confirm('Confirm Test: Continue?');
       // OR var response = window.confirm('Confirm Test: Continue?');
       if (response) alert("Your response was OK!");
       else alert("Your response was Cancel!");
}
</SCRIPT>
```

Notes Response will be:

1. true if OK is pressed
2. false if Cancel is pressed

5.9 String Manipulation Functions

The most important string manipulation functions are the following:

- length
- toLowerCase
- toUpperCase
- charAt
- indexOf

Notes

- split
- substr
- subString

Now let's see one by one what can we do with them and how to use them in a JavaScript code.

length

```
Syntax: object.length;
```

This function returns with the total number of characters in the string, or more simple it returns with the length of the string. The usage is quite simple:

Code:

```
    var str = 'Demo text.';
var len = str.length;
var len2 = 'my other text'.length;
toLowerCase and toUpperCase
Synatx: object.toLowerCase();
```

These methods can be very useful if you want to all character in a string lower case or upper case. For example if you want to compare 2 strings but you are not sure if they are capitalized or not. In this case just convert both of them to lower or upper case and you can make the comparison without any problem.

Example: Usage example:

Code:

```
var str = 'Demo Text.';
var strL = str.toLowerCase();
var strU = str.toUpperCase();
if (str.toLowerCase() == 'demo text.')
```

charAt

Synatx: object.charAt(index);

With this function you can get the character at the given position in a string. If you want you can read the complete string character by character using charAt and a loop.

Example: See the examples:

Code:

```
var str = 'Demo Text.';
var c    = str.charAt(5);
for (i=0;i<str.length;i++){
alert(str.charAt(i));
   }
```

5.10 Using Timer in Web Page

Notes

```
<xmp>
<!—CLOCK—>
<!— This part can go in the head of the html file —>
<script language=”JavaScript” type=”text/javascript”>
function sivamtime() {
   now=new Date();
   hour=now.getHours();
   min=now.getMinutes();
   sec=now.getSeconds();
if (min<=9) { min=”0"+min; }
if (sec<=9) { sec=”0"+sec; }
if (hour>12) { hour=hour-12; add=”pm”; }
else { hour=hour; add=”am”; }
if (hour==12) { add=”pm”; }
document.timeForm.field.value = ((hour<=9) ? “0”+hour : hour) + “:” + min
+ “:” + sec + “ “ + add;
setTimeout(“sivamtime()”, 1000);
}
window.onload = sivamtime;
// —>
</script>
<!— This goes into the body of the file wherever you want to have the time
placed —>
<body>
<form name=”timeForm”>
  <input type=text” name=”field” value=”” size=”11">
</form>
</body>
</xmp>
```

Output

Top of Form

12.18.57

Bottom of Form

Notes

Self Assessment

Fill in the blanks:

14. An OOP language allows you to define your own objects and make your own ……

15. A function contains code that will be executed by an …… or by a call to the function.

16. The …… method still doesn't enable you to interact with the user.

Caselet

Intel Aims to Lower PC Power Consumption by 300 Times in 10 Years

The world's largest chip- maker Intel Corporation on Thursday said it is working on a host of futuristic technologies that would improve the power efficiency of PCs 300-fold in the next 10 years, as well as ensure the security of data and user identities.

Speaking on the final day of the Intel Developer Forum (IDF) 2011 here, Intel Chief Technology Officer Mr Justin Rattner said the company was developing technologies to take computing to the next level, with better performance and lower power consumption.

Energy efficiency was a key theme of the three-day IDF summit this year and a number of Intel executives demonstrated the efforts being taken by the company in this regard. The move assumes significance in light of consumers gravitating toward always-on computing devices with a greater degree of mobility.

Mr Rattner said that Intel's multi-core technology, in which more than one processing engine is built into a single chip, has become the accepted methodology for increasing performance while keeping power consumption low.

These technologies would enable faster web access, improve PC user security and reduce the requirement for wireless infrastructure to provide the optimal online experience, among other benefits, he said.

Mr Rattner demonstrated a new technology for better PC security, wherein users would be able to see images and other data on social networking sites and other platforms only if the computer recognises his or her face.

The technology will enable parallel cryptographic and facial recognition services to improve security on Ultrabooks and traditional notebooks, besides desktop PCs, with the help of Intel microprocessors, he said.

Intel was also collaborating with China Mobile to replace existing "costly base-station hardware used on cell towers today with a fully programmable and far more cost-effective, software-based PC alternative", he noted.

Mr Rattner revealed that Intel Labs was working on a new 'Near-Threshold Voltage Processor' that has enabled an experimental Pentium-class processor unit to deliver five times better energy efficiency levels, with the ability to run a processor with a solar cell the size of postage stamp.

Contd...

Notes

He said the extreme-scale computing technologies that Intel was working on would help achieve the goal of a nearly 300 times improvement in energy efficiency levels in the next ten years and potentially even a 1,000-fold improvement in the future.

Mr Rattner also disclosed a new JavaScript solution that could speed up browser-based content such as 3D games by up to eight-fold and said that Intel would also soon launch the world's first processor with Many-Integrated Core (MIC) architecture, which promises to revolutionise high-performance computing.

5.11 Summary

- JavaScript is a browser-interpreted language.
- JavaScript can react to events.
- Conditional Statements give the JavaScript code you are writing the ability to make decisions or perform single or multiple tasks.
- A variable's purpose is to store information so that it can be used later.
- JavaScript can be used to validate data in HTML forms before sending off the content to a server.
- An OOP language allows you to define your own objects and make your own variable types.

5.12 Keywords

ASP: Active Server Pages

CGI: Common Gateway Interface

OOP: Object Oriented Programming

WWW: World Wide Web

5.13 Review Questions

1. Substantiate the mean of java script? What are the advantages and disadvantages of using java script?
2. Calculate the factorial of a number using java script.
3. How can you change an image using java script?
4. Make the distinction between java and java script. Give example to support your answers.
5. Write a java script code to display the current date and time.
6. Is java script a scripting or a programming language? Discuss
7. What do you mean by java script top level properties and function?
8. Each element in the array has its own ID so that it can be easily accessed. Write a program to explain this statement with explanation.
9. How is JavaScript syntax like C / C++?Write a program to explain
10. The Math object allows you to perform mathematical tasks. Analyze

Notes

Answers: Self Assessment

1.	HTML	2.	server
3.	JavaScript	4.	Data types
5.	double or single	6.	concatenation
7.	complex	8.	variable
9.	var	10.	Index number
11.	Single	12.	for loop
13.	while loop	14.	variable types
15.	event	16.	alert()

5.14 Further Readings

Books

Danny Goodman, Michael Morrison, Brendan Eich, *JavaScript Bible,* John Wiley & Sons

James Edward Keogh, *JavaScript demystified,* McGraw-Hill Professional

Powell, *Java Script: The Complete Reference,* Tata McGraw-Hill Education

Online links

http://www.w3schools.com/js/

http://javascriptsource.com/

Notes

Unit 6: DOM Model

CONTENTS

Objectives

Introduction

6.1 DOM Model

6.1.1 The JavaScript Assisted Style Sheets DOM (JSSS DOM)

6.2 Events Handling through JavaScript

6.2.1 Event Association

6.3 How to use Forms in JavaScript

6.3.1 Form Validation with JavaScript

6.4 Summary

6.5 Keywords

6.6 Review Questions

6.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Recognize the DOM model
- Describe event handling through JavaScript
- Demonstrate how to use forms in JavaScript

Introduction

An HTML page is rendered (painted) in a browser. The browser assembles all the elements (objects) contained in the HTML page, downloaded from the web server, in its memory. Once, done the browser then renders these objects in the browser window. Once, the HTML page is rendered in the browser window, the browser can no longer recognize individual HTML elements (objects).

To create an interactive web page, it is imperative that the browser continues to recognize individual HTML objects even after they are rendered in the browser window. This allows the browser to access the properties of these objects using the built-in methods of the object. Once the properties of an object are accessible then the functionality of the object can be controlled at will.

6.1 DOM Model

JavaScript enabled browsers are capable of recognizing individual objects in an HTML page, after the page has been rendered in the browser, because the JavaScript enabled browser recognizes and uses the Document Object Model (DOM).

Notes

Using the Document Object Model (DOM) JavaScript enabled browsers identify the collection of web page objects (web page elements) that have to be dealt with while rendering an HTML based, web page in the browser window.

Caution The HTML objects which belong to the DOM, have a descending relationship with each other.

The top most object in the DOM is the 'Navigator' (i.e., the browser) itself. The next level in the DOM the browser's 'Window'. The next level in the DOM is the 'Document' displayed in the browser's window.

Should the document displayed in the browser's window have an HTML 'Form' coded in it, the next level in the DOM is the 'Form' itself.

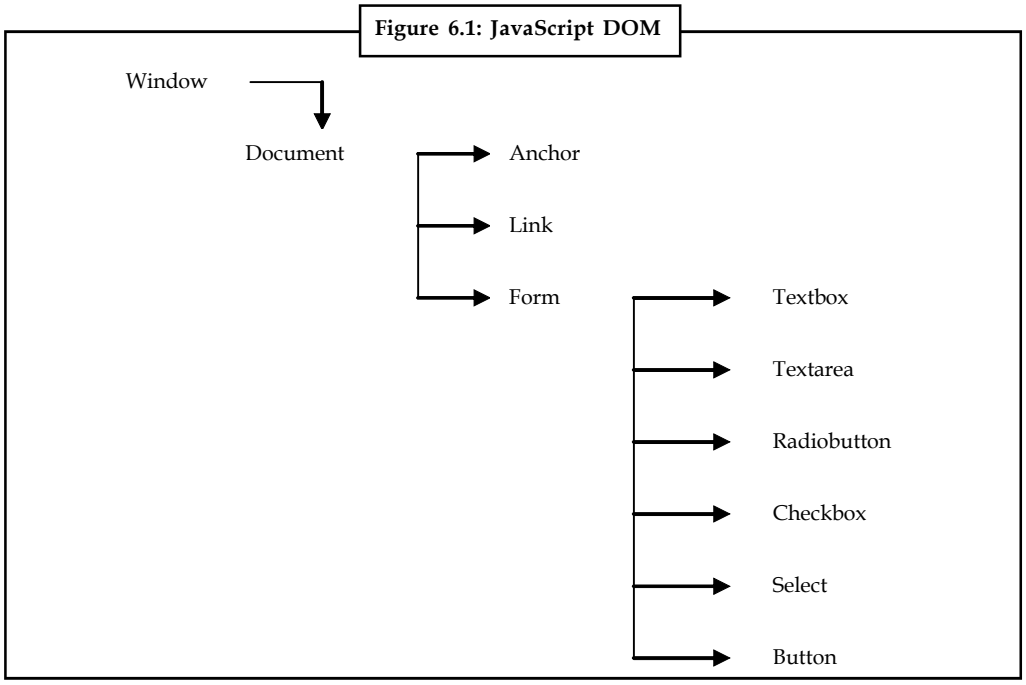
The DOM hierarchy continues downward to encompass individual elements on a 'FORM', such as Text boxes, Labels, Radio buttons, Check boxes, Push buttons and so on, which belong to the form.

JavaScript's object hierarchy is mapped to the DOM, which in turn is mapped to the web page elements in the browser window. Hence, when a web page is rendered in a JavaScript enabled browser window, JavaScript is capable of uniquely identifying each element in the web page, because major elements of a web page are bound to the DOM.

The DOM that JavaScript recognizes is described below:

The Navigator – i.e., Netscape Navigator, Internet Explorer, Opera, Mosaic, etc.

Figure 6.1: JavaScript DOM



Instance

No HTML is registered in the DOM by a JavaScript enabled browser unless they are assembled in memory prior being rendered in the browser window. What this means is, if a document does not have any 'Anchor' described in the 'Anchors' object will exist but it will be empty. If the document does not have any 'Link' described in it the Links object will exist but it will be empty.

Notes

Hierarchy

All objects on a web page are not created equal. Each exists in a set relationship with other objects on the web page. From Figure (above figure), the navigator occupies the topmost slot in the DOM followed by the Window object and so on.

Below the Window is the 'Document' object. Below the document object three other objects exist. They are the 'Anchor', 'Link' and 'Form' objects. Individual form elements are found under the 'Form' object.

In addition to the DOM, other objects currently recognized by a JavaScript enabled browser are Plug-ins, Applets and Images. Hence using a JavaScript enabled browser and JavaScript most of major web page objects are accessible.

However, every single element of a web page rendered in the browser window, is not part of DOM.

Did u know? Is HTML tags part of the DOM?

HTML tags such as <HEAD></HEAD> or <BODY></BODY> are not part of the DOM. Presentation styles, headings, body text, H1 to H6 and so on are not part of the DOM hence and not recognized by JavaScript.

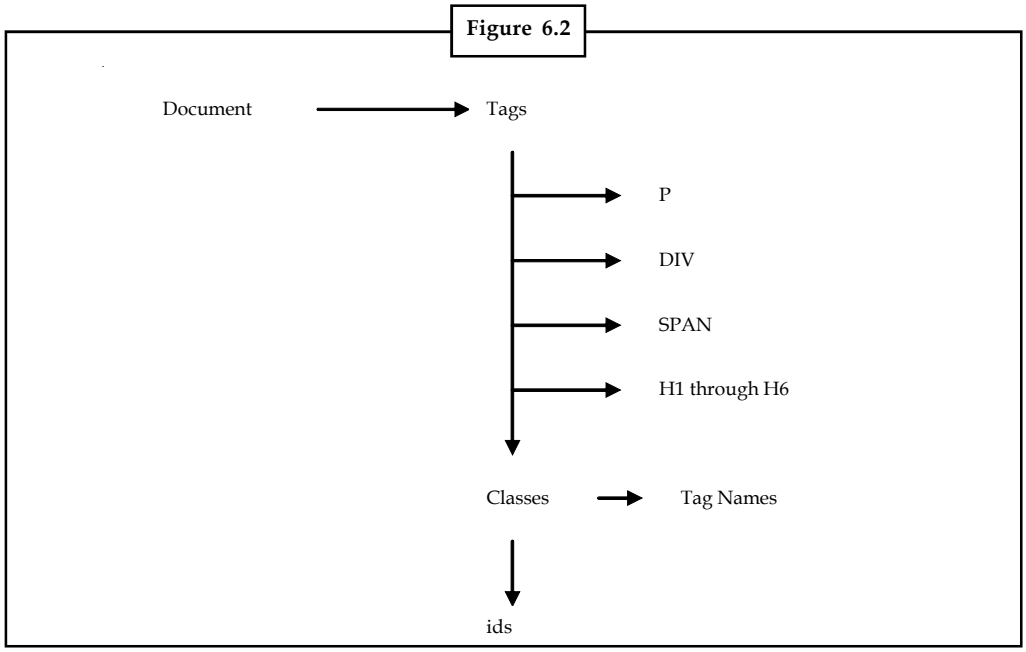
JavaScript, however, recognizes presentation styles, headings, body text, H1 to H6 and so on, when JavaScript assisted Style Sheets (JSSS) are in a web page. JSSS is usually between the <HEAD></HEAD> HTML tags in a web page.

6.1.1 The JavaScript Assisted Style Sheets DOM (JSSS DOM)

JSSS use JavaScript syntax to control a document's presentation style. When a JSSS is embedded in an HTML page within the <HEAD></HEAD> tags, then the JavaScript DOM picks up a whole new set of objects, which add to the standard DOM objects already recognized by JavaScript.

The additional objects brought into the DOM by JSSS are shown in the figure below.

Figure 6.2



Notes

By extending the DOM recognized by JavaScript by embedding JSSS in a web page, developers of web pages can access every element of a web page whether this element appears on the page when it is rendered in a client browser or not.

By accessing appropriate properties of the 'Navigator' object, (i.e., the Browser), the topmost object in the DOM, JavaScript can recognize the browser type (i.e., Netscape Navigator, Internet Explorer, etc.) and subsequently dispatch all HTML pages to the browser from the web server, with a style based on this knowledge. This is where the power of JavaScript really becomes visible in providing finely tuned web page content to a client's browser.

Since JavaScript understands the DOM and can extend the DOM with the use of JSSS in a web page JavaScript understands 'Objects'.

All objects have:

- Properties that determine the functionality of the object
- Methods that allow access to these properties
- Events to appropriate JavaScript event handlers.

Hence, when a pre-determined event occurs the code snippet will execute. This is the traditional Object, Event driven, Code execution model of any object based programming environment.

Using appropriate JavaScript snippets, which reference the properties of an object via its built-in methods, developers of web pages can actually control the functionality of any HTML, object in the DOM (or extended DOM) while the HTML program executes (i.e., at run time).

Notes JavaScript can access the methods of all objects belonging to the DOM and JSSS DOM. Hence using JavaScript, truly interactive web pages can be created.

JavaScript features are generated towards providing developers the capability to quickly generate scripts that will execute in the context of a web page within a JavaScript enabled browser or on a web server that understands JavaScript.

Although JavaScript does not provide all of the features of a full object oriented programming language, it does provide a suite of object-based features that are especially tailored to Browser or Server side scripting.

These features include the recognition of a number of predefined browser and server objects. JavaScript has the ability to control the behavior of these objects through their properties and methods.

Self Assessment

Fill in the blanks:

1. Using the Document Object Model (DOM) JavaScript enabled browsers identify the collection of …… objects.
2. The DOM hierarchy continues downward to encompass individual elements on a '……'.
3. JavaScript's object hierarchy is mapped to the DOM, which in turn is mapped to the web page elements in the …… window.

Notes

4. …… does not provide all of the features of a full object oriented programming language.
5. JSSS use JavaScript syntax to control a …… presentation style.
6. All …… on a web page are not created equal.
7. JavaScript enabled browser and JavaScript most of major web page objects are ……

6.2 Events Handling through JavaScript

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags. Examples of events are:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke.

Notes Events are normally used in combination with functions, and the function will not be executed before the event occurs!

- onLoad and onUnload

  The onLoad and onUnload events are triggered when the user enters or leaves the page.

  The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

  Both the onLoad and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page.

  For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome upendra!".

- onFocus, onBlur and onChange

  The onFocus, onBlur and onChange events are often used in combination with validation of form

  fields.

  Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

Notes

- onSubmit

  The onSubmit event is used to validate ALL form fields before submitting it.

  Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.html" onsubmit="return checkForm()">
```

- onMouseOver and onMouseOut

  onMouseOver and onMouseOut are often used to create "animated" buttons.

  Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="a.html" onmouseover="alert('An onMouseOver event');return false</a>
```

Task Create a web page, which has an image file. Use the onMouseDown event handler, which will resize the image when the user presses a mouse button over the image.

6.2.1 Event Association

Events are associated with HTML tags. The definitions of the events described below are as follows:

- abort - A user action caused an abort of an image or document load.
- blur - A frame set, document, or form object such as a text field loses the focus for input.
- click - Happens when a link or image map is clicked on.
- change - Happens when a form field is changed by the user and it loses the focus.
- error - An error happened loading a image or document.
- focus - A frame set, document, or form object such as a text field gets the focus for input.
- load - The event happens when an image or HTML page has completed the load process in the browser.
- mouseOut - The event happens when the mouse is moved from on top of a link or image map.
- mouseOver - The event happens when the mouse is placed on a link or image map.
- reset - The user reset the object which is usually a form.
- submit - The user submitted an object which is usually a form.
- unload - The object such as a framesed or HTML document was exited by the user.

Task Create a web page, which has a text hyperlink and an image file. Use the onClick event handler to display an alert when the link is clicked and the onDbClick event handler to display an alert when you double click on the image.

Notes

The events for each HTML tag are as follows:

- <A>
  - click (onClick)
  - mouseOver (onMouseOver)
  - mouseOut (onMouseOut)
- <AREA>
  - mouseOver (onMouseOver)
  - mouseOut (onMouseOut)
- <BODY>
  - blur (onBlur)
  - error (onError)
  - focus (onFocus)
  - load (onLoad)
  - unload (onUnload)
- <FORM>
  - submit (onSubmit)
  - reset (onReset
- <FRAME>
  - blur (onBlur)
  - focus (onFocus)
- <FRAMESET>
  - blur (onBlur)
  - error (onError)
  - focus (onFocus)
  - load (onLoad)
  - unload (onUnload)
- <IMG>
  - abort (onAbort)
  - error (onError)
  - load (onLoad)
- <INPUT TYPE = "button">
  - click (onClick)
- <INPUT TYPE = "checkbox">
  - click (onClick)

Notes

- <INPUT TYPE = "reset">
  - click (onClick)
- <INPUT TYPE = "submit">
  - click (onClick)
- <INPUT TYPE = "text">
  - blur (onBlur)
  - focus (onFocus)
  - change (onChange)
  - select (onSelect)
- <SELECT>
  - blur (onBlur)
  - focus (onFocus)
  - change (onChange)
- <TEXTAREA>
  - blur (onBlur)
  - focus (onFocus)
  - change (onChange)
  - select (onSelect)

Example:

```
<html>
<head>
<script type="text/javascript">
function popup()
        {
        alert("Hello upendra!")
        }
</script>
</head>
<body>
<input type="button" value="Click Me!" onclick="popup()"><br />
<a href="#" onmouseover="" onMouseout="popup()">
Hover Me!</a>
</body>
</html>
```

Output

Notes

Figure 6.3

Task Create a web page, which takes input from the user, and send an alert message to the user if he has failed to fill in the required fields. Use the onSubmit event handler is to validate the form input. In this way you can verify whether users have filled in required fields, made the required selection, or whether they have filled in an appropriate combination of fields.

Self Assessment

Fill in the blanks:

8. Every element on a web page has certain events which can trigger a …….
9. The ……. function will be called when the user clicks the submit button in the form.
10. The ……. event is used to validate ALL form fields before submitting it.
11. onMouseOver and ……. are often used to create "animated" buttons.

6.3 How to use Forms in JavaScript

6.3.1 Form Validation with JavaScript

JavaScript can be used to validate data in HTML forms before sending off the content to a server.

Form data that typically are checked by a JavaScript could be:

- Has the user left required fields empty?
- Has the user entered a valid e-mail address?
- Has the user entered a valid date?
- Has the user entered text in a numeric field?

Form validation is the process of checking that a form has been filled in correctly before it is processed.

Example: If your form has a box for the user to type their e-mail address, you might want your form handler to check that they've filled in their address before you deal with the rest of the form.

Notes

There are two main methods for validating forms: server-side (using CGI scripts, ASP, etc.), and client-side (usually done using JavaScript). Server-side validation is more secure but often more tricky to code, whereas client-side (JavaScript) validation is easier to do and quicker too (the browser doesn't have to connect to the server to validate the form, so the user finds out instantly if they've missed out that required field!).

Figure 6.4: Client-side form Validation (usually with JavaScript Embedded in the Web Page)

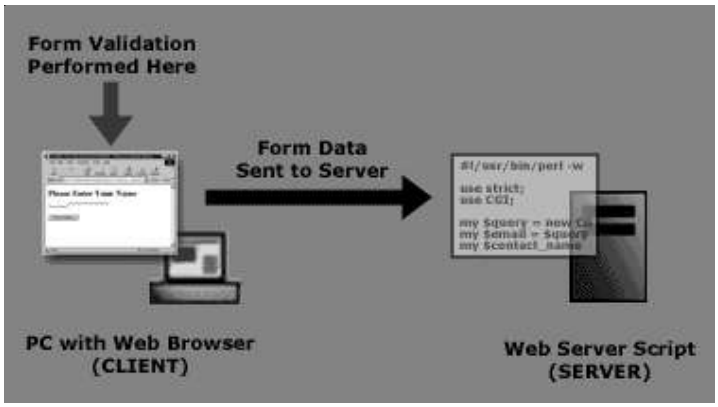
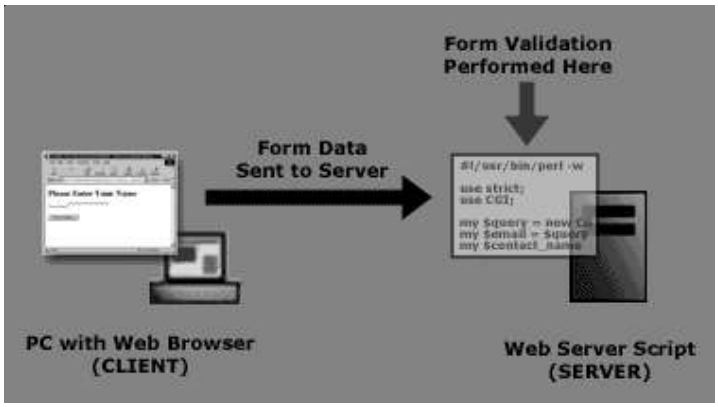


Figure 6.5: Server-side form Validation (usually Performed by a CGI or ASP Script)



Required Fields

The function below checks if a required field has been left empty. If the required field is blank, an alert box alerts a message and the function returns false. If a value is entered, the function returns true (means that data is OK):

```
function  validate_required(field,alerttxt)
{
with  (field)
  {
   if  (value==null||value==“”)
     {
```

Notes

```
      alert(alerttxt);return false;
    }
  else
    {
     return true;
    }
  }
}
```

In the below example the entire script, with the HTML form could look something like this:

Example: Illustrating form validation.

```
<html>
<head>
<script type="text/javascript">
function validate_required(field,alerttxt)
{
with (field)
  {
   if (value==null||value==“”)
    {
     alert(alerttxt);return false;
    }
  else
    {
     return true;
    }
  }
}
function validate_form(thisform)
{
with (thisform)
       {
   if (validate_required(email,”Email must be filled out!”)==false)
         {email.focus();return false;}
         }
       }
       </script>
       </head>
       <body>
```

Notes

```
<form action="submit.htm" onsubmit="return validate_form(this)" method="post">
  Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Output

After clicking submit button on empty e-mail the message is as shown in Figure 6.4.

Figure 6.6

E-mail Validation

The function below checks if the content has the general syntax of an e-mail.

This means that the input data must contain at least an @ sign and a dot (.). Also, the @ must not be the first character of the e-mail address, and the last dot must at least be one character after the @ sign:

```
function validate_email(field,alerttxt)
{
with (field)
  {
   apos=value.indexOf("@");
   dotpos=value.lastIndexOf(".");
   if (apos<1||dotpos-apos<2)
      {alert(alerttxt);return false;}
   else {return true;}
  }
}
```

In the below Example the entire script, with the HTML form could look something like this.

Example: Illustrating E-mail validation.

```
<html>
<head>
<script type="text/javascript">
function validate_email(field,alerttxt)
{
```

Notes

```
with (field)
  {
   apos=value.indexOf("@");
   dotpos=value.lastIndexOf(".");
   if (apos<1||dotpos-apos<2)
      {alert(alerttxt);return false;}
   else {return true;}
  }
}
function validate_form(thisform)
{
with (thisform)
  {
   if (validate_email(email,"Not a valid e-mail address!")==false)
      {email.focus();return false;}
  }
}
</script>
</head>
<body>
<form action="submit.htm" onsubmit="return validate_form(this);"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Output

Without giving the @ the error will be like the below output.

Figure 6.7

Notes

Self Assessment

Fill in the blanks:

12. JavaScript can be used to validate data in HTML forms before sending off the content to a ……...
13. If the required field is blank, an ……... alerts a message and the function returns false.
14. ……... validation is more secure but often more tricky to code.
15. ……... is the process of checking that a form has been filled in correctly before it is processed.

Caselet

IT is India's Tomorrow

THE Grameen phone scheme in Bangladesh provided one cellular phone each to 10,000 villages for community use. Kerala's fishermen bargain rates for their catch using mobiles they carry out to sea. An Internet bus goes about with 20 computers in Malaysia, "bringing a new world of information and learning opportunities to school children in rural communities". And Dr Devi Shetty, a cardio-surgeon in Bangalore, is connected to 27 districts in the state and also to West Bengal and Assam for consultation. Is there a correlation between Information and Communication Technology (ICT) usage and economic growth? Yes, says Vinod Vaish in his foreword to The Great Digital Transformation by D.K. Ghosh, from Sunrise Publications (sunrisepublications@front.ru). "The intensive application of ICTs has enabled emergence of a new company characterised by high productivity, efficient markets, innovations in products and services, technologies, business models and organisational structures." The book, subtitled `a saga of sustainable development', notes that the South Asian countries are a fertile group to cooperate on ICT implementation because of their geographic continuity, common historic experience, close cultural and linguistic environment and so forth.

Peter Drucker's book Management Challenges for the 21st Century is cited in a chapter that begins with a new definition of IT by the Prime Minister, Atalji - as 'India's tomorrow'. But Drucker had a different explanation: That for almost four decades people thought IT meant merely T - that is, data processing using a computer; the significance of I in IT came much later. The enquiry is "leading rapidly to redefining the tasks to be done with the help of information and, with it, to redefining the institutions that do these tasks."

The Malaysian Model, dealt with in a separate chapter, discusses the 'Multimedia Super Corridor' (MSC) - a forum for "new roles of government, new cyber laws and guarantees, collaborations between government and firms, companies and companies, new broadcasting, new types of entertainment, education and delivery of healthcare." Ghosh delves into something philosophical when laying down what are desirable as features in an international telecom order: "open, flexible, and competitive; user, rather than operator-oriented; containing an element of universal service both at the domestic and the international level; and economically efficient." But there is a telecommunications gap; it has three main dimensions. "The international gap, qualitative and technological gap, and the domestic gap."

Contd...

Notes

Towards the end of the book, the author writes: "India, Malaysia and the Philippines, the three South Asian countries benefiting from outsourcing phenomenon would themselves be outsourcing their work once they too grow to be of world class." What a pipedream, you may think. But he adds: "This is happening already; an Indian company has set up a BPO unit in Malaysia. And Indian IT companies are buying up many small US companies and turning them around." Is that making you sit up already?

Blessed are the Perl-iterates

BIOLOGY is a life science, while computing is a machine world. But computers have become commonplace in biology, writes D. Curtis Jamison in "Perl Programming for Bioinformatics & Biologists", from Wiley Dreamtech (www.wileydreamtech.com) . "Almost every biology lab has some type of computer, and the uses of the computer range from manuscript preparation to Internet access, from data collection to data crunching." The field of bioinformatics can be split into two broad areas, states the intro: "Computational biology and analytical bioinformatics." The former is about "formal algorithms and testable hypotheses of biology, encoded into various programs"; computationists "spend their time thinking about the mathematics of biology" and develop bioinformatic tools such as BLAST or FASTA. Analytical bioinformatics puts those tools to use for tasks such as sequencing or regression.

Why Perl? Because it is the most widely used scripting language in bioinformatics, notes the author. What is Perl? Its author Larry Wall christened it so for 'practical extraction and reporting language', because it was originally created "for parsing files and creating formatted reports". The name could just as easily stand for 'pathologically eclectic rubbish lister' Wall had jested because the language is "perfect for rummaging through files looking for a particular pattern or characters, or for reformatting data tables."

How do these scientists put the language to use? For quick and dirty creation of small analysis programs, such as "to parse a nucleotide sequence into the reverse complement sequence". Such a program is called `glutility' - because it takes the output of one program and changes it into a form suitable for import into another program.

The book is replete with bio examples, such as storing DNA segment into a string, and using Perl's power "to find motifs, translate DNA sequences to RNA, or transcribe RNA sequences to protein"; deploying Bioperl that ships with Tools distribution; applying splice to truncate an array, e.g. splice(@genes, 1). What a blessing to have Perl help in bio work! But `blessing' a referent is the actual trick to creating object-oriented Perl code, writes the author. "The bless command marks the referent as belonging to a particular class or package." Okay, how to bless? bless($reference, "package_name"). Count yourself blessed if you are Perl-iterate!

Coding is the 'easy' part

THE Mars expedition has Java running far, far away. For the earthlings, Paul Hatcher and John Gosney write JavaScript Professional Projects, a book from Easwar Press (www.eswarbooks.com) . "This book is not beginner-level basic tutorial, but a more advanced exploration of a real-world project that will show you how to implement JavaScript in actual applications," warns the intro. Center Park School is the fictitious project for which you play Web designer. "Rather than just throwing a bunch of sample code at you and asking you to make sense of it on your own, the project is divided into chapters that deal with a specific aspect of the final Web site." If you thought all design is about coding, you could be wrong. "Actual coding of a project is often the `easy' part, and developing a design plan and project template the real challenge," say the authors. "Working with clients can be a daunting task, especially if those customers are not technically minded."

Contd...

Notes

All right, what is JavaScript? Designed by Netscape Communications and Sun, it is a "lightweight programming language that you can use to add dynamic effects to your Web pages." HTML has limitations, because it can only describe the way a page's elements such as text, forms, hyperlinks and tables look like; it cannot dictate how they behave which is where JavaScript steps in. "The ability to embed JavaScript scripts in a Web page gives you, the programmer, much more control over how your Web page behaves." When you use it in combination with the browser's Document Object Model (DOM), JavaScript can produce intricate, dynamic HTML effects as well as animation and sound.

If your job is in IS security, you must remember that JavaScript has a history of security problems. Most of these security holes have been caught and fixed, "but new ones are being discovered all the time." So, a developer has to "keep up-to-date on the current status" of patches and bugs.

The book's lingo is simple. Try this: "The most important thing to know about using functions is how to make them work. Only three conditions need to be met for a function call to succeed. First, the function must have been previously defined in the program. Second, the correct number of parameters must be passed to it. Lastly, the correct object must be present - you cannot call the string object's split function without a string object." A book to invoke before you launch upon your own project.

6.4 Summary

- JavaScript enabled browsers recognizes and uses the Document Object Model (DOM) which makes them capable of recognizing individual objects in an HTML page after the page has been rendered in the browser.
- Style Sheets are a language used for specifying formatting and layout properties for a document.
- When JavaScript Assisted Style Sheets (JSSS) are embedded in an HTML page, the JavaScript DOM picks up a whole new set of objects which add to the standard DOM objects which enables web developers to access every element of a web page whether this element appears on the page when it is rendered in a client browser or not.
- Since JavaScript understands the DOM and can extend the DOM with the use of JSS in a web page JavaScript understands objects such as text box which is an object belonging to the DOM.
- All objects have properties that determine the functionality of the object, methods which allow access to these properties and events to appropriate JavaScript event handlers.
- To handle document objects JavaScript enabled browser creates an array in memory per HTML object thus registering each of the objects.
- JavaScript provides access to the arrays and their elements. Using the Methods of the object, specific object properties can be set too.
- Updating of any web page element based on client input to make the web page interactive can be done by referencing the appropriate element in its associated array, which provides access to each element of the web page.
- JavaScript provides 'Event Handlers' which facilitate the passing of knowledge to the JavaScript that an 'event' has occurred.
- JavaScript event handlers can be divided into two types interactive and non-interactive. JavaScript identifies a web page object first and chooses an appropriate event associated with the object and JavaScript event handlers mapped to an objects event connect the object event and JavaScript code snippets.

Notes

6.5 Keywords

DOM: Document Object Model used by JavaScript enabled browses to identify the collection of web page objects.

JavaScript Event Handlers: JavaScript use event handlers to facilitate the passing of knowledge that an event has occurred from the HTML code to the JavaScript.

JSSS: JavaScript Assisted Style Sheets, which uses JavaScript syntax to control a document's presentation style.

Object Hierarchy: HTML objects belonging to DOM have a descending relationship with each other starting from the browser itself, window, document, form and so on.

Objects: Elements that have properties that determine the functionality of the object, methods that allow access to these properties and events to appropriate event handlers.

6.6 Review Questions

1. List some of the ways a web page event could be associated with the action of the mouse cursor on a web page.
2. Create a web page using two image files, which switch between one another as the mouse pointer moves over the images. Use the onMouseOver and onMouseOut event handlers.
3. JavaScript has the ability to control the behavior of these objects through their properties and methods. Explain
4. What implications does the DOM have for other standards?
5. What is the relationship between stylesheets and the DOM?
6. Will accessibility considerations be an active part of the DOM development?
7. If the DOM is language-neutral, what language do you specify the interface in?
8. What does it mean to be compliant with the DOM specification?
9. Will broken HTML pages work in the DOM? Explain with examples.
10. The onLoad and onUnload events are triggered when the user enters or leaves the page. Explain with a suitable program.

Answers: Self Assessment

1.	web page	2.	FORM
3.	browser	4.	JavaScript
5.	document's	6.	objects
7.	accessible	8.	JavaScript
9.	checkForm()	10.	onSubmit
11.	onMouseOut	12.	server
13.	alert box	14.	Server-side
15.	Form validation		

Notes

6.7 Further Readings

Book

Jeremy Keith, *DOM Scripting: Web Design with JavaScript and the Document Object Model* (Paperback), Friends of ED.

Online links

http://www.w3schools.com/jsref/default.asp

http://www.javascriptkit.com/domref/

Unit 7: Introduction to ASP (Active Server Pages)

Notes

CONTENTS

Objectives
Introduction
7.1 Active Server Pages
7.2 Installing Internet Information Server (IIS)
7.2.1 Using ASP without IIS or PWS
7.3 ASP Variables
7.4 ASP Operators
7.5 ASP Conditional Statement
7.6 Loops
7.7 Select Case
7.8 ASP Arrays
7.9 Summary
7.10 Keywords
7.11 Review Questions
7.12 Further Readings

Objectives

After studying this unit, you will be able to:

- Scan Active Server Pages
- Install IIS
- Discuss ASP variables
- Explain ASP operators
- Demonstrate conditional, loops and case statements
- Describe Arrays

Introduction

Getting started with Active Server Pages: What are Active Server Pages? Understanding the Client Server Model; How ASP differs from Client-side scripting technologies; Running ASP pages (Setting up personal Web Server, Setting up Internet Information Server, Using ASP without IIS or PWS); Creating your first ASP pages.

7.1 Active Server Pages

ASP was "born" in November 1996 when Microsoft announced its design of an Active Platform. The Active Platform reflects Microsoft's ideas about how a desktop computer and a server computer should communicate. It consists of two parts: the Active Desktop and the Active

Notes

Server. The Active Desktop refers to the client side, or the user's side, where HTML files are displayed on a web browser. The Active Server refers to the server-side component. This consists of pages that can be interpreted by the server, hence the term Active Server Pages.

Over the past couple of years, we have seen some major changes concerning the Internet. Initially, the internet served as a medium for members of government and education institutions to communicate. With the advent of the World Wide Web, the internet became a multimedia, user friendly environment. Originally, the internet served as a place for enthusiasts to create personal home pages, but as more people began going online the internet transformed into an informational resource for the common man.

As the internet has matured into a viable market place, Website design has changed in step. In the early days of the World Wide Web, HTML was used to create static Web pages. Now you can create dynamic Web pages in many ways. Microsoft solution to building dynamic Web pages is through the use of ACTIVE SERVER PAGES (ASP).

Active Server Pages contains two parts:

- Programmatic code
- Embedded HTML

The programmatic code can be written in a number of scripting languages.

Did u know? What is Scripting Language?

Scripting language is a particular syntax used to execute commands on a computer.

Some popular Web related scripting languages include VBScript and JavaScript. When creating an ASP page, you can use one of four programming languages:

- VBScript-Similar to visual basic syntax, the most commonly used scripting language for active server pages.
- Jscript-Similar to java script.
- PerlScript-Similar to Perl.
- Python-A powerful scripting language commonly used for Web development.

Most ASP pages are created using VBScript as it has the most English-like syntax of the four scripting languages and is similar to Visual basic syntax.

ASP page contain embedded HTML. This allows for existing static Web pages to be easily converted into dynamic ASP pages. An ASP page must contain an .ASP extension.

Notes Many large Websites use active server pages to serve dynamic Web content. For example, Buy.com and Dell.com use active server pages to build interactive, dynamic Web sites.

Self Assessment

Fill in the blanks:

1. The Active Platform reflects Microsoft's ideas about how a desktop computer and a …… computer should communicate.

Notes

2. The programmatic code can be written in a number of ……
3. In the early days of the World Wide Web, HTML was used to create static ……
4. Asp page contain embedded ……
5. Most ASP pages are created using VBScript as it has the most English-like syntax of the four scripting languages and is similar to ……

7.2 Installing Internet Information Server (IIS)

Internet Information Server (IIS) is Microsoft's professional Web server. The latest version of IIS is 5.0 which ships with Windows 2000. To install IIS 5.0 following steps are done:

- Choose start, programs, administrative tools, configure your server.
- A dialog box appears that has several configuration options in the left pane. Choose the bottommost option from the left pane labeled Advanced.
- On clicking this four new options appear: Cluster service, Message Queuing, Support Tools and Optional Components. Click the last option. In the right hand pane, a description of the Option Components option appears, as well as a hyperlink labeled start.
- Click the Start hyperlink-this launches the Windows Components Wizard. Through this wizard you can install or uninstall optional components. Scroll down till the Internet Information Services (IIS) option is seen.
- To decide what IIS components to install, click IIS components in the Windows 2000 Components Wizard and then click the Details button. A list of components that can be installed with IIS is displayed.
- Common Files, Documentation, Internet Information Services Snap-In and World Wide Web Server should surely be selected.
- After the selection of all the IIS subcomponents that are to be installed, click OK, which takes you back to the Windows 2000 Components Wizard.
- To start installing IIS 5.0, click the Next button. When the installation is complete, one can access the Internet Services Manager.

Task In a group try to install IIS practically into your systems

7.2.1 Using ASP without IIS or PWS

ASP Pages can run even without IIS or PWS as your Web server. A couple of companies have created software to allow ASP pages to run on various Web servers and platforms.

One of these products is Halcyon Software's Instant ASP, abbreviated as iASP. Another product created Chili! Soft, is Chili! ASP. These products can run on many non-IIS Web servers as the following:

- Apache
- Sun Web Server
- Java Web Server
- Netscape Enterprise Server

Notes

These products can also run on a number of platforms, including

- Linux
- Sun Solaris
- Apple Mac OS
- IBM/AIX

Self Assessment

Fill in the blanks:

6. In the right hand pane, a description of the Option Components option appears, as well as a ..... labeled start.
7. ASP Pages can run even without ..... as your Web server.
8. Internet Information Server (IIS) is Microsoft's professional .....

7.3 ASP Variables

A Variable is used to Store Information.

If the variable is declared outside a procedure it can be changed by any script in the ASP file. If the variable is declared inside a procedure, it is created and destroyed every time the procedure is executed.

Different variables in ASP are mentioned below:

Dim 'em first

To "Dim" it means to Dimension it. That's VB lingo. A variable is declared in _VBScript using the Dim keyword.

```
<%
Dim myVar
%>
```

VB programmers will notice here, that we have not included any indication of the type of the said variable. E.g. Dim myString as String, or Dim myString$.

In _VBScript, all variables are variants. Their type is determined automatically by the runtime interpreter, and the programmer need not (and should not) bother with them.

By default, _VBScript does not force requiring variable declaration. That is, it is allowed to use a variable directly without declaring it first. However, experienced programmers know the importance of making it compulsory to declare all your variables first – without that, the bugs that may result are very difficult to detect. Considering this, we have here a simple directive to make variable declaration compulsory.

```
<%
Option Explicit
Dim myVar
%>
```

Remember that Option Explicit must necessarily be the first statement of your ASP page, otherwise a server error is generated.

Notes

To illustrate what I mean, if you had a page that read:

```
<%
Pi = 3.141592654
Response.Write Pi
%>
```

This is a perfectly valid page. You will get the value of Pi written back to the page, as you really expected.

Now, using the Option Explicit directive as above, let's rewrite the same page as follows:

```
<%
Option Explicit
Dim Pi
Pi = 3.141592654
%>
```

Did u know? What are the uses of session variables?

Session variables are used to store information about ONE single user, and are available to all pages in one application. Typically information stored in session variables are name, id, and preferences.

Self Assessment

Fill in the blanks:

9. No scripts outside the procedure can access or change the .....
10. A variable declared outside a ..... can be accessed and changed by any script in the ASP file.

7.4 ASP Operators

Arithmetic Operators

Arithmetic operators are used to perform mathematical calculations.

Operator	Function	Example	Result
+	Addition	aNumber = 2 + 2	aNumber = 4
-	Subtraction	aNumber = 4 - 2	aNumber = 2
*	Multiplication	aNumber = 10 * 10	aNumber = 100
/	Division	aNumber = 32 / 4	aNumber = 8

Comparison Operators

Comparison operators are used to compare two values and make a decision. Comparison operators return a true or false value.

Notes

Operator	Function	Example	Result
=	Equal to	9 = 10	False
<	Less than	aNumber = 5, aNumber < 10	True
>	Greater than	aNumber = 5, aNumber > 10	False
<>	Not equal to	5 <> 10	True

Logical Operators

Logical operators take the result produced by comparison operators and compares them to create new true or false values. For example, true AND false = false.

Operator	Function	Example	Result
AND	True if both are true, otherwise false	9 = 10 AND 10 = 10	False
OR	True if at least one is true, otherwise false	9 = 10 OR 10 = 10	True
Not	Reverses truth value	5 = 5	False

String Operators

There is only one string operator and that is the concatenation operator"&". This operator is used to combine text strings.

Example: `<% Dim stringOne = "This is " Dim stringTwo = "a text string" Response. Write (stringOne & stringTwo & "<br />") Response.Write("ASP stands for " & "Active " & " Server " & " Pages ") %>`

Output:

This is a text string ASP stands for Active Server Pages

7.5 ASP Conditional Statement

When performing assignments using code, sometimes you must find out whether a given situation bears a valid value. This is done by checking a condition. To support this, VBScript provides a series of words that can be combined to perform validations. Checking a condition usually produces a True or a False result. Once the condition has been checked, you can use the result (as True or False) to take action. Because there are different ways to check a condition, there are also different types of keywords to check different things. To use them, you must be aware of what each does or cannot do so you would select the right one.

The Boolean Type

Sometimes, you may want to store the result of a condition, being true or false, in a variable. To do this, you can declare a variable that would hold a Boolean value. Here is an example of declaring a Boolean variable when the form opens:

Dim IsMarried

When a Boolean variable has been declared, you can assign it a True or a False value.

To convert a value or an expression to Boolean, use the CBool() function.

Techniques of Checking a Condition

Notes

The If...Then Statement

The simplest technique used to validate a condition is to check whether it is true. This can be done using an If...Then statement. The formula to use is:

If ConditionToCheck is True Then Statement

The program examines a condition, in this case ConditionToCheck. This ConditionToCheck can be a simple expression or a combination of expressions. If the ConditionToCheck is true, then the program will execute the Statement.

There are two ways you can use the If...Then statement. If the conditional formula is short enough, you can write it on one line, like this:

If ConditionToCheck is True Then Statement

If there are many statements to execute as a truthful result of the condition, you should write the statements on alternate lines. Of course, you can use this technique even if the condition you are examining is short.

Notes In this case, one very important rule to keep is to terminate the conditional statement with End If.

The formula used is:

```
If ConditionToCheck is True Then
   Statement
End If
```

The If...Then...Else Statement

The If...Then statement offers only one alternative: to act if the condition is true. Whenever you would like to apply an alternative in case the condition is false, you can use theIf...Then...Else statement. The formula of this statement is:

```
If ConditionToCheck is True Then
   Statement1
Else
   Statement2
End If
```

When this section of code executes, if the ConditionToCheck is true, then the first statement,Statement1, is executed. If the ConditionToCheck is false, the second statement, in this

caseStatement2, is executed.

The If...Then...ElseIf Statement

The If...Then...ElseIf statement acts like the If...Then...Else expression, except that it offers as many choices as necessary. The formula to use is:

If Condition1 is True Then

Notes

Statement1

ElseIf Condition2 is True Then

Statement2

ElseIf Conditionk is True Then

Statementk

End If

The program will first examine Condition1. If Condition1 is true, the program will executeStatment1 and stop examining conditions. If Condition1 is false, the program will examineCondition2 and act accordingly. Whenever a condition is false, the program will continue examining the conditions until it finds one. Once a true condition has been found and its statement executed, the program will terminate the conditional examination at End If.

There is still a possibility that none of the stated conditions is true. In this case, you should provide a "catch all" condition. This is done with a last Else section.

Caution The Else section must be the last in the list of conditions and would act if none of the primary conditions is true.

The formula to use would be:

If Condition1 is True Then

Statement1

ElseIf Condition2 is True Then

Statement2

ElseIf Conditionk is True Then

Statementk

Else

CatchAllStatement

End If

Self Assessment

Fill in the blanks:

11. The simplest technique used to validate a ..... is to check whether it is true.
12. Whenever a condition is ....., the program will continue examining the conditions until it finds one.

7.6 Loops

For-Next LoopsFor-Next Loops

The syntax is as follows

```
<%
For I = 1 to 10
```

Notes

```
        Response.Write "Number = " & I & vbCrLf
        Next
%>
```

And the output ...

```
        Number = 1
        Number = 2
        Number = 3
        Number = 4
        Number = 5
        Number = 6
        Number = 7
        Number = 8
        Number = 9
        Number = 10
```

The vbCrLf used in the statement above is a predefined constant that equals the combination of the Carriage-Return character (CR for short), and the Line Feed character (LF for short.) Using it causes the output to continue on the next line.

Without the vbCrLf, our output would have appeared on one long line:

```
        Number = 1Number = 2Number = 3Number = 4Number = 5Number = 6Number
        = 7Number = 8Number = 9Number = 10
```

Let us take a case of nested loops to clarify things:

```
        <%
        For I = 1 to 8
        For j =1 to 8
        Response.Write "X"\
        Next
        Response.Write vbCrL
        Next
        %>
```

This will draw a nice chessboard pattern on the screen. (You will need to view the source of the page in your browser however. If you look at the page in the browser itself, you will not see the true result.)

Notes A very important point to note is that the Next statement that completes the For does not take an argument. You cannot say:

```
Next I
Or
Next J
```

This is invalid. Each Next statement encountered is automatically assumed to complete the immediately preceding For statement.

Notes

Finally, _VBScript also allows the Step keyword to modify the interval or step-size of the For-loop variable.

```
<%
For I = 1 to 10 Step 2
Response.Write "Number = " & I & vbCrLf
Next
%>
```

gives you:

```
Number = 1
Number = 3
Number = 5
Number = 7
Number = 9
```

The loop counted I in steps of 2, thus taking on only odd values from the entire set of 10.

For Each Object In Collection ...

The For-Each construct is unique to _VBScript (and its parent, Visual Basic, of course!) It allows you to iterate through the items in a collection one by one.

```
<%
For Each Member in Team
Response.Write Member
Next
%>
```

Here, Team is assumed to be a collection of items. This statement is very useful in scenarios, where the size of the collection is not known in advance. Using the For Next statement assures that all items in that collection will be processed, and no "Array Index Out Of Bounds" errors will be generated.

While ... Wend

Again, here is one of the popular looping constructs of all time.

```
<%
While Not RS.EOF
Response.Write RS.Fields ("Name")
RS.MoveNext
Wend
%>
```

or,

```
<%
Do While Not RS.EOF
Response.Write RS.Fields ("Name")
RS.MoveNext
```

Notes

```
Loop
%>
```

The While statement executes the statements in its loop until the given condition remains true. The moment it becomes false, the loop halts.

Remember to end the While Statement with the Wend Keyword.

A variation of the While loop that tests the condition after the loop is the Do-loop.

```
<%
Do
TimePass()
Until ExamDate - Now = 30
%>
```

I hope the meaning is amply clear from the example above.

Self Assessment

Fill in the blank:

13. Using the ..... statement assures that all items in that collection will be processed, and no "Array Index Out Of Bounds" errors will be generated.

7.7 Select Case

To make a choice between a set of items that can be assigned to a variable, use the Select Case statement.

```
<%
    Select Case Choice
    Case "1":
    Response.Write "You chose 1"
    Case "2":
    Response.Write "You chose 2"
    Case "3":
    Response.Write "You chose 3"
    Case "4":
    Response.Write "You chose 4"
    End Select
%>
```

7.8 ASP Arrays

Arrays in ASP follow the exact same form and rules as those arrays in VBScript. You can create an array of specific size or you can create a dynamic sized array. Below we have examples of both types of arrays.

Notes

ASP Code:

<%

Dim myFixedArray(3) 'Fixed size array

Dim myDynArray() 'Dynamic size array

%>

We are going to focus on fixed size arrays first and cover dynamic arrays later on in this lesson.

Assigning values to an array

Let's fill up our fixed size array with values. Our fixed array is going to store the names of famous people. To assign a value to an array you need to know three things:

- The name of the array
- The value you want to store
- The position in the array where you want to store the value.

An array is a group of variables that you can access by specifying the position inside the array. Our array myFixedArray has four positions: 0, 1, 2 and 3. Let's assign some values to our array.

Example:

ASP Code:

```
<%
Dim myFixedArray(3) 'Fixed size array
myFixedArray(0) = "Albert Einstein"
myFixedArray(1) = "Mother Teresa"
myFixedArray(2) = "Bill Gates"
myFixedArray(3) = "Martin Luther King Jr."
%>
```

ASP Arrays are Zero based!

If you're a programmer you might look at that above example and thing "Hey, you only allocated three elements for that array, but you assigned four values! Well, this can be done because arrays in ASP are zero based. This means when you declare fixed array of size X then you can assign values for each value from 0 through X.

Below is perfectly functional ASP code that will go through our array and print out the contents of each element.

Example:

ASP Code:

```
<%
Dim myFixedArray(3) 'Fixed size array
myFixedArray(0) = "Albert Einstein"
```

Notes

```
myFixedArray(1) = "Mother Teresa"
myFixedArray(2) = "Bill Gates"
myFixedArray(3) = "Martin Luther King Jr."
For Each item In myFixedArray
       Response.Write(item & "<br />")
Next
%>
```

Display:

Albert Einstein Mother Teresa Bill Gates Martin Luther King Jr.

ASP Dynamic Sized Arrays

To create an array whose size can be changed at any time simply do not put a number within the parenthesis when you declare the array. When you know what size you want the array to be use the ReDim keyword. You may ReDim as many times as you wish.

If you want to keep your data that already exists in the array then use thePreserve keyword. Below is an example of all these things we have just talked about.

Example:

ASP Code:

```
<%
Dim myDynArray() 'Dynamic size array
ReDim myDynArray(1)
myDynArray(0) = "Albert Einstein"
myDynArray(1) = "Mother Teresa"
ReDim Preserve myDynArray(3)
myDynArray(2) = "Bill Gates"
myDynArray(3) = "Martin Luther King Jr."
For Each item In myDynArray
       Response.Write(item & "<br />")
Next
%>
```

Self Assessment

Fill in the blanks:

14. To create an array whose size can be changed at any time simply do not put a number within the ..... when you declare the array.
15. An array is a group of variables that you can access by specifying the ..... inside the array.

Notes

Caselet

Google Blocks Web Worm Santy.A

GOOGLE Inc has announced that it has blocked Santy.A, the Web worm which had identified potential victims through its search and had spread among online bulletin boards using vulnerability in phpBB, an open-source software product managed by the phpBB Group.

The Santy worm is the first to use a popular search engine to propagate itself.

The worm apparently worked by sending Google a specific search request, asking for a list of vulnerable sites. On obtaining a list, the worm spread to the sites in it by using a PHP request designed to exploit the vulnerability of the phpBB bulletin board software. On infecting a Web site, Santy searched Google for other sites running phpBB. and tried to infect those sites too. After Santy took over a site, it deleted all HTML, PHP, active server pages (ASP), Java server pages (JSP), and secure HTML pages, and replaced them with the text, "This site is defaced!!! This site is defaced!!! NeverEverNoSanity WebWorm generation X."

For X, the worm inserted a number representing its 'generation' - that is, how far it had descended from the original worm release. According to one report, MSN searches had suggested the existence of 24 generations of the worm.

Further, a phpBB component called viewtopic.php allowed malicious commands to be passed to and executed on servers running a vulnerable version of the phpBB software.

The worm infected Web sites - but did not infect computers used to view those sites.

According to antivirus companies, Google has been successful in blocking the worm as Santy. A does not have any native ability to scan for vulnerable computers.

They further point out that the worm is yet another instance of the practice known as Google hacking which uses the search major's service as an attack tool.

As it happens, the numero uno of search is also one of the most popular search engines among hackers who often use it to find vulnerable targets for an attack. For instance, attackers, by searching for default server page titles, are able to find servers which can be exploited easily. Applications left in default modes can also be found by searching for error pages generated by the software. Searches on Google for specific file names can also identify vulnerable servers hooked up to the Internet.

Ironically, it is the very features that have made Google the most popular search engine in the world that makes hackers use it. Most other search engines do not have the advanced search option available on Google and do not cache old versions of Websites.

Security experts point to the spread of Santy to underline the need to keep on top of software patches and "harden" the configuration of public-facing servers.

7.9 Summary

- The Active Platform reflects Microsoft's ideas about how a desktop computer and a server computer should communicate. It consists of two parts: the Active Desktop, the client side, and the Active Server, the server side.
- The Active Server consists of pages called the Active Server pages that can be interpreted by the server. Active Server Pages contains two parts which are programmatic code and embedded HTML

Notes

- When creating an ASP page, you can use one of four programming languages which are VBScript, Jscript, PerlScript and Python. Most ASP pages are created using
- ASP page contain embedded HTML which allows for existing static Web pages to be easily converted into dynamic ASP pages. An ASP page must contain an .ASP extension.
- To execute ASP pages on the computer, you need to be running a Web server capable of handling ASP pages. The Server processes the request to ASP page before it is sent to the client. During this processing the programmatic code in the ASP page is interpreted by the Web server and the Web server informs the browser that it is sending HTML information and sends the output of the ASP page. The browser, receiving this raw HTML renders it for the user.
- If you are creating a professional website, it is important that the web site run on computer that has Windows NT Server or Windows 2000 installed. You can install Personal Web Server (PWS) which is a stripped down version of professional server intended to run on Microsoft Windows 95 or 98 or Microsoft Windows NT workstation or you can choose to install the Internet Information Server (IIS) which is Microsoft's professional Web server.
- ASP Pages can run even without IIS or PWS as your Web server. Software such as Halcyon Software's Instant ASP (iASP), Chili! ASP allows ASP pages to run on various Web servers and platforms. These products can run on many non-IIS Web servers and also on a number of platforms

7.10 Keywords

Active Server Pages: ASP is used to create dynamic pages that interact with a database. It contains two parts which are programmatic code and embedded HTML.

Internet Information Server (IIS): It is Microsoft's professional Web server.

Jscript: Similar to java script and used as scripting language for active server pages.

PerlScript: Similar to Perl and used as scripting language for active server pages.

Personal Web Server (PWS): It is a stripped down version of professional server intended to run on Microsoft Windows 95 or 98 or Microsoft Windows NT workstation.

Python: A powerful scripting language commonly used for Web development.

The Active Platform: It reflects Microsoft's ideas about how a desktop computer and a server computer should communicate. It consists of two parts: the Active Desktop, which refers to the client side, and the Active Server, which refers to the server side.

The Active Server: It refers to the Server side and consists of pages called the Active Server pages that can be interpreted by the server.

VBScript: Similar to visual basic syntax, the most commonly used scripting language for active server pages.

Web Server: It is a computer that contains all the web pages for a particular web site and has special software installed to send these web pages to web browsers that request them.

7.11 Review Questions

1. You can run Active Server pages on your own computer without an external server. How would you do that?

Notes

2. What happens when you don't have a web server installed and you try to view ASP pages on your browser?
3. What is the Internet Information Server? How would you set it up?
4. How would you run ASP pages without installing IIS or PWS as your web server?
5. Explain the difference in the way when a browser visits a static web page and when the browser requests an ASP page.
6. Explain how ASP differs from client side scripting technologies? Who can create dynamic content with Active Server Pages?
7. What do I need to run Active Server Pages? What can Active Server Pages do for my business?
8. What ActiveX Server Components are supported? On which platforms does Active Server Pages run?
9. What data sources can my Web application integrate with? How does Active Server Pages compare to CGI?
10. How does Active Server Pages compare to ISAPI applications? How does Active Server Pages compare to PERL?
11. What does Active Server Pages do better than other Web application tools? How does Active Server Pages compare to Netscape LiveWire?

Answers: Self Assessment

1.	server	2.	scripting languages
3.	Web pages	4.	HTML
5.	Visual basic syntax	6.	Hyperlink
7.	IIS or PWS	8.	Web server
9.	Variable	10.	Procedure
11.	Condition	12.	False
13.	For Next	14.	Parenthesis
15.	position		

7.12 Further Readings

Books

A Keyton Weissinger, *ASP in a Nutshell*, 2nd Edition [ILLUSTRATED] (Paperback), O'Reilly Media; 2 edition (January 1, 2000)

Bob Reselman, *Active Server Pages 3.0 by Example,* Que Publishing

David Buser, Chris Ullman, Brian Francis, *Beginning Active Server Pages 3.0,* Dave Sussman, John Wiley & Sons

Notes

John W. Gosney, *ASP Programming for the Absolute Beginner* (For the Absolute Beginner) (Paperback), Course Technology PTR; 1 edition (July 15, 2002)

Keith Morneau, Jill Batistick, *Active server pages,* Course Technology

Scott Mitchell, *Designing Active Server pages,* O'Reilly Media

Online links

http://msdn.microsoft.com/en-us/library/aa286483.aspx

http://www.w3schools.com/asp/default.asp

Notes

Unit 8: ASP Web Forms

CONTENTS

Objectives

Introduction

8.1 Using Form Fields

8.2 Introduction to CGI Forms and CGI

8.3 Understanding Client-Server Scripting

8.3.1 How ASP differs from Client-Side Scripting Technologies?

8.4 Building and Processing Web Forms

8.5 Summary

8.6 Keywords

8.7 Review Questions

8.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Scan the introduction to CGI
- Describe Client-Server side scripting
- Demonstrate building and processing web forms

Introduction

A form has two duties:

1. To collect information from the user and to send that information to a separate Web page for processing.
2. Through the use of form, an ASP page can acquire the user's input, and make programmatic decisions based on that input.

Creating form is straightforward and simple. It requires as little as two lines of HTML.

It is created using the <FORM> Tag

```
1:  <FORM METHOD= POST ACTION= "somepage.asp">
2:  </FORM>
```

Caution The above form is useless and serves no function. It has no text boxes for users to enter information into. It has no list boxes, radio buttons or check boxes either.

On a website, this form would be useless; however it does demonstrate how a <FORM> tag is used to collect information from the user.

Notes

Each text box, list box, check box or radio button in a form is a field. We need a way to create form fields within our form.

8.1 Using Form Fields

To create text boxes, check boxes, and radio buttons, we should use <INPUT> tag. The <INPUT> tag has a number of properties, but we will only concentrate on the following three:

- NAME- The name tag uniquely identifies each element in the form.
- TYPE- The TYPE tag determines what type of form field is displayed. To display a text box, we should set TYPE equal to TEXT. To create a check box we should assign TYPE equal to CHECKBOX.
- VALUE- The value tag determines the default value for the form field. This property is important when processing the information submitted by list boxes, check boxes, and radio buttons.

Designing Forms

A few things are important worth considering before designing form:

1. To make sure that the form has a submitting button.
2. The form should be easy for the user to complete. We have four form fields at your disposal: the text box list box check box and radio button. Make sure that the type of form field used best fits the information needing to be collected. For example, if we are going to ask for a user's mailing address it makes more sense to have a list box that contains the 50 states than to have the user type in the name of his or her state of residency.

Example: The first thing to do is to ask," What information do I need from the user?" In this case the information needed would be user's name, street address, city, state and zip code. We may also want to obtain some background information on the user as well, such as how often he purchase widgets.

Based on these understandings we need following form fields:

- Text boxes would work well for entering the first and last names of the customer, as well as for the street address and zip code.
- A list box should be employed so that the users can select their states, instead of typing state names or postal abbreviations list box, containing 50 states, may seem unwieldy, but resist the temptation to use a text box. When using a text box, we are allowing users to enter anything, where a list box requires them to choose one of the viable options. If users are permitted to type in the names of their states, several different values can be received for one particular state.
- Radio buttons would work well for the background information on the user's widget buying habits.
- A simple check box would suffice for the online newsletter subscription.

Send Information by Using Forms

A common use of intranet and Internet server applications is to process a form submitted by a browser. With ASP, we can embed scripts written in VBScript directly into an HTML file to process the form. ASP processes the script commands and returns the results to the browser.

Notes

Using a standard web browser, a user can surf to a web page with a form on it and enter information when the user does this, the information he/she id typing in has not yet been sent to the Web server. This information is not available for the web sever to process until the user submits the form by checking the form's submit button.

Self Assessment

Fill in the blanks:

1. To collect information from the user and to send that information to a separate ……. for processing.
2. The value tag determines the …… value for the form field.
3. The form should be easy for the user to ……
4. …… would work well for the background information on the user's widget buying habits.
5. A …… should be employed so that the users can select their states.
6. A common use of intranet and Internet server applications is to process a form submitted by a ……

8.2 Introduction to CGI Forms and CGI

The communication for static **HTML** works only one way. There is no way to send information back to a Web server. To fix this problem, **forms** and **CGI** were created. **Forms** are **HTML** tags that allow Web page creators to include controls like check boxes, and radio buttons in their Web pages. That way, the user can enter information. It also provides a Submit button that sends the information off to the server.

But now the server has to be smarter, too. It can't just get requests for pages and send out pages. The server has to know what to do with this form information when it gets it. That's where **CGI** comes in.

CGI stands for **Common Gateway Interface**. **CGI** makes it possible for the Web server to talk to another application that can handle the form information when it's sent back. Often these **CGI** applications are written in a language called **Perl**. When the **CGI** application receives the form information, it can save it to a text file or store it in a database.

This system works great for simple guest books, but if we want to make our Web pages really interactive, then we will soon start running into big trouble.

The problem with **CGI** is that if five people are submitting form information at the same time, five different copies of the **CGI** application have to be running on the server to handle them. If a hundred people are submitting form information at once - a hundred copies of the application run at the same time. This is a great way to make a popular Web server fall to its knees, start crawling slowly and then fall over.

Did u know? How to read Form Values from an ASP Page?

The RequestQuery.String and Request.Form commands may be used to retrieve information from forms, like user input.

Notes

Self Assessment

Fill in the blanks:

7. CGI stands for ……

8. When the CGI application receives the ……, it can save it to a text file or store it in a database.

9. Forms are …… that allow Web page creators to include controls like check boxes, and radio buttons in their Web pages.

8.3 Understanding Client-Server Scripting

In a client-server model, two computers work together to perform a task. A client computer requests some needed information from a server computer. The server returns this information and the client acts on it. The internet runs on a client-server model. With internet the server is a particular Web server and the client is a Web browser.

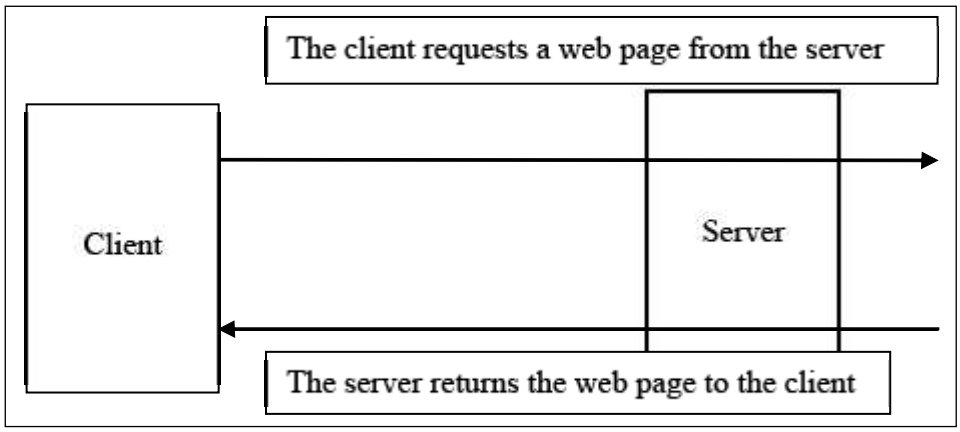
A web server is a computer that contains all the web pages for a particular web site and has special software installed to send these web pages to web browsers that request them.

The following steps occur when a web browser visits a static web page:

- The client (web browser) locates the web server specified by the first part of the URL(www.something.com)
- The client then requests the static web page specified by the second part of the URL (/index.htm)
- The web server sends the contents of that particular file to the client in HTML format.
- The client receives the HTML sent by the server and renders it for the user.

Task But client and server script are generally independent. Analyze what does the author wants to say from this statement? Give examples to support your answer

Figure 8.1

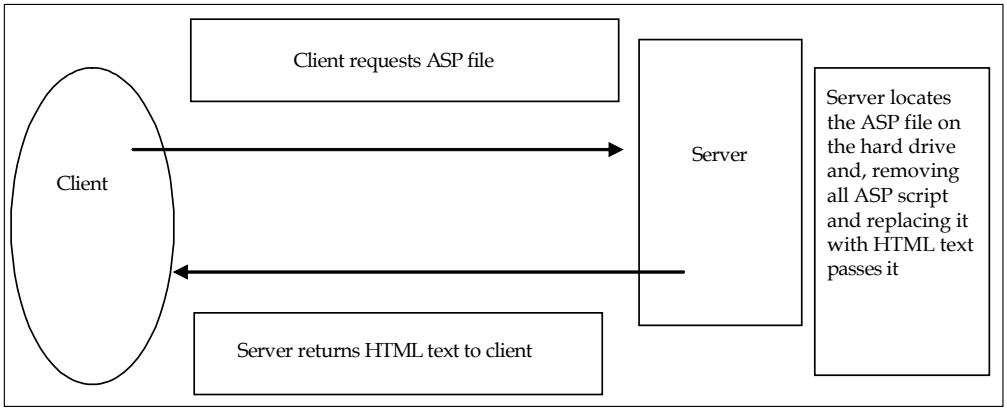


Notes

When a web browser requests an ASP page, the following steps occur:

- The client locates the web server specified by the first part of the URL(www.something.com)
- The client then requests the ASP page specified by the second part of the URL(/default.asp)
- The web server reads the ASP file and processes the code.
- After the ASP page has been completely processed by the web server, the output is sent in HTML format to t==he client.
- The client receives the HTML sent by the server and renders it for the user.

Figure 8.2



8.3.1 How ASP differs from Client-Side Scripting Technologies?

Client-Side Scripting is programmatic code in an HTML file that runs on the browser. It is denoted by <SCRIPT> HTML tag. It is commonly written using the JavaScript programming language due to the fact that Netscape Navigator only supports the JavaScript scripting language for Client-Side scripting. ASP scripts are server -side scripts.

While using ASP, its code exists on the server only. ASP code, which is surrounded by the <% and %> delimiters is processed completely on the server. The client cannot access this ASP code.

Notes Note that the Server-side script is also used to create applications which need a broad reach: applications that might be accessed by many different types of browsers, each with different capabilities.

Differences

A Client-Side script is processed only by the client. It is the client's responsibility to execute all Client-Side scripts on the other hand; Server-Side scripts are processed completely on the web server. The client does not receive any code from server side scripts rather the client receives just the output of the Server-Side scripts.

Client-Side scripts and Server-Side scripts cannot contract with one another because the Client-Side script are executed on the client, after the service side scripts have finished processing completely.

Notes

Self Assessment

Fill in the blanks:

10. A client computer requests some needed information from a …… computer.
11. …… is programmatic code in an HTML file that runs on the browser.
12. The client does not receive any code from server side scripts rather the client receives just the output of the ……

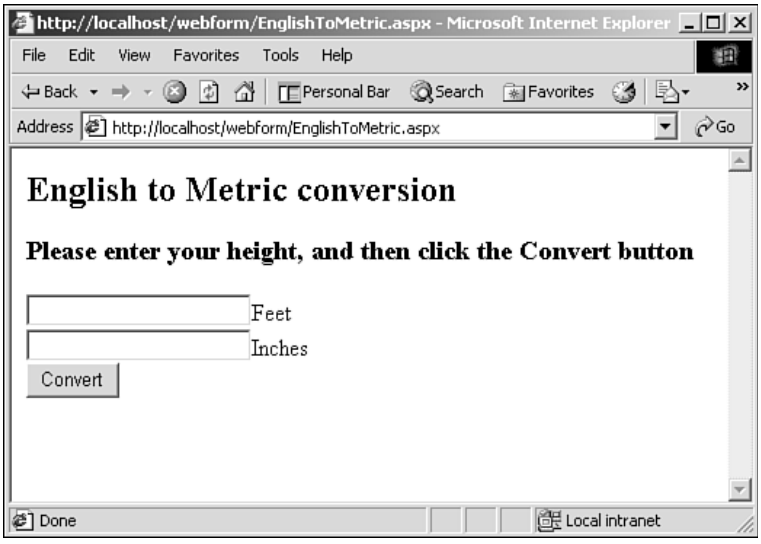
8.4 Building and Processing Web Forms

Understanding how Web Forms are Processed

A Web form is another name for an ASP.NET page. A Web form can be made up of a single file with an .aspx extension. An .aspx file and a code behind file can be combined to make a Web form. As we'll show in the following few days, Web forms can also contain your own custom controls, defined similarly to .aspx pages, called user controls.

Because a Web form can include HTML, ASP.NET Web controls, custom (user) controls, and code behind, creating them can get complicated pretty quickly. However, we need to remember only a few key ideas about ASP.NET page processing so that we can develop and debug effectively.

The first key idea is that much of ASP.NET infrastructure is set up so that a single ASP.NET page can post form data back to itself repeatedly. This technique is most useful when we divide our Web site's functionality into a few main pages.



The second key idea to remember is that all ASP.NET pages are eventually compiled into executable files by the ASP.NET infrastructure. This means that every time a page is processed and rendered, a small program corresponding to each Web form is executed by the ASP.NET infrastructure.

Let's explore in more detail how Web forms are processed. Listing shows a sample Web form that performs an English unit to metric unit conversion, and this is shown below.

Notes

Did u know? What is the ACTION Property, Client-Side Form Validation

We have covered some of the basics of forms; it's time to discuss some of the more advanced form techniques. We've focused most of our attention so far on the various type of form fields. Each form field is responsible for gathering a discrete chunk of information from the user, while the form is responsible for gathering this information and sending it to an ASP page for processing.

To accomplish this, the <FORM> tag has two properties: Method and Action. The METHOD property determines how the information is passed to the form processing script. As we know that Method property can be set to either GET or POST. The ACTION property determines to what form processing script to send the users' information.

Listing : EnglishToMetric.aspx: Converting Feet and Inches to Meters

```
<%@ Page Language="C#" %>
<html>
<body>
<form runat="server">
<h2>English to Metric conversion</h2>
<h3>please enter your height, and then click the Convert button</h3>
<asp:Textbox id="Feet" runat="server"/>Feet
  <br>
<asp:Textbox id="Inches" runat="server"/>Inches
<br>
<asp:Button OnClick="OnConvert" Text="Convert" runat="server"/>
<br>
<br>
<asp:Label id="lblMeters" runat="server"/>
</form>
</body>
</html>

<script runat="server">
void Page_Load(Object Sender, EventArgs e)
{
 if(IsPostBack) {
 if(Feet.Text == "") {
Feet.Text = "0";
}
if(Inches.Text == "") {
 Inches.Text = "0";
```

Notes

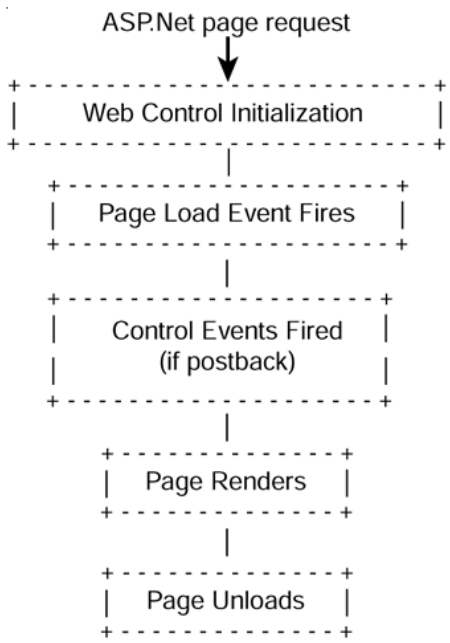
```
28:  }
29:  }
30:  }
31:
32:  void OnConvert(Object Sender, EventArgs e)
33:  {
34:  Single fFeet = Single.Parse(Feet.Text);
35:  Single fInches = Single.Parse(Inches.Text);
36:  Double fMeters = 0.305*fFeet + 0.0254*fInches;
37:  lblMeters.Text = "<b>You are " + fMeters.ToString() + " meters tall</b>";
38:  }
39:  </script>
```

Let's examine how Web forms work using above Listing as our reference. Web forms like Listing 1 are processed in two different ways:

- A Web form is rendered when the user initially browses to the .aspx file. In this case, the Web form doesn't process any events because there has been no user interaction.
- A Web form may be processed after the user interacts with one of the page controls. For instance, the user might click a button on the page or select an item from a drop-down list. When the user does so, the same Web form gets hit by the user's browser, this time with information about what the user has done.

Through both cases, a Web form goes through five distinct stages when it's rendered, as shown in Figure 8.3.

Figure 8.3: Web forms Processing Stages



Notes

In Figure 8.3, the first stage deals with Web control initialization. We don't need to be too concerned about this stage in our own programming tasks.

Our code is frequently involved in the second stage of Web forms processing, when the ASP.NET infrastructure fires the Load event in the Page class. As Listing 1 shows, we can use the Page_Load event (Lines 20-30) to change the contents of controls or to perform any other kinds of processing before a page is rendered. If the page posts back to itself (that is, if the user clicks the Convert button), the Page_Load event will fill in zeros if the user leaves any field blank (Lines 22-27).

The third stage of Web forms processing is for control event handling, which happens only if the user manipulates one of the Web controls on the page. This stage won't happen if this is the first time the user is browsing this particular page. In above Listing, the On Convert method is called during this stage (Lines 32-38). Note that event handling happens after the Page_Load event. In the fourth stage, the HTML for the page is rendered. This process involves calling the Render method for the page, which will call the Render method for every control, among other things.

Notes Listing 1 doesn't contain an explicit Render method because the page class and Web control classes have default implementations for it already. We will need to override the Render method only if we want to customize the exact HTML that each class produces.

In the last stage, the Page_Unload event for every gets called. We can use the Unload event to clean up database connections and other objects that have to be closed explicitly, for example. (Listing 1 also doesn't define the Page_Unload event because it doesn't need to clean up any objects.)

Self Assessment

Fill in the blanks:

13. A …… is another name for an ASP.NET page.
14. A Web form may be processed after the user interacts with one of the …… controls.
15. A Web form is rendered when the user initially browses to the …… file.

Caselet

Get the Writing Craft Right

TECHNICAL Writing Process and Product, by Sharon and Steven Gerson, offers numerous tools for writing successful correspondence on the job. A sampler:

- For most of us, when we get out that piece of paper or turn on the computer, one of two things happens.

Either we stare at the blank page or screen until beads of sweat form on our foreheads and we break out in hives because of the dreaded blank page syndrome, or we wander about aimlessly for several pages.

- A good rule of thumb is to write to express, not to impress; write to communicate, not to confuse. If your reader must use a dictionary, you're not writing clearly.

Contd...

Notes

- When it's time for you to look for a job, how will you begin your search? You know you can't just wander up and down the street, knocking on doors randomly. That would be time-consuming, exhausting, and counterproductive.
- Even pop-up cans of soda include instructions: "1. Lift up. 2. Pull back. 3. Push down." When such simple items need instructions, you can imagine how necessary instructions are for complex products and procedures.
- Professionals in electronics, engineering, medical fields, the computer industry, and other technologies often rank the ability to communicate as highly as they do technical skills.
- The best delivery by the most polished professional will lack credibility if the speaker has nothing to say.

We have ample examples of the last thing when we watch seasoned politicians answering queries or bureaucrat spokesmen warding off the press.

For the ASPirants

DOUGLAS J. Reilly's Designing Microsoft ASP.NET applicationsgives expert guidance — as it is from Microsoft Press — on how to use the powerful new functionality of ASP.NET (the next generation of Active Server Pages) to build dynamic, scalable Web-based applications. A few picks:

Compiled languages allow the developer to verify that code is at least syntactically correct. ASP doesn't provide any such facility, so simple syntax errors might not be caught until the first time the code is executed.

When developing real-world systems today, you'll encounter two significant problems: one is the problem of making software work on multiple platforms, and the other is the problem of enabling the various pieces of an application written in different languages to communicate.

Using ASP.NET gives you access to the newest and greatest in many areas of server programming, but if you want to use ASP.NET to display a message box, you're out of luck. However, there is a way to present message boxes and other similar close user interactions. The answer is client-side scripting.

The ability to create data-driven, dynamic content is at the heart of what made ASP such a hit. ASP.NET continues the tradition of easy data access. It adds integrated, pervasive support for XML.

Read it if you're not already gasping for breath.

Net concepts

DOUGLAS E. Comer's Computer networks and Internets is an introductory text that uses analogies and examples to define concepts instead of sophisticated mathematical proofs. A few picks:

- In practice, no electronic device can produce an exact voltage or change from one voltage to another instantly. Furthermore, no wire conducts electricity perfectly — as electric current travels down the wire, the signal loses energy.
- In most countries, laws prevent an organisation from connecting sites with optical fibre unless the organisation owns all the property between the sites and the fibre does not need to cross a public street.

Contd...

Notes

- When applied to a network, the term public refers to the availability of the service, not the data transferred. In particular, a public network offers private communication.
- How does TCP achieve reliability? The answer is complex because TCP uses a variety of techniques to handle parts of the problem. One of the most important techniques is retransmission.
- Application programs that communicate across a network or Internet all use a single form of interaction. The interaction is called the client-server paradigm. A program that passively waits for contact is called a server, and a program that actively initiates contact with a server is called a client.
- Because no absolute definition of secure network exits, the first step an organisation must take to achieve a secure system is to define the organisation's security policy.

The policy does not specify how to achieve protection. Instead, it states clearly and unambiguously the items that are to be protected.

A book that can make you understand what the techies talk about.

(Books courtesy: WordPower, Chennai. www. wppublishers.com)

Tailpiece

"The minister told me what we should do first, after receiving crores of money as drought relief."

"I understand."

8.5 Summary

- Most web applications involve both client-side and server-side script. Client-side script is often used to program the user interface for an application, for example, to change a web page's text dynamically, respond to user actions such as button clicks, and perform such client-oriented tasks as input validation.
- Client-side script executes locally in the browser, which provides the user with a lively and responsive interface.
- The web server is not only responsible for processing HTML tags but also the ASP scripts.
- The ASP engine, installed on the web server, is responsible for processing ASP scripts.
- Because the browser can only display HTML compatible text, it is the responsibility of ASP engine to process the ASP scripts and send the results back as HTML text to the client. In other words, the Web server sends HTML text, part of which may be the result of the ASP scripts.
- A Web form can be made up of a single file with an .aspx extension.
- An .aspx file and a code behind file can be combined to make a Web form.
- As we'll show in the following few days, Web forms can also contain your own custom controls, defined similarly to .aspx pages, called user controls.

8.6 Keywords

ASP: Active Server Pages

CGI: Common Gateway Interface

Notes

HTML: Hypertext Markup Language

URL: Universal Resource Locator

8.7 Review Questions

1. Server-side script, in contrast, is usually used to program an application's back end. Explain
2. To create text boxes, check boxes, and radio buttons, we should use <INPUT> tag. Discuss all with the proper programs.
3. How to invoke a server-side function with ASP?
4. What are the few things which are important worth considering before designing form?
5. Explain how to send information by Using Forms?
6. How CGI makes it possible for the Web server to talk to another application that can handle the form information when it's sent back? Discuss
7. How ASP differs from Client-Side Scripting Technologies?
8. The client locates the web server specified by the first part of the URL. Scrutinize
9. Examine how an .aspx file and a code behind file can be combined to make a Web form?
10. While using ASP, its code exists on the server only. Analyze by giving suitable examples

Answers: Self Assessment

1.	Web page	2.	default
3.	complete	4.	Radio buttons
5.	list box	6.	browser
7.	Common Gateway Interface	8.	form information
9.	HTML tags	10.	Server
11.	Client-Side Scripting	12.	Server-Side scripts
13.	Web form	14.	Page
15.	.aspx		

8.8 Further Readings

Books

A Keyton Weissinger, *ASP in a Nutshell*, 2nd Edition [ILLUSTRATED] (Paperback), O'Reilly Media; 2 edition (January 1, 2000)

Bob Reselman, *Active Server Pages 3.0 by Example,* Que Publishing

David Buser, Chris Ullman, Brian Francis, Dave Sussman, *Beginning Active Server Pages 3.0,* John Wiley & Sons

John W. Gosney, *ASP Programming for the Absolute Beginner* (For the Absolute Beginner) (Paperback), Course Technology PTR; 1 edition (July 15, 2002)

Notes

Keith Morneau, Jill Batistick, *Active server pages,* Course Technology

Scott Mitchell, *Designing Active Server pages*, O'Reilly Media

Online links

http://msdn.microsoft.com/en-us/library/aa286483.aspx

http://msdn.microsoft.com/en-us/library/ms972337.aspx

http://www.w3schools.com/asp/default.asp

Notes

Unit 9: ASP Objects

CONTENTS

Objectives

Introduction

9.1 What is an Object?

9.1.1 The Building Blocks of Objects (Properties, Methods, Instances of Objects)

9.2 ASP Server Object

9.3 ASP Session Object

9.4 ASP Application Object

9.5 ASP the Global.asa file

9.6 # include

9.7 Summary

9.8 Keywords

9.9 Review Questions

9.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Scan the ASP objects
- Describe Server Object
- Demonstrate the Session Object
- Recognize the Application object
- Explain the purpose of Global.asa File
- Discuss # include

Introduction

In our study of objects, we will find that an object is a software representation of a real-world item, or object. Software objects feature properties (that describe the object), methods (that allow the object to do things for you), and events (code that is executed automatically when a certain situation - an event - occurs).

Once we have looked at what an object is, we'll then be able to look at how objects are useful to us in developing ASP applications. Developing web applications requires us to deal with both the client-side and server-side programming, and therefore we'll take a look at how objects can be used on both sides of the wire.

Notes

9.1 What is an Object?

In the real world, we already know what objects are. They're the things we look at, pick up, and use every day - things like our chairs, our telephones, and our computers. All these are solid, tangible entities.

However, if we want to describe a telephone to somebody in abstract terms, we can do this by talking about it in terms of its essential characteristics - what properties it has, what it can do, and how we can interact with it. All telephones have these characteristics, and we can use them to establish exactly how one telephone differs from the next.

So, for our telephone's physical properties, we could say that it has a microphone, a speaker, and a dialing pad. We could also say that it lets us do things, such as call someone and talk to them. Our telephone will also tell us when certain events are happening: for example, if a friend is trying to call you, your telephone will ring to let you know. This ringing will prompt you to take some action, like picking up the handset and talking to your friend. As an abstract object, our telephone has:

- Certain properties that define and describe it
- A set of things or methods that it lets us do
- The ability to prompt action when events occur.

We can use these three attributes to describe physical objects and abstract concepts. In a few minutes we will describe how these real-world ideas are replicated in software objects, but for now let's go a little deeper into our real-world telephone. By learning about what objects are, we can then look at how to use them in a way known as object-based programming. In the object-based way of programming, the application is broken down into a set of objects. In doing this, you can build the application in a two stage process. Firstly, you create the objects you will need in your application and then you set up the relationships and interactions between objects. Later in this chapter, we will see how the objects of Active Server Pages relate and interact with each other and allow us to build our applications.

Example:

Here is our telephone. To look at this as an object, let's put down some information about it. We will be classifying the information into three categories:

- Things that describe the telephone
- Things that we can do with the phone
- Things that the telephone tells us and that we can react to.

Notes

Let's look at each of these aspects in turn:

Describe the telephone	The telephone is gray The telephone is made of plastic The handset weighs 6.5 ounces The telephone has 12 keys The telephone number is (714) 555-1523 The telephone is connected to the exchange
What can we do with it?	We can place an outgoing call We can answer an incoming call We can hang up the current call We can enter our calling card number We can disconnect it from the exchange
What can it tell us that we can react to?	Someone is trying to call us The person we are calling is busy Another person is calling while we are talking

How It Works

The three categories that we have created in the left-hand column can be applied to any object. In fact, the best way to describe an object is to break down its characteristics into these three categories, and put information about your object into these categories. Any information that you have about a particular object can be included in one of these categories.

If you have another telephone that features all these characteristics, except that its color is blue, then we can describe your telephone by changing that one part of the description above. Moreover, this technique works for any type of object, both real world and software.

9.1.1 The Building Blocks of Objects (Properties, Methods, Instances of Objects)

Object Terms

So far, we have used verbose English terms to describe our three categories. In the world of objects, we need terms that concisely describe each of these three categories. These terms are properties, methods and events. In addition to these terms, we need to look at the term instance as it relates to objects. In this section, we'll look more carefully at what each of these means in abstract terms.

Instances and Classes

When we are talking about a unique object, we can use the term instance to say that we are talking about a particular telephone object - your telephone for example - that has a specific set of properties and values. When we want to talk about another telephone, we use a different instance of the telephone object. In this way, both you and we can have instances of the telephone object. For example, my telephone (my instance of a telephone object) is gray and comes with special answer-phone facilities, your telephone (your instance of a telephone object) may be red, blue, green, etc. These instances represent completely different physical objects. However, since they are both instances of the same object description or template, they share the same types of characteristics such as methods, properties (although the values can be different), and events. When a specific instance of an object is created from the template for the object, the object is said to have been instantiated. What actually happens is that a copy is made of all of the properties and events from the object description, but the methods (frequently a big chunk of code) remain in the original place and this section of code is used by all of the different instantiations of that one object.

Notes

So we've mentioned object descriptions or templates, but it's time to give them their proper name in ASP; classes. We mentioned that each object can have different instances. For instance, my telephone is small and white and has 12 buttons. Your telephone will probably be different to that, but they're both recognizable as telephones and they both provide the same function. They both conform to a set of rules for what a telephone does - they connect to the local telephone line, they both have a numeric keypad and somewhere to speak into. A class in ASP is like a set of design rules that an object must conform to. It would be no good if my telephone didn't have a handset, or a numeric keypad, even if it did plug into the telephone socket on the wall.

Caution In a class there should be a minimum set of functions that your object must be able to perform.

Properties

When talking about those characteristics that describe an object, we are talking about the properties of the object. Each property of the object describes a particular aspect of the object. The property is actually described as a name/value pair. This means that for every named property, there is a single unique value that describes that property for this instance of the object. If we go back to our telephone example, we can create a table that lists each of the property names and the value of each property.

Property Name	Property Value
Color	Grey
Material	Plastic
Weight	6.5 ounces
NumberOfKeys	12
TelephoneNumber	(714) 555-1523
Connected	Yes

We now have a set of properties that describe this instance. The properties of an object are used to represent a set of values associated with the object. A new instance of the object may have different property values, but it has the same property names.

Even with different property values, these two telephones are instances of the same object template. Since we know that all telephone objects have a 'Color' property, we can determine the color of each of the phones by examining its 'Color' property value. We can use properties in two ways. We can read from them or we can also write to them. So if we wanted, we could have changed the cover of our telephone to a different color, if we required.

Notes

Now that we have a way of describing the telephone object, let's take a look at what we can do with it.

Methods

Another characteristic of objects is that they can perform functions for us. For example, the Place Call method would perform several functions for you. It will connect you to the local exchange, the exchange will route your call, and then when it reaches the destination, and it will make the destination phone ring. These built-in actions occur whenever you pick up the handset and dial a number. This is a capability that has been built in to the machine.

However not all objects have functions like this. A chair object allows you to sit in it, so you could say that it is functioning to support your body. Objects that perform tasks that are more 'functional' are said to have methods. The tasks that an object can perform are called methods.

A method is defined as an action that an object can take. The code in a method is executed when the method is called. This calling is done by a command you write in the script of your ASP page. Once we have created an instance of an object, we can tell it to perform a certain task calling one of its methods.

Let's illustrate this using the telephone example. Our telephone object can carry out five methods. Each of these methods will cause the telephone object to perform an action. Here is a list of functions that the methods of the telephone object can perform:

These methods are used when we want our telephone object to perform a certain function; all we need to do is tell it to execute the corresponding method.

Methods are actually blocks of code that are written by the designer of the object (Microsoft, for example).

Did u know? What is the reason of method existing?

The reason methods exist is because lots of programmers want to do the same job, so it is worth it for the gurus at Microsoft to write the code to do that job, test it, optimize it, and get it in great shape, and then bundle it up in the object.

We, as programmers, can then use that code pre-made. Instead of re-inventing the wheel we spend our time on the unique parts of our project.

Self Assessment

Fill in the blanks:

1. When a specific instance of an object is created from the …….. for the object, the object is said to have been instantiated.
2. A new instance of the object may have different property values, but it has the same ……..
3. A method is defined as an action that an …….. can take.
4. The properties of an object are used to represent a set of …….. associated with the object.

Notes

9.2 ASP Server Object

The ASP Server object is used to access properties and methods on the server. Its properties and methods are described below:

Properties

Property	Description
Script Timeout	Sets or returns the maximum number of seconds a script can run before it is terminated

Methods

Method	Description
Create Object	Creates an instance of an object
Execute	Executes an ASP file from inside another ASP file
GetLastError()	Returns an ASPError object that describes the error condition that occurred
HTMLEncode	Applies HTML encoding to a specified string
Map Path	Maps a specified path to a physical path
Transfer	Sends (transfers) all the information created in one ASP file to a second ASP file
URLEncode	Applies URL encoding rules to a specified string

Example: When a file was last modified?

Checks when this file was last modified.

ASP CODE	OUTPUT

`<html>`
`<body>`
 	
`<%`
`Set fs = Server.CreateObject ("Scripting.FileSystemObject")`
`Set rs = fs.GetFile(Server.MapPath("demo_lastmodified.asp"))`
`modified = rs.DateLastModified`
`%>`
`This file was last modified on:`
`<%response.write(modified)`
`Set rs = Nothing`
`Set fs = Nothing`
`%>`
 	
`</body>`
`</html>`
	This file was last modified on: 18/05/2003 14:48:10

Notes

Open a text file for reading

This example opens the file "Textfile.txt" for reading.

ASP CODE	OUTPUT
<html> <body> <% Set FS = Server.CreateObject("Scripting.FileSystemObject") Set RS = FS.OpenTextFile(Server.MapPath("text") & "\TextFile.txt",1) While not rs.AtEndOfStream Response. Write RS.ReadLine Response. Write(" ") Wend %> <img border="0" src="/images/btn_view_text.gif"> </body> </html>	Hello World line 1 Hello World line 2 Hello World line 3 ■ View TextFile

Homemade hit counter

This example reads a number from a file, adds 1 to the number, and writes the number back to the file.

ASP CODE	OUTPUT
<% Set FS=Server.CreateObject("Scripting.FileSystemObject") Set RS=FS.OpenTextFile(Server.MapPath("counter.txt"), 1, False) fcount=RS.ReadLine RS.Close fcount=fcount+1 'This code is disabled due to the write access security on our server: 'Set RS=FS.OpenTextFile(Server.MapPath("counter.txt"), 2, False) 'RS.Write fcount 'RS.Close Set RS=Nothing Set FS=Nothing %> <html> <body> This page has been visited <%=fcount%> times. </body> </html>	This page has been visited 12345 times.

Notes

Self Assessment

Fill in the blanks:

5. …….. applies HTML encoding to a specified string.
6. …….. Sets or returns the maximum number of seconds a script can run before it is terminated.
7. …….. returns an ASPError object that describes the error condition that occurred.

9.3 ASP Session Object

The Session Object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application.

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

ASP solves this problem by creating a unique cookie for each user. The cookie is sent to the client and it contains information that identifies the user. This interface is called the Session object.

The Session object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application. Common information stored in session variables are name, id, and preferences. The server creates a new Session object for each new user, and destroys the Session object when the session expires.

Task Analyze the working of the session object.

When does a Session Start?

A session starts when:

- A new user requests an ASP file, and the Global.as a file includes a Session_OnStart procedure
- A value is stored in a Session variable
- A user requests an ASP file, and the Global.asa file uses the <object> tag to instantiate an object with session scope.

When does a Session End?

A session ends if a user has not requested or refreshed a page in the application for a specified period. By default, this is 20 minutes.

If you want to set a timeout interval that is shorter or longer than the default, you can set the Timeout property.

The example below sets a timeout interval of 5 minutes:

```
<%
Session.Timeout=5
%>
```

Notes

To end a session immediately, you may use the Abandon method:

```
<%
Session.Abandon
%>
```

Notes The main problem with sessions is WHEN they should end. We do not know if the user's last request was the final one or not. So we do not know how long we should keep the session "alive". Waiting too long for an idle session uses up resources on the server, but if the session is deleted too soon the user has to start all over again because the server has deleted all the information. Finding the right timeout interval can be difficult!

Tip: If you are using session variables, store SMALL amounts of data in them.

Store and Retrieve Session Variables

The most important thing about the Session object is that you can store variables in it.

The example below will set the Session variable *username* to "Donald Duck" and the Session variable *age* to "50":

```
<%
Session("username")="Donald Duck"
Session("age")=50
%>
```

When the value is stored in a session variable it can be reached from ANY page in the ASP application:

```
Welcome <%Response.Write(Session("username"))%>
```

The line above returns: "Welcome Donald Duck".

You can also store user preferences in the Session object, and then access that preference to choose what page to return to the user.

The example below specifies a text-only version of the page if the user has a low screen resolution:

```
<%If Session("screenres")="low" Then%>
  This is the text version of the page
<%Else%>
  This is the multimedia version of the page
<%End If%>
```

Remove Session Variables

The Contents collection contains all session variables.

It is possible to remove a session variable with the Remove method.

The example below removes the session variable "sale" if the value of the session variable "age" is lower than 18:

Notes

```
<%
If Session.Contents("age")<18 then
   Session.Contents.Remove("sale")
End If
%>
```

To remove all variables in a session, use the RemoveAll method:

```
<%
Session.Contents.RemoveAll()
%>
```

Loop Through the Contents Collection

The Contents collection contains all session variables. You can loop through the Contents collection, to see what's stored in it:

```
<%
Session("username")="Donald Duck"
Session("age")=50
dim i
For Each i in Session.Contents
   Response.Write(i & "<br />")
Next
%>
```

Result:

```
Username
Age
```

If you do not know the number of items in the Contents collection, you can use the Count property:

```
<%
dim i
dim j
j=Session.Contents.Count
Response.Write("Session variables: " & j)
For i=1 to j
   Response.Write(Session.Contents(i) & "<br />")
Next
%>
```

Result:

```
Session variables: 2
Donald Duck
50
```

Notes

Task Set the Session variable whose *username* is "Sonali Bose" and the Session variable *age* to "26"

Loop Through the StaticObjects Collection

You can loop through the StaticObjects collection, to see the values of all objects stored in the Session object:

```
<%
dim i
For Each i in Session.StaticObjects
   Response.Write(i & "<br />")
Next
%>
```

Pitfalls of Session Variables

- Session variables and cookies are synonymous. So if a user has set his browser not to accept any cookies, your Session variables won't work for that particular web surfer.
- An instance of each session variable is created when a user visits the page, and these variables persist for 20 minutes AFTER the user leaves the page! (Actually, these variables persist until they "timeout". This timeout length is set by the web server administrator. I have seen sites that the variables will collapse in as little as 3 minutes, and others that persist for 10, and still others that persist for the default 20 minutes.) So, if you put any large objects in the Session (such as ADO recordsets, connections, etc.), you are asking for serious trouble! As the number of visitors increase, your server will experience dramatic performance woes by placing large objects in the Session.
- Since Session variables can be created on the fly, used whenever, and do not require the developer to dispose of them explicitly, the overuse of Session variables can lead to very unreadable and unmaintainable code.
- Session variables take you one step closer to VB programming in the sense that you can grab one without initializing the variable, use it whenever you want to, and not have to worry about releasing it when you've finished using it. And WHO wants to go there? Not me.

Session Variable Cookies

Kind of. Session variables are bits of information that can be stored on a user-by-user basis. These bits of information are stored on the Web server. To tie these bits of memory with a particular user, for session variables to persist across Web pages, the user must have cookies enabled. When a user first visits a page on a site that uses Session variables, two things happen:

— A block of memory is allocated for that particular user's session variables on the Web server. This block of memory is identified by a unique SessionID — A cookie is written to the user's computer, storing the value of the SessionID

When the user visits another page on the site that uses session variables, the users SessionID cookie is referenced to determine if the user already has a session variable memory block setup. If he does, the values that are stored there are referenced. Through this mechanism, one can create seemingly "global" variables that persist from one page to another.

Notes

But what if the user doesn't have cookies enabled? Session variables can still be used, but there will be no way for the user to store the SessionID as he jumps from page to page. Hence, the session variables will not be saved for the user who has cookies disabled. However, session variables that are created and referenced on the same page will work fine. For example, the following script will output "Hello, World" regardless of whether or not the user has cookies enabled:

```
<%
  Session("Message") = "Hello, World!"
  Response.Write Session("Message")
%>
```

Notes Of course, if, on another page, the value of Session("Message") is referenced, only those users who have cookies enabled will have a value of "Hello, World!" there.

Self Assessment

Fill in the blanks:

8. The Session object is used to store information about, or change settings for a .....
9. The server creates a new Session object for each new user, and destroys the ..... when the session expires.
10. Kind of Session variables are bits of information that can be stored on a ..... basis.

9.4 ASP Application Object

A group of ASP files that work together to perform some purpose is called an application. The Application object in ASP is used to tie these files together.

An application on the Web may be a group of ASP files. The ASP files work together to perform some purpose. The Application object in ASP is used to tie these files together.

The Application object is used to store and access variables from any page, just like the Session object. The difference is that ALL users share one Application object, while with Sessions there is one Session object for EACH user.

The Application object should hold information that will be used by many pages in the application (like database connection information). This means that you can access the information from any page. It also means that you can change the information in one place and the changes will automatically be reflected on all pages.

Notes You can store values in the Application Collections. Information stored in the Application collections is available throughout the application and has application scope.

Store and Retrieve Application Variables

Application variables can be accessed and changed by any page in the application.

Notes

You can create Application variables in "Global.asa" like this:

```
<script language="vbscript" runat="server">
Sub Application_OnStart
application("vartime")=""
application("users")=1
End Sub
</script>
```

In the example above we have created two Application variables: "vartime" and "users".

You can access the value of an Application variable like this:

```
There are
<%
Response.Write(Application("users"))
%>
active connections.
```

Loop Through the Contents Collection

The Contents collection contains all application variables. You can loop through the Contents collection, to see what's stored in it:

```
<%
dim i
For Each i in Application.Contents
   Response.Write(i & "<br />")
Next
%>
```

If you do not know the number of items in the Contents collection, you can use the Count property:

```
<%
dim i
dim j
j=Application.Contents.Count
For i=1 to j
   Response.Write(Application.Contents(i) & "<br />")
Next
%>
```

Loop Through the StaticObjects Collection

You can loop through the StaticObjects collection, to see the values of all objects stored in the Application object:

Notes

```
<%
dim i
For Each i in Application.StaticObjects
   Response.Write(i & "<br />")
Next
%>
```

Did u know? What is Staicobject collection?

The StaticObjects collection contains all of the objects created by using the <OBJECT> tags within the scope of the Application object. You can use the collection to determine the value of a specific property for an object or to iterate through the collection and retrieve all properties for all static objects.

Lock and Unlock

You can lock an application with the "Lock" method. When an application is locked, the users cannot change the Application variables (other than the one currently accessing it). You can unlock an application with the "Unlock" method. This method removes the lock from the Application variable:

```
<%
Application.Lock
   'do some application object operations
Application.Unlock
%>
```

Did u know? What is lock and unlock method?

The Lock method blocks other clients from modifying the variables stored in the Application object, ensuring that only one client at a time can alter or access the Application variables. If you do not call the Application.Unlock method explicitly, the server unlocks the locked Application object when the .asp file ends or times out.

Pitfalls of Application Variables

Because only one instance of the Application object exists for the entire Web application, application variables can be used more liberally than session variables. However, the Application object does take up memory on the Web server, so only items that need to be stored in application scope should be entered into the Application object. Two common pitfalls should be avoided when working with application variables:

- Pitfall 1 - Do not put objects into the Application object unless vitally needed.
- Pitfall 2 - Only create application variables that are necessary. Why create unneeded application variables when they'll only waste your Web server's memory?

A common pitfall among developers is wanting to place objects in the Application. One object that is particularly alluring to put into the Application is the ADO Connection object, which is used to connect to a database. We'll discuss this object in detail during Week 3. It may seem like

Notes

a good idea to create a single database connection object and have all users communicate to a database through that object. However, as with most other objects, it is always best to wait to create the object until you need it.

Task The ADO Connection object will degrade your server's performance if put into the Application object. Analyze

Like the Session object, the Application object is easy to use, and the temptation to create a plethora of application variables is high indeed. Many developers use the Application object to store many Web site-wide constants - for example, a navigational footer common to all Web pages, or perhaps the Webmaster's email address. Although it's better to place these items in the Application object than in the Session object, they belong best in a static text file on the Web server. This text file can then be included into any ASP page that needs to display the navigation footer or the Webmaster's email address. We will discuss how to include files on Day 13.

Sometimes, however, the use of application variables is preferred to include files. If the data you need to store changes often, such as the last post to a message board, then the information should be stored in the Application object. Include files should only be used if the data is static and not susceptible to frequent change.

Although you can afford to be less prudent when creating application variables than you can be when creating session variables, you should still strive to use the minimum needed amount of such variables. Because the Application object persists in the Web server's memory, the fewer the application variables stored within it, the less drain on the Web server's performance. Therefore, the Application should remain free of objects and contain only needed application variables.

Initializing Application and Session Variables

Recall that the Session and Application objects serve as warehouses for session and application variables, respectively. When the user first visits the site, a new Session warehouse is created. What is inside this warehouse by default? Initially, the warehouse is empty. Attempting to retrieve a nonexistent session variable from the users Session returns an empty string.

However, the Session and Application objects both have an event you can use to initialize the contents of your users Session and your Website's Application. This event is called the OnStart event, and it occurs at different times for the Session and Application objects. The Session's OnStart event occurs whenever a new user comes to the site. This is when a new Session object instance is created for this particular user. In the OnStart event, you can create the session variables you plan to use and initialize them to certain values. The Application's OnStart event fires when the first Session object is instantiated - that is, when the first Session's OnStart event fires.

To create and initialize session and application variables, you need to write event handlers for the OnStart events. Listing 11.13 displays the OnStart event handler for both the Session and Application objects.

Listing - Initializing Application and Session Variables in the OnStart Events

```
1:   Sub Application_OnStart()

2:   Application("LastPost") = ""

3:   End Sub
```

Notes

```
4:

5:   Sub  Session_OnStart()

6:   Session("LogonTime")  =  Now()

7:   Session("Name")  =  ""

8:

9:   Dim  aPi(2)

10:  aPi(0)  =  3

11:  aPi(1)  =  1

12:  aPi(2)  =  4

13:  Session("Pi")  =  aPi

14:  End  Sub
```

Listing above displays the OnStart event handlers for the Session and Application objects. The Application object's OnStart event handler begins on line 1. A single application variable, LastPost, is initialized to a blank string (line 2). The Session object's OnStart event handler begins on line 5. Line 6 creates a session variable named LogonTime that is initialized to the Web server's current date and time. Next, a session variable, Name, is created and initialized to an empty string (line 7). Line 9 creates aPi, an array, and lines 10 through 13, set the three elements of aPi. Line 13 creates the final session variable, setting it equal to aPi.

Notes Note the syntax for creating an event handler (lines 1 and 5). First, a subroutine is created and named with the object's name, followed by an underscore, and then followed by the event name. Because the OnStart event passes in no parameters, no arguments are in the event handler definition.

Whenever these events fire, the functions in above Listing are executed. The OnStart event for the Session object occurs whenever a new visitor (one that does not already have a Session instance) arrives at the site. If your Web site experiences a lot of traffic, this event can fire hundreds, or even thousands of times an hour. The Application's OnStart event, however, fires only once, right before the first Session is created. If you restart your Web server, though, the next visit after restarting causes the Application object's OnStart event to fire.

Self Assessment

Fill in the blanks:

11. An application on the Web may be a group of.
12. An ASP-based application is defined as all the .asp files in a directory and its subdirectories.
13. …… contains all of the objects added to the session with the <OBJECT> tag.

9.5 ASP the Global.asa file

The Global.as a file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application.

Notes

The Global.asa file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application. All valid browser scripts (JavaScript, VBScript, JScript, PerlScript, etc.) can be used within Global.asa.

The Global.asa file can contain only the following:

- Application events
- Session events
- <object> declarations
- TypeLibrary declarations
- the #include directive

Notes The Global.asa file must be stored in the root directory of the ASP application, and each application can only have one Global.asa file.

Events in Global.asa

In Global.asa you can tell the application and session objects what to do when the application/session starts and what to do when the application/session ends. The code for this is placed in event handlers. The Global.asa file can contain four types of events:

Application_OnStart - This event occurs when the FIRST user calls the first page from an ASP application. This event occurs after the Web server is restarted or after the Global.asa file is edited. The "Session_OnStart" event occurs immediately after this event.

Session_OnStart - This event occurs EVERY time a NEW user requests his or her first page in the ASP application.

Session_OnEnd - This event occurs EVERY time a user ends a session. A user ends a session after a page has not been requested by the user for a specified time (by default this is 20 minutes).

Application_OnEnd - This event occurs after the LAST user has ended the session. Typically, this event occurs when a Web server stops. This procedure is used to clean up settings after the Application stops, like delete records or write information to text files.

A Global.asa file could look something like this:

```
<script language="vbscript" runat="server">
sub Application_OnStart
'''some code
end sub
sub Application_OnEnd
'''some code
end sub
sub Session_OnStart
'''some code
end sub
sub Session_OnEnd
'''some code
```

Notes

```
end sub
</script>
```

Notes We cannot use the ASP script delimiters (<% and %>) to insert scripts in the Global.asa file, we will have to put the subroutines inside the HTML <script> tag.

9.6 # include

The #include directive tells the preprocessor to treat the contents of a specified file as if those contents had appeared in the source program at the point where the directive appears. You can organize constant and macro definitions into include files and then use #include directives to add these definitions to any source file. Include files are also useful for incorporating declarations of external variables and complex data types. You need to define and name the types only once in an include file created for that purpose.

```
#include   "path-spec"
#include   <path-spec>
```

The path-spec is a filename optionally preceded by a directory specification. The filename must name an existing file. The syntax of the path-spec depends on the operating system on which the program is compiled.

Both syntax forms cause replacement of that directive by the entire contents of the specified include file. The difference between the two forms is the order in which the preprocessor searches for header files when the path is incompletely specified.

Syntax Form	Action
Quoted form	This form instructs the preprocessor to look for include files in the same directory of the file that contains the#include statement, and then in the directories of any files that include (#include) that file. The preprocessor then searches along the path specified by the /I compiler option, then along paths specified by the INCLUDE environment variable.
Angle-bracket form	This form instructs the preprocessor to search for include files first along the path specified by the /I compiler option, then, when compiling from the command line, along the path specified by the INCLUDE environment variable.

If the filename enclosed in double quotation marks is an incomplete path specification, the preprocessor first searches the "parent" file's directory. A parent file is the file containing the #include directive. For example, if you include a file named file2 within a file named file1, file1 is the parent file.

Include files can be "nested"; that is, an #include directive can appear in a file named by another #include directive.

Example: File2, above, could includefile3. In this case, file1 would still be the parent of file2 but would be the "grandparent" of file3.

When include files are nested and when compiling from the command line, directory searching begins with the directories of the parent file and then proceeds through the directories of any

Notes

grandparent files. Thus, searching begins relative to the directory containing the source currently being processed. If the file is not found, the search moves to directories specified by the /I compiler option. Finally, the directories specified by the INCLUDE environment variable are searched.

From within the development environment, the INCLUDE environment variable is ignored. To set the directories searched for include files (this information also applies to the LIB environment variable.), see VC++ Directories, Projects, Options Dialog Box.

The following example shows file inclusion using angle brackets:

```
#include <stdio.h>
```

This example adds the contents of the file named STDIO.H to the source program. The angle brackets cause the preprocessor to search the directories specified by the INCLUDE environment variable for STDIO.H, after searching directories specified by the /I compiler option.

Example: The following example shows file inclusion using the quoted form:

```
#include "defs.h"
```

This example adds the contents of the file specified by DEFS.H to the source program. The double quotation marks mean that the preprocessor searches the directory containing the parent source file first.

Nesting of include files can continue up to 10 levels. Once the nested #include is processed, the preprocessor continues to insert the enclosing include file into the original source file.

Self Assessment

Fill in the blanks:

14. The Global.asa file must be stored in the root directory of the …… application.
15. Include files can be "…… "; that is, an #include directive can appear in a file named by another #include directive.

Caselet

Web Hacking Contest puts Firms on Guard

INTERNET managers of establishments worldwide are likely to work overtime on Sunday, the date of a world hackers' contest which starts at 9 a.m. and ends at 3 p.m. (the time zone to be followed from a site zone-h.org, say the contest instructions).

Establishments have been advised to deactivate unneeded public facing Web servers and turn off servers that are not urgently required.

The challenge could even be a hoax or just a hyped up minor event, said security experts. But no one is taking any chances - the US department of homeland security has issued warnings, so has the Ministry for Information and Communication in Korea which feels Korean Web servers could be specially vulnerable to the threat. So have European Internet security experts.

Security companies came to know of the contest through noticing that there was increased monitoring of Websites and also from tracking Internet relay chats.

Contd...

Notes

Yet another suspicion is that the timing given might just be to divert attention and that the actual hacking might take place some time before or after the so-called scheduled time when companies might be less on their guard.

The contest, referred to as 'Defacers Challenge' has been put up on the Web site www.defacers-challenge.com, by "Eleonara (67)" who claims the "help of some groups from underground"

Everyone who can notify a defacement is entitled to participate, says the challenge. It will check zone-h.org for statistics, "due to its notorious independency". (This site tracks daily hackings server and operation-system-wise).

The contest targets 6,000 sites worldwide, participants have to penetrate as many Web servers as possible and deface the Web sites within the six hours given to them.

The contest follows a point-system, awarding one point for a Windows server, 5 for an HP-UX or Macintosh server. The winners have been promised free Web space, hosting services, a free domain name, and the like.

Captain Raghu Raman, Global Practice Head, Mahindra Consulting Special Services Group (information security company), says the context seems to be more of a hoax since so many Web sites cannot be hacked in the space of just six hours. "It is extremely impossible since most commercial Web sites are hosted on single servers and have mirrored sites which take over in the case of a single site faces any problem."

He feels that the interest generated in the contest could attract many "kiddish" amateurs whose simultaneous activity during those hours could create major bandwidth issues.

But he too advises precautionary actions. "Companies should look to strengthen their systems and guard against any vulnerabilities. They should lock down their servers and public domains, create awareness amongst internal security teams and remember that hackers do not have to wait for the specified date to start. They can commence anytime."

The most reliable indicator of whether an organisation's Web site will be defaced or otherwise compromised is if the organisation's Web server is not appropriately secured, or if it exhibits known vulnerabilities, which can be exploited, he says.

According to him, the following technical guidelines may be followed; making sure that default passwords are changed. This should include Web servers and any other servers that the Web server has a trusted relationship with; removing sample applications that are not being used, such as CGI scripts and Active Server Pages, from Web servers; locking down Microsoft Front Page Extensions (by default, those extensions are installed in a manner that gives every user the ability to author Web pages, even through proxy servers).

The other measures recommended are turning the Web server logging on. Logs are essential to determining how a defacement was accomplished so a recurrence can be prevented. Use of the extended log format is recommended; and having a current backup of one's Web server. In the event of a defacement, a good backup is essential to quickly restore the server to its original look.

Lastly, he recommends application of the latest security patches to one's Web server and the underlying operating system after appropriate testing.

Over the last six months, he says, nearly 571 Indian sites have been hacked. This is twice the number of sites hacked in 2002. Most of the hackings have been claimed by Pakistani hacker groups.

Notes

9.7 Summary

- An object is a software representation of a real-world item, or object. Software objects feature properties, methods, and events.
- Each property of the object describes a particular aspect of the object. The property is actually described as a name/value pair. This means that for every named property, there is a single unique value that describes that property for this instance of the object.
- By an instance of a object we are about a particular telephone object that has a specific set of properties and values The instances of an object represent completely different physical objects. However they share the same types of characteristics such as methods, properties and events. When a specific instance of an object is created from the template for the object, the object is said to have been instantiated.
- Once we have created an instance of an object, we can tell it to perform a certain task calling one of its methods.
- A class in ASP is like a set of design rules that an object must conform to. In a class there should be a minimum set of functions that your object must be able to perform.
- A method is defined as an action that an object can take. The code in a method is executed when the method is called. This calling is done by a command you write in the script of your ASP page.
- Methods are actually blocks of code that are written by the designer of the object. It relieves the programmers of writing the same code again and again as they come bundled with the object.
- Some of the methods can be executed directly, while others need additional information. A method can have none, one, or more than one parameter. Information passed as parameters of the method for execution by the method, will only be executed if all parameters have been supplied.
- Method can also return information to us. The value returned by a method is called a return value. If a method has a return value, then it will pass information back to us. This information could have a number of purposes. As the user of an object, we can decide whether we want to do anything with the return value. If the information is pertinent to what we are doing, then we can capture the return value and do something with it later. If we do not care what the return value is, we can just ignore it and continue with our work.
- An object generates an event whenever something of interest happens. Just like methods, events can have parameters. These parameters can hold specific information about the event. When an object generates an event, the object can be said to fire the event. When the object has fired the event, we say that the user must handle the event.
- The inbuilt ASP objects include Application, ASPError, ObjectContext, Response, Request, Server and Session.
- An application on the Web may be a group of ASP files. The ASP files work together to perform some purpose. The Application object in ASP is used to tie these files together.
- The Application object is used to store and access variables from any page, just like the Session object. The difference is that ALL users share one Application object, while with Sessions there is one Session object for EACH user.
- The Application object should hold information that will be used by many pages in the application (like database connection information). This means that you can access the information from any page. It also means that you can change the information in one place and the changes will automatically be reflected on all pages.

Notes

- The Session object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application.
- When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the Internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state. ASP solves this problem by creating a unique cookie for each user. The cookie is sent to the client and it contains information that identifies the user. This interface is called the Session object.
- The Session object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application. Common information stored in session variables are name, id, and preferences. The server creates a new Session object for each new user, and destroys the Session object when the session expires.
- The ASP Server object is used to access properties and methods on the server.
- The ASP Error object is used to display detailed information of any error that occurs in scripts in an ASP page. The ASP Error object is implemented in ASP 3.0 and it is only available in IIS5. The ASP Error object is created when Server.GetLastError is called, so the error information can only be accessed by using the Server.GetLastError method.

9.8 Keywords

Application Object: Stores information (variables and objects) needed for all users of a particular application. Information stored in the Application object persists for the lifetime of the application.

Class: A class in ASP is like a set of design rules that an object must conform to. In a class there should be a minimum set of functions that your object must be able to perform.

Event: Object generates an event whenever something of interest happens. When an object generates an event, the object can be said to fire the event. When the object has fired the event, we say that the user must handle the event.

Instance of Object: By an instance of a object we mean a particular object that has a specific set of properties and values The instances of an object represent completely different physical objects but sharing the same types of characteristics such as methods, properties and events.

Method: A method is defined as an action that an object can take. The code in a method is executed when the method is called. This calling is done by a command you write in the script of your ASP page.

Object: An object is a software representation of a real-world item, or object. Software objects feature properties, methods, and events.

Property of Object: Property of the object describes a particular aspect of the object. The property is actually described as a name/value pair.

Server Object: Provides access to methods and properties on the server. These methods and properties typically serve as utility functions.

Session Object: Stores information (variables and objects) needed for a particular user session. Information stored in the Session object is not discarded when the user jumps between pages in the application, but instead persists for the entire user session.

Notes

9.9 Review Questions

1. The information in ASP built-in objects can also be obtained in a COM component or an ISAPI application. Explain
2. What three things make up an object? What built-in object would you use to read a client's cookies?
3. Suppose that you want an object to represent a desk lamp. What properties and methods would you need? Assume that the lamp has three settings: off, dim, and full power. Design the object. (No coding required)
4. Suppose that you want an object to represent a microwave. What properties, methods, and events would you need? It can have variable power and time settings and should beep when the food is done. Design the microwave. (No coding required).
5. Examine that why a block of memory is allocated for that particular user's session variables on the Web server?
6. The preprocessor stops searching as soon as it finds a file with the given name. Analyze
7. Which built-in ASP object would you use to check if the user is still connected to the server? Explain with program
8. The ASP built-in objects are organized by the type of information they contain. What is meant by this statement?
9. Briefly make the distinction between Application object and Session Object of ASP.
10. Why a common pitfall among developers wants to place objects in the Application? Give reasons.

Answers: Self Assessment

1.	template	2.	property names
3.	object	4.	values
5.	HTMLEncode	6.	Script Timeout
7.	GetLastError()	8.	user session
9.	Session object	10.	user-by-user
11.	ASP files	12.	Virtual
13.	Application. Static Objects Collection	14.	ASP
15.	nested		

9.10 Further Readings

Books

A Keyton Weissinger, *ASP in a Nutshell, 2nd Edition* [ILLUSTRATED] (Paperback), O'Reilly Media; 2 edition (January 1, 2000)

Bob Reselman, *Active Server Pages 3.0 by Example,* Que Publishing

David Buser, Chris Ullman, Brian Francis, Dave Sussman, *Beginning Active Server Pages 3.0,* John Wiley & Sons

Notes

John W. Gosney, *ASP Programming for the Absolute Beginner* (For the Absolute Beginner) (Paperback), Course Technology PTR; 1 edition (July 15, 2002)

Keith Morneau, Jill Batistick, *Active server pages,* Course Technology

Scott Mitchell, *Designing Active Server Pages,* O'Reilly Media

Online links

http://msdn.microsoft.com/en-us/library/ms525316(v=vs.90).aspx

http://www.w3schools.com/asp/default.asp

Notes

Unit 10: Working with Response Object

CONTENTS

Objectives

Introduction

10.1 Response Object

10.2 Response.Write

10.3 Response.Buffer

10.4 Cookies

10.5 Summary

10.6 Keywords

10.7 Review Questions

10.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Recognize Response object
- Describe Response.Write
- Demonstrate the Response.Buffer
- Explain Cookies

Introduction

This unit will familiarize you with the one of the built in ASP objects, the Response object used for sending output which might be text, cookie data etc to the user from the Server. It has several methods and properties. Through its methods and properties you will learn how the output to be sent to the user can be controlled. Using the cache control property you can instruct the proxy whether to cache or not to cache.

10.1 Response Object

The response object is one of the five built-in ASP objects. Response is used to send output to the client. This output might be text displayed in a browser window, cookie data or it might have to do with how your pages are sent to the client and stored. For example, users might be at a form-based page on our site that is a quiz. Users select the answer to the question from a list of possible answers that are in an Option drop-down list. The name of the Option list is "answer". Once visitors select an answer, they hit the SUBMIT button. This action sends the answer that they have selected to our .asp page, which will respond to their choice with this code:

```
If Request. Form ("answer") = "42" then
      Response. Write "you got the correct answers!"
```

Notes

```
Else
    Response. Write "WRONG! Try again…"
End if
```

Hence, we are displaying dynamic content to visitors based on their choice. We can also set other browser properties that deal with how to display the response.

Dissecting the Response Object

(Sending HTML to the Browser, Buffering ASP Pages, Sending the User to another page, Cookies, Caching your ASP Pages)

Response allows you to send information to the browser and control how information is sent to the browser. This has several methods and properties.

Sending HTML to the Browser

The most common use of the Response object is to send data to the client's Web browser to be displayed as part of a Web page. It is done in two ways. The first is to use the Write method; the other is to use the shortcut

```
<%=_%>.
```

Task The ASP Response object is used to send output to the user from the server. Analyze

Self Assessment

Fill in the blanks:

1. The Response object methods give you potent control over what you can send to the ……. from the web server.
2. ……. sets the HTML header name to value.
3. ……. allows you to send information to the browser and control how information is sent to the browser.
4. The most common use of the Response object is to send data to the client's Web browser to be displayed as part of a …….

10.2 Response.Write

Without Response.Write, ASP is almost useless. The most important thing in using Response. Write is that the string being written cannot contain "%>". If you need to write a string that contains "%>", use "%\>". Since we use %> to indicate the end of a block of ASP code, putting it in your strings will confuse the system.

Like:

Response.Write ("<HR WIDTH=50%>") will result in an error because the server will interpret the "%>" as the closer to a block of ASP code.

```
Response.Write ("<HR WIDTH=50%\>") will write "<HR WIDTH=50%>" to the HTTP.
```

Notes

As the double quote (") is used to indicate the beginning and end of a string, writing strings that contain double quotes can cause problems. So, one way is to use the single quote in your string. At times, though you need to use double quotes. Putting two double quotes together will have the effect of including a double quote within your string. For example, if you write

```
Response.Write ("<IMG SRC= ""banner.gif"" >")
```

The browser will receive `<IMG SRC= "banner.gif">`.

Server.HTML Encode can be used to send text encoded so that the browser will not interpret it as HTML. For example,

```
Response.Write (Server.HTML Encode("<P align=right>")) will cause
```

`<P align=right>` to be printed to the screen without being interpreted as HTML.

Here is a list demonstrating the different ways of sending text containing quotation marks.

Notes Note that also demonstrates the use of Server.HTML Encode to send text so that the browser does not interpret it.

List 1

Special characteristics with Response.Write

```
1:  <%@Language=VBScript %>
2:  <% Option Explicit %>
3:  <HTML>
4:  <BODY>
5:  <A HREF= "index.html">click here</a>
6:  <BR>
7:  <%
8:  Response.Write("< A HREF= ""index.html"" > click here</a>")
9:  Response.Write("<BR>")
10: Response.Write("<A HREF= 'index.html'>click here</a>")
11: Response.Write("<BR>")
12: Response.Write(Server.HTML Encode ("<BR>"))
13: %>
14: </BODY>
15: </HTML>
```

Line 5 writes an HTML link statement normally. Line 8 writes the link statement using Response.Write and double quotes. Line 10 writes the same statement again, but using single quotes. Finally, line 12 demonstrates Server.HTML Encode.

USING<%=

```
<%=expression %> is equivalent to <% Response.Write (expression) %>.
```

This shortcut can be used only to send a single expression.

Notes

To use the shortcut in the middle of a long block of ASP code would require that you first add a closing %>, and then reopen with a <%=; close again with a %> and finally reopen with a <%.

Caution So many openings and closings of ASP code sections hurts the readability of your code, besides, it may hurt the performance as well.

Therefore, it is a good practice to go ahead with the Response.Write.

Sometimes, when the script looks like this:

```
1:  <%@Language=VBScript %>
2:  <HTML>
3:  <BODY>
4:  Hello. My name is Joe. This is my page.
5:  <P>
6:  today is <%=date %>
7:  </BODY>
8:  </HTML>
```

If you had a long block of HTML where you needed a single line of output from the ASP, it would probably be better to use the shortcut as is used in line 6; it is quite readable.

Using Response.Write in this case:

```
1:  <%@ Language=VBScript %>
2:  <HTML>
3:  <BODY>
4:  Hello. My name is Joe. This is my page.
5:  <P>
6:  today is
7:  <%
8:  Response.Write (Date)
9:  %>
10: </BODY>
11: </HTML>
```

Response.Write is used in line 8. It is a little less readable.

Buffering ASP Pages

Besides sending output to the client, Response can control how and when the output is sent to the client. Output can be sent in two different ways: buffered or unbuffered.

Unbuffered output is sent immediately.

Buffered output is not sent until the script is finished, or until a special command is given to send it. For example, let's say that you have a script with one Response.Write at the top and another

Notes

one later. Without buffering, the first one is sent to the client immediately before the latter Response.Write is executed. With buffering, though, all the output is collected in a buffer on the server and sent at once.

Did u know? How Buffering Works in ASP?

Buffering controls how content is written out from ASP to the end user. It affects how response.redirect works as well as the speed of your webpage!

Self Assessment

Fill in the blanks:

5. Response.Write writes a variable or text to the current …….. output as a string.
6. …….. output is not sent until the script is finished, or until a special command is given to send it.
7. The use of Server. HTML Encode to send text so that the …….. does not interpret it.
8. As the double quote (") is used to indicate the beginning and end of a …….., writing strings that contain double quotes can cause problems.
9. The most important thing in using …….. is that the string being written cannot contain "%>".

10.3 Response.Buffer

The buffer property is a Boolean property that determines whether the output of your ASP is sent as it runs or it is stored until all the code is complete or the Flush method is called. It is a Boolean property which means that it is set to true or false. Let's take a look how a couple of code blocks would differ based on the setting of the buffer property. The first code block is buffered.

```
<%
Option explicit
Response.Buffer =true
Response.Write "Running query.."
Dim conn
Dim RSTotal Sales
Set conn=server.createobject ("adodb.connnection")
Conn.open "sales", "sa", "your password"
Set RSTotalSales=conn.Execute (Select Sum (total amount) as Total Sales"_
& "from sales")
Response.Write RSTotalSales("TotalAmount")
%>
```

The scenario of this code block is that we are presenting a page that shows the total amount of sales for all records in a database table. We will estimate that the query will take 20s to run. The buffer is on. So when the code gets to this line:

Response.Write "running query.."

Notes

The text is placed in the buffer and is not sent to the browser. The code continues and 20s later the code completes. The text running query..as well as the result of the query are now sent to the browser. Now look at the flow of this code block:

```
<%
Option explicit
Response.Buffer=false
Response.Write "running query.."
Dim conn
Dim RSTotalSales
Set conn=server.createobject ("adodb.connection")
Conn.open "sales", "sa", "your password"
Set RSTotalSales=conn.Execute ("select Sum (TotalAmount) as_ TotalSales" _
&"from sales"
Response.Write RSTotalSales ("TotalAmount")
%>
```

This time when the code gets here:

Response.Write "running query.."

The text is immediately sent to the browser. So now visitors receive the feedback that something is happening while they are waiting for the query to run. This technique of giving the visitor gradual feedback is very important for procedures that take more than a few seconds to complete. Without the feedback, especially on the internet, visitors may assume something is wrong and leave your site.

Response.Clear

Suppose you have buffering turned on. As your script is executing, output is being sent to the buffer. Calling Response.Clear causes that buffer to be wiped out. In case, you have a page that you do not want to be viewed under a special set of circumstances, it can be used.

Response.Flush

Like Response.Clear, Response.Flush flushes all the data from the system buffer. However, Response.Flush first sends it to the client. This is useful Response.Clear, Response.Flush produces an error message when buffering is turned off.

Response.End

Response.End ends execution of the script. If buffering is turned on and there is any buffered data, it is sent. Any statements after the Response.End are not carried out. This is a more abrupt end than simply allowing the script to end on its own, and so should be avoided if possible. It can be useful, though, when problems (such as bad data) are detected to prevent them from making things worse.

Task Give the answers to following questions:

1. Explain the concept of sending output buffered or unbuffererd.
2. Explain how the Response.Buffer property can be used to specify whether the current page output should be buffered or not.

Notes

Sending the User to Another Page

There are HTML Web pages that are on the screen for just a few seconds and then suddenly it takes you to another page. This can be done in many ways.

One way uses the META tag:

```
<META HTTP-EQUIV=REFRESH CONTENT=_ "2;URL=http://www.macmillan.com">
```

this causes the browser to be sent to www.macmillan.com after 2 seconds.

It can also be done with client-side scripting like JavaScript, using the window object like this:

```
<script language= "JavaScript">
<!—
Window .location= 'anotherpage.html';
//— >
</script>
```

the response object offers another way i.e.

Response.Redirect

Response.Redirect URL takes the user to the page URL. If it is within the same site, a relative URL (such as "products/index.html") will work. If it is a separate site, the full address including http:// should be provided.

Notes Response.Redirect =URL is equivalent to Response.Redirect URL.

Self Assessment

Fill in the blanks:

10. …….. sends a redirect message to the browser, causing it to attempt to connect to a different URL.
11. …….. sends buffered output immediately.
12. …….. stops processing the .asp file and returns the current result.
13. …….. erases any buffered HTML output.

10.4 Cookies

Cookies are a way for you to store nuggets of information on the visitor's computer. You can then use your code to retrieve the values stored on the visitor's system at a later time.

Example: You could store the most recent searches a visitor has made on your website in a cookie. Then, when visitors search again, we could present them with their last five search criteria, or you could store visitor's location in a cookie and when they return to your site, you could display local information for your visitors.

But since the data are on the visitor's system, you cannot use cookies as your only way to identify visitors or in situations where the cookie must be present. You should always try to provide an alternative to cookies.

Notes

You place the cookies on the visitor's machine by using the cookies collection of the Response object. To write a single simple cookie to the visitor's system you would code this:

```
Response.Cookies ("NameOfCookies") = "value"
```

The name of cookie is the name you want to store the cookie as on the visitor's system. The value represents the text to store in the cookie. So if you coded this:

```
Response.Cookies ("Search criteria 1") = "ASP"
```

This could add the cookie with the name Search Criteria 1to the visitor's system. The cookie would contain the value ASP. If the cookie already exists, the old value will be overwritten.

Response Object Collections

Object	Purpose
Cookies	Allows you to write cookies to the visitor's browser.

In our search criteria example, we could store the five most recent searches in a complex cookie. The recent searches field is blank. But after the visitor has made a few searches, the select element would be populated with those searches.

Task Mention the two properties of Response that will determine how long an ASP page will be cached.

Self Assessment

Fill in the blanks:

14. Cookies are a way for you to store nuggets of information on the …….. computer.
15. Use your …….. to retrieve the values stored on the visitor's system at a later time.

Caselet

The Importance of Data Warehousing

If you were ever curious about datamining/datawarehousing, you'd have heard this story about "weekend daddies". In the US, a shopping chain discovered that adult males who had recently become daddies, sauntered into the mall, picked up beer for themsel ves for the weekend, and napkins for their kids' use during the week. Datawarehousing technology helped in the discovery. Once this was learnt, the mall placed the nappies' and beer stands next to each other and saw sales soar.

And, if you ever get to run a bank, datawarehousing technology will have told you that more often than nought, customers that contribute the most to your profits are also the ones who default on cheques frequently.

Warehousing and mining of data are hot issues in the backrooms of Net enterprises and traditional companies wanting to take advantage of the Net. The two concepts are simple and closely related.

Data warehousing is a way of storing content in databases to enable data mining. Data mining is the technique that reads the data warehouses and makes the tonnes of content on

Contd...

Notes

the databases intelligent to decision makers in an organisation. The result of a successful implementation of the two can be a report in form of charts and graphs on the effectiveness of the latest marketing campaign available online to top management.

While the scope of data mining and warehousing is vast, this article will be restricted to the importance of data warehousing and the subsequent ones will deal with techniques to using the technology and products available in the market.

In the old economy, data would have been stored at best in a tabular form to be accessed by the a few persons in the organisation. A software program written specially for that organisation to display that information on to the computer screen was common. However, now with the Internet using hypertext transfer protocol (HTTP _ the same technology that helps you surf the Net by clicking on links) to access and transfer data, the old way of storing data is of little use.

The data has to be stored in a way that it can be read by active server pages or Java programs. What active server pages do is to take a request made by a client, process it, access the database, collect the relevant information and send the reply back t o the client. The client in most cases is a person using a browser and submitting a request for say, the last quarter's sales report.

For a Net enterprise, it is important to configure the databases so that clients can access them remotely over the Net. For instance, in a business-to-business transaction, suppliers would like to access inventory levels of their clients to initiate supp ly of goods and bills. Or borrowers might want to access account information through the Net. In a legacy system where there is little connectivity, access was limited to a few people. Now to be connected on the Net the old data has to be marked up in say extensible markup language (XML) so that it can be easily accessed.Repackaging the data into modern databases is the most challenging aspect of warehousing. New enterprises should begin using modern databases at the first instance. On a very small scale MS Access, the database that comes with Microsoft Office 97, is a good example. However, for large enterprises much bigger databases are needed.

Old enterprises might have to go through the painful process of using data entry operators to re-input all the data in a Net-accessible format. This is much easier if data is stored in a tabular form. Plus for large organisations say like the State Bank of India, collecting data from all the branches and putting it all into one format will be a large-scale, costly operation.

This, however, cannot be a deterrent if one wants to compete or survive on the Net. Clients will soon demand access to the information from databases. This will be information pertaining to the client itself on which day-to-day business decisions will be made. If an enterprise is not able to provide it on the very convenient medium of the Internet, it might lose the client to a Net-savvy competitor.

10.5 Summary

- The response object is one of the five built-in ASP objects. Response is used to send output to the client. This output might be text displayed in a browser window, cookie data or it might have to do with how your pages are sent to the client and stored.
- The most common use of the Response object is to send data to the client's Web browser to be displayed as part of a Web page. It is done in two ways. The first is to use the Write method; the other is to use the shortcut < %=_%>.
- The most important thing in using Response. Write is that the string being written cannot contain "%>". If you need to write a string that contains "%>", use "%\>". Since we use %> to indicate the end of a block of ASP code, putting it in your strings will confuse the system.

Notes

- Server.HTML Encode can be used to send text encoded so that the browser will not interpret it as HTML.

```
<%=expression %> is equivalent to <% Response.Write (expression) %>.
```

- Besides sending output to the client, Response can control how and when the output is sent to the client. Output can be sent in two different ways: buffered or unbuffered. Unbuffered output is sent immediately. Buffered output is not sent until the script is finished, or until a special command is given to send it.
- The buffer property is a Boolean property that determines whether the output of your ASP is sent as it runs or it is stored until all the code is complete or the Flush method is called. It is a Boolean property which means that it is set to true or false.
- Suppose you have buffering turned on. As your script is executing, output is being sent to the buffer. Calling Response.Clear causes that buffer to be wiped out. In case, you have a page that you do not want to be viewed under a special set of circumstances, it can be used.
- Like Response.Clear, Response.Flush flushes all the data from the system buffer. However, Response.Flush first sends it to the client. This is useful Response.Clear, Response.Flush produces an error message when buffering is turned off.
- Response.End ends execution of the script. If buffering is turned on and there is any buffered data, it is sent. Any statements after the Response.End are not carried out. This is a more abrupt end than simply allowing the script to end on its own, and so should be avoided if possible. It can be useful, though, when problems (such as bad data) are detected to prevent them from making things worse.
- There are HTML Web pages that are on the screen for just a few seconds and then suddenly it takes you to another page. This can be done in many ways. One way is to use the META tag and the other is with client-side scripting like JavaScript, using the window object.
- Response.Redirect URL takes the user to the page URL. If it is within the same site, a relative URL (such as "products/index.html") will work. If it is a separate site, the full address including http:// should be provided.
- Cookies are a way for you to store nuggets of information on the visitor's computer. You can then use your code to retrieve the values stored on the visitor's system at a later time.You place the cookies on the visitor's machine by using the cookies collection of the Response object.
- The proxy serves as a funnel for many computers making requests to the Internet. One of the things proxy does is to store a cache of pages requested by all the users of the proxy. So instead of retrieving the same page many times form the Internet, the proxy merely returns the cached page to the person making the request.
- The cache control property is your way of instructing the proxy to cache or not to cache. HTML pages and graphics can be stored in the cache. Response has two properties that can be used to determine how long an ASP page be cached.These two properties are Response.Expires and Response.ExpiresAbsolute

10.6 Keywords

Buffer: This property can be use to indicate whether the page is buffered.

CacheControl: This property can be used to set whether a proxy server can cache the output generated by asp or not.

Cookies: This value can be used to set the cookie value.

Notes

Response Object: The response object is one of the five built-in ASP objects. Response is used to send output to the client.

Response.Clear: This method clears any buffered HTML output.

Response.End: This method stops the current processing of the asp page and only returns the result before this method.

Response.Flush: This method sends buffered output immediately.

Response.Redirect: This method can be used to redirect the user to another URL.

Response.Write: This method can be used to write out a variable or text.

Server.HTML Encode: Used to send text encoded so that the browser will not interpret it as HTML.

10.7 Review Questions

1. Examine the difference between Response.Flush and Response.Clear?
2. Write a Response.Write statement to send <A HREF="aboutme.html"> to the browser.
3. Write the code that sends the user to another page if it is an odd-numbered day and prints a message if it is an even-numbered day.
4. Write a statement using Response.ExpiresAbsolute that is equivalent to the following:
   (a) Response.Expires = 5
5. Write the code to have the cached version of the current page expire the first of the next month.
6. The response.write command is used to write output to a browser. Comment. Give a program to support your answer.
7. Explain the ways in which HTML can be send to the browser using the Write method and the shortcut method. Why is that when string is being written one cannot use %>?
8. Which one will result in an error: Response.Write (< HR LENGTH=30%> or <HR LENGTH=30\>)? Which one is correct: Response.Write (< IMG SRC= "Trial.gif">) or Response.Write ("<IMG SRC= ""Trial.gif"" >")?
9. Explain giving a few examples how you can use the Server.HTML Encode to send text coded so that the browser does not interpret it as HTML. What is the shortcut < %=_%> equivalent to?
10. When is it preferable to use the shortcut < %=_%> and when should you avoid using it? What are the ways, as mentioned in the chapter, through which you can send the user from one page to another?
11. What are cookies? How would place cookies on a visitor's machine? What property of the Response Object would you use to instruct the proxy whether to cache or not? What happens when you set the CacheControl value to "private"?

Answers: Self Assessment

1.	client	2.	Response.AddHeader
3.	Response	4.	Web page
5.	HTTP	6.	Buffered

Notes

7. Browser
8. String
9. Response. Write
10. Response.Redirect
11. Response.Flush
12. Response.End
13. Response.Clear
14. visitor's
15. code

10.8 Further Readings

Books

A Keyton Weissinger, *ASP in a Nutshell*, 2nd Edition [ILLUSTRATED] (Paperback), O'Reilly Media; 2 edition (January 1, 2000)

Bob Reselman, *Active Server Pages 3.0 by Example,* Que Publishing

David Buser, Chris Ullman, Brian Francis, Dave Sussman, *Beginning Active Server Pages 3.0,* John Wiley & Sons

John W. Gosney, *ASP Programming for the Absolute Beginner* (For the Absolute Beginner) (Paperback), Course Technology PTR; 1 edition (July 15, 2002)

Keith Morneau, Jill Batistick, *Active server pages,* Course Technology

Scott Mitchell, *Designing Active Server Pages,* O'Reilly Media

Online links

http://msdn.microsoft.com/en-us/library/ms525405(v=vs.90).aspx

http://www.itechies.net/tutorials/asp/index.phpindex-pid-res.htm

Notes

Unit 11: Using Request Objects

CONTENTS

Objectives

Introduction

11.1 Standard HTTP Headers

11.2 Accessing Environmental Variables

11.3 Using Cookies

11.3.1 Practical Uses for Cookies

11.3.2 What Information can a Cookie Extract?

11.3.3 Advantages and Disadvantages of Using Cookies

11.4 Summary

11.5 Keywords

11.6 Review Questions

11.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Recognize the standard HTTP headers
- Describe Environmental variables
- Demonstrate the Cookies

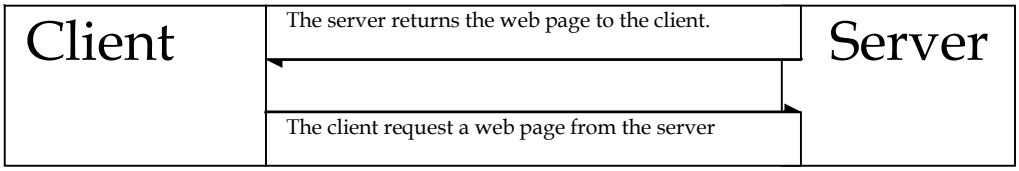
Introduction

The Request object can be used to get the information from the client HTTP header and body. A HTTP header is a single piece of information, sent either from the client to the server or from the server to the client, whenever a client request a page. The HTTP headers are useful for obtaining information about the current visitor. This unit will introduce you to HTTP headers, standard HTTP headers and how one can read these headers using the Request object. But we need more than HTTP headers to tell about the web server or the asp page that is being requested by the client, which leads us to environmental variables. Environmental variables contain information such as the name of the web server, the URL of the currently processing ASP page, or the name of the name of the web server software being used. Lastly you will learn about using cookies and to write cookies using the Response Object. Retrieving the Results of a Form.

Caution The Request object retrieves the values that the client browser passed to the server during an HTTP request.

Accessing the HTTP Headers (Useful HTTP Headers, Reading the HTTP Headers with Request, Server Variables). In a client-server model, when client is a web browser communicating with the server, requesting a web page.

Notes



When the client requests a web page from the server it not only sends the URL of the web page requested but also some additional information. This extra information consists of useful facts about the client for example, what browser is being used, what operating system the client is running, what URL the user just came from. Each piece of additional information is referred as a request header,

A request header is a single line of text that browser sends to the web server when requesting to view any web page.

When the server sends back the requested web page to the client, it also sends a set of headers, known as response headers. Response headers are additional bits of information about the web page being sent to the client. Both the request headers and the response headers are referred to, more generally, as HTTP headers.

Did u know? What is HTTP header?

An HTTP header is a single piece of information, sent either from the client to the server, when requesting a page, request.

11.1 Standard HTTP Headers

HTTP Header name	Description
HTTP_ACCEPT	A list of MIME type the client will Accept
HTTP_ACCEPT_LANGAGUE	what type of language the browser expects
HTTP_CONNECTION	the type of connection established between the client and web server
HTTP_HOST	the host name of the computer
HTTP_USER_AGENT	the browser type and version and operatingsystem information system of the client
HTTP_REFERER	the full URL of the web page containing the hyperlink use to reach the currently executing ASP page
HTTP_COOKIE	the cookies sent from the browser

Reading HTTP Headers with Request.ServerVariables

Using ASP, one can read the headers that the browser sends to the web server using the request object. Specifically, the use of the server variables is collections of the request objects. To display HTTP headers simply issue the following statement:-

```
<%=Request.ServerVariables("all_raw")%>
```

This displays the exact list of header sent be the browser to the web server to display a formulated list of headers, use the following commands:

```
<%=request.server variable ("HTTP_headername") %>
```

Notes

Notes

1. Must prefix the name of the header with HTTP
2. How Request.ServerVariables ("ALL_HTTP) formats the list of HTTP headers. All header names are capitalized and prefixed by HTTP_also, all dashes in the header names are replaced with underscores, and the space between the colons at the value of the header is removed. Request.ServerVariables ("ALL_RAW") performs no formatting to the request headers.
3. Reference header is present if the page has reached through a hyperlink on a different web page.

Task Give answers to following questions:

1. Explain what a HTTP header is. Name some of the standard HTTP headers.
2. Explain how HTTP headers are read with Request.ServerVariables.

11.2 Accessing Environmental Variables

Useful Environment Variables, Reading the Environment Variables, Using Request, ServerVariables.

The HTTP headers are useful for obtaining information about the current visitor but tell nothing about the web server or the asp page that is being requested by the client.

Environmental variables are bits of information that the web server makes available to any program that requests them. Environmental variables contain information such as the name of the web server, the URL of the currently processing ASP page, or the name of the name of the web server software being used.

Commonly used Environment Variables

Environment Variables	Description
URL	the URL of the ASP page from after http\\www.your.webserver.com/up to the query string.
Path_info	the same as the URL environment variable
Path_translated	the full, physical path of the currently executing ASP page.
Appl_physical_path	the physical address of the web's root directory
Query_string	the query string (equivalent to request every string)
Server_name	the web server's computer name
Server_software	the name of the web software

Reading Environmental Variables using Request. Server Variables

The environmental variables are accessed much like the HTTP headers. The Request. ServerVariables collection is used in the following format:

Notes

Request.Server Variables (Environment Variable Name)

List all server variables. ASP contains ASP code that lists all the items in the Request. ServerVariables collection. The server variable collection contains both environment variables and HTTP headers< both will be displayed when listing the contents of the collection.

Many environment variables do not contain a value. For example, the environment variables prefixed with CERT all contain empty strings as their values. This is because these variables are used only when the client and server use certificates. When a browser and web server communicate over a secure channel, certificates are used to ensure the identity of the client to the server.

All environment variables do not have values all the times. When a web server is not using SSL, the certificate environment variables are empty strings, much like the referrer HTTP header contains an empty string when the page is not visited via a hyperlink. There are some that are never empty such as URL, PATH_INFO, and PATH_ TRANSLATED

If you want to display the URL of the currently running ASP page, you would use the URL environment variable. The URL environment variable does not show the http: web server name, simply the full virtual path and filename. Another environment variable, Server_Name, contains the actual host name of the web server. The SERVER_NAME environment variable can be used in confection with the URL environment variable to retrieve the full URL of the ASP page.

Task What are environmental variables? Name some of the commonly used environmental variables.

Self Assessment

Fill in the blanks:

1. Using ASP, one can read the headers that the browser sends to the web server using the ……..
2. A …….. is a single line of text that browser sends to the web server when requesting to view any web page.
3. All environment variables do not have …….. all the times.
4. When a web server is not using SSL, the certificate environment variables are …….. strings.
5. The URL environment variable does not show the http: web server name, simply the full …….. and filename.
6. Response headers are additional bits of information about the …….. being sent to the client.
7. The server variable collection contains both …….. and HTTP headers both will be displayed when listing the contents of the collection.

11.3 Using Cookies

What are cookies? How to read Cookies, Using the Request Object, How to write Cookies using the Response Object, Advantages and disadvantages of using cookies.

Cookies are a general mechanism which server side connections (such as CGI scripts) can use to both store and retrieve information on the client side of the connection. The addition of a

simple, persistent, client-side state significantly extends the capabilities of Web-based client/server applications.

An Overview

A server, when returning an HTTP object to a client, may also send a piece of state information which the client will store. Included in that state object is a description of the range of URLs for which that state is valid. Any future HTTP requests made by the client which fall in that range will include a transmittal of the current value of the state object from the client back to the server. The state object is called a cookie, for no compelling reason.

This simple mechanism provides a powerful new tool which enables a host of new types of applications to be written for web-based environments. Shopping applications can now store information about the currently selected items, for fee services can send back registration information and free the client from retyping a user-id on next connection, sites can store per-user preferences on the client, and have the client supply those preferences every time that site is connected to.

Did u know? What is the specification of a cookie?

A cookie is introduced to the client by including a Set-Cookie header as part of an HTTP response, typically this will be generated by a CGI script.

Syntax of the Set-Cookie HTTP Response Header

This is the format a CGI script would use to add to the HTTP headers a new piece of data which is to be stored by the client for later retrieval.

```
Set-Cookie: NAME=VALUE; expires=DATE;
path=PATH; domain=DOMAIN_NAME; secure
NAME = VALUE
```

This string is a sequence of characters excluding semi-colon, comma and white space. If there is a need to place such data in the name or value, some encoding method such as URL style %XX encoding is recommended, though no encoding is defined or required.

This is the only required attribute on the Set-Cookie header.

```
expires=DATE
```

The expires attribute specifies a date string that defines the valid life time of that cookie. Once the expiration date has been reached, the cookie will no longer be stored or given out.

The date string is formatted as:

```
Wdy, DD-Mon-YYYY HH:MM:SS GMT
```

This is based on variations that the only legal time zone is GMT and the separators between the elements of the date must be dashes.

Expires is an optional attribute. If not specified, the cookie will expire when the user's session ends.

Notes There is a bug in Netscape Navigator version 1.1 and earlier. Only cookies whose path attribute is set explicitly to "/" will be properly saved between sessions if they have an expires attribute.

```
domain=DOMAIN_NAME
```

Notes

When searching the cookie list for valid cookies, a comparison of the domain attributes of the cookie is made with the Internet domain name of the host from which the URL will be fetched. If there is a tail match, then the cookie will go through path matching to see if it should be sent. "Tail matching" means that domain attribute is matched against the tail of the fully qualified domain name of the host. A domain attribute of "acme.com" would match host names "anvil.acme.com" as well as "shipping.crate.acme.com".

Only hosts within the specified domain can set a cookie for a domain and domains must have at least two (2) or three (3) periods in them to prevent domains of the form: ".com", ".edu", and "va.us". Any domain that fails within one of the seven special top level domains listed below only require two periods. Any other domain requires at least three. The seven special top level domains are: "COM", "EDU", "NET", "ORG", "GOV", "MIL", and "INT".

The default value of domain is the host name of the server which generated the cookie response.

path=*PATH*

The path attribute is used to specify the subset of URLs in a domain for which the cookie is valid. If a cookie has already passed domain matching, then the pathname component of the URL is compared with the path attribute, and if there is a match, the cookie is considered valid and is sent along with the URL request. The path "/foo" would match "/foobar" and "/foo/bar.html". The path "/" is the most general path.

If the path is not specified, it as assumed to be the same path as the document being described by the header which contains the cookie.

Secure

If a cookie is marked secure, it will only be transmitted if the communications channel with the host is a secure one. Currently this means that secure cookies will only be sent to HTTPS (HTTP over SSL) servers.

If secure is not specified, a cookie is considered safe to be sent in the clear over unsecured channels.

Syntax of the Cookie HTTP Request Header

When requesting a URL from an HTTP server, the browser will match the URL against all cookies and if any of them match, a line containing the name/value pairs of all matching cookies will be included in the HTTP request. Here is the format of that line:

```
Cookie: NAME1=OPAQUE_STRING1; NAME2=OPAQUE_STRING2 ...
```

11.3.1 Practical Uses for Cookies

Cookies were created to maintain user information and to customize web sites. In many cases, they make it easier to navigate and use the Internet.

Example: Upon are first visit to a site the user are often asked to register by giving name and a password for access to that site. The site will then place a cookie on the user's hard drive, which contains that information. When the user return to that site, the cookie is retrieved and read and the web site "recognizes" the user as an authorized guest. This means that the user only have to register once, instead of having to enter information every time the user access the site.

Because cookies allow a site to know who you are, they can customize information for you. It's like going into a store where the salesperson knows you personally and knows your preferences so he or she is able to present you with customized merchandise in order to make your shopping easier.

Notes

11.3.2 What Information can a Cookie Extract?

Cookies cannot be used to get data or view data off your hard drive. Cookies do not give anyone access to your computer or any personal information about you unless you have given that information to the website by answering questions or filling in a form. For example, the site cannot determine your e-mail name or your address unless you gave it to them. Cookies cannot give your computer a virus. Allowing a website to create a cookie does not give that or any other site access to the rest of your computer. Only the site that created the cookie can read it. And yet, cookies have a very bad reputation.

Loss of Privacy

The reason that cookies have gotten so much bad press recently is that cookies represent a potential loss of privacy. Cookies, by design are meant to work invisibly. They are used to track people and their activities and that makes many people uncomfortable.

Cookies can potentially be used to build detailed profiles of your interests, spending habits and lifestyle. An innocent use of this information might be to target advertising campaigns to specific groups or individuals. However, it is scary to contemplate the fact that some individual or group might be able to accumulate information about our private activities and personal preferences. There is a possibility that some unscrupulous group could potentially accumulate such information and sell it to companies to be used for their own purposes.

Cookies are like a personal tag or tracer. Some people see this as the most invasive of privacy. However, you must realize that every time you log on to a website you give away a lot of information. Any website that you visit can determine your:

- Service provider
- Operating System
- Browser type
- CPU type
- IP address

Cookie Use is Something to Ponder

The main concern about cookies is that they work without anyone's knowledge or permission. Some people consider the use of this information harmless, but some find the gathering of information in this manner invasive to their privacy. We, personally, do not mind the use of cookies, but I fear that this loss of privacy, however small, may lead to more loss of privacy as technology continues its onward march into our live

Task Analyze in a group of four that how cookies work without anyone's knowledge or permission?

11.3.3 Advantages and Disadvantages of Using Cookies

We used cookies in our application to allow us to store what the user has selected to be ordered. We used cookies for the following reasons:

- The data becomes attached to the customer. If he/she starts to make an order and then jumps to a different HTML document and returns later, the items that had been selected earlier are still selected.

Notes

- The Order Form could be on several screens, with each screen adding information to the cookie.
- The disadvantages of using cookies for this application are:
  - If a user has an old browser that doesn't support cookies, this application will not work.
  - In some browsers a user can turn on a warning every time an application attempts to write a cookie. If the user accepts the cookie, everything works fine.

Self Assessment

Fill in the blanks:

8. Cookies are a general mechanism which server side connections (such as CGI scripts) can use to both store and retrieve information on the ……. of the connection.
9. A…….., when returning an HTTP object to a client, may also send a piece of state information which the client will store.
10. Any future HTTP requests made by the client which fall in that range will include a transmittal of the current value of the ……. from the client back to the server.
11. If a cookie contains a collection of multiple values, we say that the cookie has …….
12. If your application deals with …… that do not support cookies, you will have to use other methods to pass information from one page to another in your application.
13. Some people consider the use of this information harmless, but some find the gathering of information in this manner invasive to their …….
14. A cookie is introduced to the client by including a …… header as part of an HTTP response.
15. …… cannot be used to get data or view data off your hard drive.

Caselet

Test Your Network

The focus of this week's pick of software is on tools for network testing. The programs featured here are mostly shareware, except where indicated.

Webserver Stress Tool

Webserver Stress Tool is an easy-to-use, versatile load and stress test for Web servers. It simulates any number of simultaneous users accessing a Web server and helps to streamline your Web application. Any Web page can be tested. It optionally handles proxies, passwords, user agents, cookies and ASP-session IDs.

Filesize: 2969 KB

Platform: Windows (All)

Download location: www.paessler.com

TracePlus32 Web Detective

TracePlus32 Web Detective is an HTTP trace/analysis tool specifically designed for Web development. It decodes HTTP, DASL, or WebDAV protocol and displays it in an easy-to-

Contd.....

Notes

understand format. HTTP protocol help is provided by a Windows help file which is hyperlinked to the Web Detective. 128 MB of RAM is recommended.

Filesize: 2030 KB

Platform: Windows (All)

Download location: www.sstinc.com/webanal.html

WAPT

WAPT is a load and stress-testing tool for Web sites and intranet applications. It provides ways for accurate load simulation, run-time test data generation and more.

You can use the task editor to create task or use the built-in recorder to record the actions of a virtual user, which can then be replayed, edited and tested. WAPT can simulate multiple users for each scenario, custom implement delays between requests and dynamically generate test data parameters. Additional features include support for secure sites and user authorisation, simultaneous testing, batch operation and more.

Filesize: 1650 KB

Platform: Windows (All)

Download location: www.loadtestingtool.com

WebPartner Test and Performance Center

This software measures Website performance from a customer's perspective. It provides a browser-based Web interface that allows you to quickly find, fix and prevent performance bottlenecks and failures within your Web applications.

Features include stress testing, Web service testings and URL monitoring. You can simulate unlimited simultaneous users accessing your Website applications from the Internet, test Web services and monitor any HTTP, HTTPS, XML and SOAP formatted transactions.

In addition, the URL monitoring option provides intelligent Website monitoring, escalation and notification features to help e-businesses ensure that Web pages are available and downloading quickly.

Filesize: 81075 KB

Platform: Windows NT/2K

Download location: www.webpartner.com

EasyWebLoad

EasyWebLoad (EWL) is a tool to load-test your company Website. You can use it to performance and stress test one page or multiple pages or an entire Website.

You can simulate the behaviour of virtual users and have full control of what URLs each virtual user uses, the URL loading sequence and also the time interval between Web page clicks.

EasyWebLoad supports static HTML pages, .JSPs, .ASPs and Perl CGI scripts as well as HTTP, Authentication, Proxy Server, .Get and .Post methods and name pair query strings for Get and Post methods. It does not support HTTPS (SSL) at this time.

Filesize: 852 KB

Platform: Windows (All)

Contd...

Notes

Download location: www.easywebload.com

Jblitz Pro

JBlitz Pro is a multi-threaded, multi-page Web site stress and load test tool. It exercises critical pathways on your site using hundreds of threads to simulate multiple concurrent users.

Features include cookie support, performance monitoring, auto error detection, fine-grained thread control, built-in logging, error string searching, graphing and thread activity display.

HTTP request headers are fully configurable, and you can view returned Web pages along with their headers. The program is fine for resilience testing dynamic sites with CGI, ASP, JSP, database access, etc.

Filesize: 7796 KB

Platform: Windows (All)

Download location: www.clanproductions.com

Web Roller

Web Roller is a load-testing tool that provides a consistent and reliable way to test and learn about performance characteristics of Web and Intranet applications before putting them on the market.

It records real-time user activity in terms of HTTP requests and creates a simple script, that you can modify using embedded test data generating capabilities. You can set various parameters and then run script or a batch of scripts and test performance.

Filesize: 1219 KB

Platform: Windows NT/2000/XP

Download location: www.webapplicationstesting.com

11.4 Summary

- A request header is a single line of text that browser sends to the web server when requesting to view any web page. When the server sends back the requested web page to the client, it also sends a set of headers, known as response headers. Response headers are additional bits of information about the web page being sent to the client. Both the request headers and the response headers are referred to, more generally, as HTTP headers.
- Some of the standard HTTP headers include HTTP_ACCEPT, HTTP_ACCEPT_LANGUAGE, HTTP_CONNECTION, HTTP_HOST, HTTP_USER_AGENT, HTTP_REFERER, HTTP_COOKIE
- Using ASP, one can read the headers that the browser sends to the web server using the request object. Specifically, the use of the server variables is collections of the request objects.
- Environmental variables are bits of information that the web server makes available to any program that requests them. Environmental variables contain information such as the name of the web server, the URL of the currently processing ASP page, or the name of the name of the web server software being used.
- The commonly used Environmental Variables are URL, Path_info, Path_translated, Appl_physical_path, Query_string, Server_name, Server_software

Notes

- The environmental variables are accessed much like the HTTP headers. ASP contains ASP code that lists all the items in the Request.ServerVariables collection. The server variable collection contains both environment variables and HTTP headers. Both will be displayed when listing the contents of the collection.
- Many environment variables do not contain a value. For example, the environment variables prefixed with CERT all contain empty strings as their values. This is because these variables are used only when the client and server use certificates.
- All environment variables do not have values all the times. When a web server is not using SSL, the certificate environment variables are empty strings. There are some that are never empty such as URL, PATH_INFO, and PATH_ TRANSLATED.
- URL environmental variable is used to display the URL of the currently running ASP page. The URL environment variable does not show the http: web server name, simply the full virtual path and filename.
- Another environment variable, Server_Name, contains the actual host name of the web server. The SERVER_NAME environment variable can be used in confection with the URL environment variable to retrieve the full URL of the ASP page.
- Cookies are a general mechanism which server side connections can use to both store and retrieve information on the client side of the connection.
- A server, when returning an HTTP object to a client, may also send a piece of state information which the client will store and included in that state object is a description of the range of URLs for which that state is valid. Any future HTTP requests made by the client, which fall in that range will include a transmittal of the current value of the state object from the client, back to the server. The state object is called a cookie. This simple mechanism provides a powerful new tool, which enables a host of new types of applications to be written for web-based environments.
- A cookie is introduced to the client by including a Set-Cookie header as part of an HTTP response. Syntax of the Set-Cookie HTTP Response Header is the format a CGI script would use to add to the HTTP headers a new piece of data which is to be stored by the client for later retrieval.
- If a cookie is marked secure, it will only be transmitted if the communications channel with the host is a secure one. If secure is not specified, a cookie is considered safe to be sent in the clear over unsecured channels.
- When requesting a URL from an HTTP server, the browser will match the URL against all cookies and if any of them match, a line containing the name/value pairs of all matching cookies will be included in the HTTP request.

11.5 Keywords

Cookies: Cookies are a general mechanism which server side connections can use to both store and retrieve information on the client side of the connection.

Environmental Variables: Environmental variables are bits of information that the web server makes available to any program that requests them.

HTTP Header: A HTTP header is a single piece of information, sent either from the client to the server or from the server to the client, when requesting a page. Thus, it includes both the request headers and the response headers.

HTTP: Hypertext Transfer Protocol, which is an application level protocol.

Notes

HTTPS: HTTP over Secure Socket Layer (SSL).

Request Header: A request header is a single line of text that browser sends to the web server when requesting to view any web page.

Request.Cookies Property: The Request.Cookies property can be used to retrieve the values stored in the client's cookie.

Request.ServerVariables: The Request.ServerVariables property can be used to retrieve information about the predefined server variables.

Response Header: Response headers are additional bits of information about the web page being sent to the client.

11.6 Review Questions

1. Examine the name of the response header that sends detailed information about the browser and operating system being used by the client.
2. Explain what environment variable returns the physical address of your Web's root directory?
3. What does the URL environment variable contain? What kinds of data types can a cookie not store?
4. Create an ASP page named DeleteAllCookies.asp. Write code that will delete all the cookies your Web site has created on the client's computer. (Hint: Remember that cookies are deleted when they expire!)
5. Substantiate ServerVariables collection? Some environmental variables do not contain values. Why is that? Name some environmental variables, which do not contain values.
6. Demonstrate some of the environmental variables, which do not contain variables all the time. Explain with the proper example.
7. Discuss some of the environmental variables which contain values all the time. Explain with the proper example.
8. The Cookies collection is used to set or get cookie values. Comment
9. Cookies were created to maintain user information and to customize web sites. Analyze
10. Cookies can potentially be used to build detailed profiles of your interests, spending habits and lifestyle. Do you agree with this statement? Why or why not? Give reasons to support your answer.

Answers: Self Assessment

1. request object
2. request header
3. values
4. empty
5. virtual path
6. web page
7. environment variables
8. client side
9. server
10. state object
11. Keys
12. Browsers
13. Privacy
14. Set-Cookie
15. Cookies

11.7 Further Readings

Notes

Books

A Keyton Weissinger, *ASP in a Nutshell*, 2nd Edition [ILLUSTRATED] (Paperback), O'Reilly Media; 2 edition (January 1, 2000)

Bob Reselman, *Active Server Pages 3.0 by Example,* Que Publishing

David Buser, Chris Ullman, Brian Francis, Dave Sussman, *Beginning Active Server Pages 3.0,* John Wiley & Sons

John W. Gosney, *ASP Programming for the Absolute Beginner* (For the Absolute Beginner) (Paperback), Course Technology PTR; 1 edition (July 15, 2002)

Keith Morneau, Jill Batistick, *Active server pages,* Course Technology

Scott Mitchell, *Designing Active Server Pages,* O'Reilly Media

Online links

http://www.w3schools.com/asp/coll_cookies_request.asp

http://www.webcheatsheet.com/ASP/request_object.php

Notes

Unit 12: Recordset Object

CONTENTS

Objectives

Introduction

12.1 The Recordset Object

12.1.1 CursorType Property

12.2 Filter Property

12.2.1 Settings and Return Values

12.3 Fields Collection

12.3.1 Sorting of Recordset

12.4 Summary

12.5 Keywords

12.6 Review Questions

12.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Define record set object
- Describe filter property
- Explain fields collection

Introduction

Enhancing Information Retrieval (Using the fields Collection); Understanding the cursor type and cursor location properties; sorting recordsets; filtering recordsets (filtering recordsets bases on user input).

The Recordset object represents a set of records returned from a database query. It is used to examine and manipulate data within a database. Combined with the cursor service, it enables us to move through the records, find particular records that fit certain criteria, sort records in a particular order, and update records.

The Recordset object is probably the most commonly used ADO object. It has a rather more complex interface than the other objects in the ADO object model which exposes many more methods and properties. Once we have created and populated a recordset, we can then use other parts of the interface to work with the contents of the recordset.

A Recordset object allows us to access individual records(the rows of the database) and fields (the columns). The set of fields associated with a recordset (and with each individual record) is accessible through the Fields collection and Field object.

Like the Command and Record objects, a Recordset can either exist on its own or be attached to a Connection. The latter is a preferred option when we are creating several recordsets within a

Notes

page, because it means that the connection to the data doesn't have to be opened each time we create a recordset. We can either update record in records in a recordset one record at a time, or we can batch a set of changes to various records, and then execute database update in one step. Recordset objects can also be disconnected from a data store, so that changes can be made to the data in an off-line state, and then updated when the recordset is reconnected to the database.

Caution This allows for the movement of entire recordsets from the server to client for update and manipulation.

12.1 The Recordset Object

Remarks

You use Recordset objects to manipulate data from a provider. When you use ADO, you manipulate data almost entirely using Recordset objects. All Recordset objects consist of records (rows) and fields (columns). Depending on the functionality supported by the provider, some Recordset methods or properties may not be available.

ADODB.Recordset is the ProgID that should be used to create a Recordset object. Existing applications that reference the outdated ADOR. Recordset ProgID will continue to work without recompiling, but new development should reference ADODB.Recordset.

Task Analyze practically that how Recordset object represents a set of records returned from a database query.

There are four different cursor types defined in ADO:

- ***Dynamic cursor*** — allows you to view additions, changes, and deletions by other users; allows all types of movement through the Recordset that doesn't rely on bookmarks; and allows bookmarks if the provider supports them.
- ***Keyset cursor*** — behaves like a dynamic cursor, except that it prevents you from seeing records that other users add, and prevents access to records that other users delete. Data changes by other users will still be visible. It always supports bookmarks and therefore allows all types of movement through the Recordset.
- ***Static cursor*** — provides a static copy of a set of records for you to use to find data or generate reports; always allows bookmarks and therefore allows all types of movement through the Recordset. Additions, changes, or deletions by other users will not be visible. This is the only type of cursor allowed when you open a client-side Recordset object.
- ***Forward-only cursor*** — allows you to only scroll forward through the Recordset. Additions, changes, or deletions by other users will not be visible. This improves performance in situations where you need to make only a single pass through a Recordset.

Notes

Set the CursorType property prior to opening the Recordset to choose the cursor type, or pass a CursorType argument with the Open method. Some providers don't support all cursor types. Check the documentation for the provider. If you don't specify a cursor type, ADO opens a forward-only cursor by default.

If the CursorLocation property is set to adUseClient to open a Recordset, the UnderlyingValue property on Field objects is not available in the returned Recordset object. When used with some providers (such as the Microsoft ODBC Provider for OLE DB in conjunction with Microsoft SQL Server), you can create Recordset objects independently of a previously defined Connection object by passing a connection string with the Open method. ADO still creates a Connection object, but it doesn't assign that object to an object variable. However, if you are opening multiple Recordset objects over the same connection, you should explicitly create and open a Connection object; this assigns the Connection object to an object variable.

Notes If you do not use this object variable when opening your Recordset objects, ADO creates a new Connection object for each new Recordset, even if you pass the same connection string.

12.1.1 CursorType Property

Indicates the type of cursor used in a Recordset object.

Settings and Return Values

Sets or returns a CursorTypeEnum value. The default value is adOpenForwardOnly.

Remarks

Use the CursorType property to specify the type of cursor that should be used when opening the Recordset object.

Only a setting of adOpenStatic is supported if the CursorLocation property is set to adUseClient. If an unsupported value is set, then no error will result; the closest supported CursorType will be used instead.

If a provider does not support the requested cursor type, it may return another cursor type. The CursorType property will change to match the actual cursor type in use when the Recordset object is open. To verify specific functionality of the returned cursor, use the Supports method. After you close the Recordset, the CursorType property reverts to its original setting.

The following chart shows the provider functionality (identified by Supports method constants) required for each cursor type.

For a Recordset of this CursorType	The Supports method must return True for all of these constants
adOpenForwardOnly	None
AdOpenKeyset	adBookmark, adHoldRecords, adMovePrevious, adResync
adOpenDynamic	AdMovePrevious
AdOpenStatic	adBookmark, adHoldRecords, adMovePrevious, adResync

Notes

Notes Although Supports (adUpdateBatch) may be true for dynamic and forward-only cursors, for batch updates you should use either a keyset or static cursor. Set the LockType property to adLockBatchOptimistic and the CursorLocation property to adUseClient to enable the Cursor Service for OLE DB, which is required for batch updates.

The CursorType property is read/write when the Recordset is closed and read-only when it is open.

Remote Data Service Usage When used on a client-side Recordset object, the CursorType property can be set only to adOpenStatic.

CursorLocation Property

Indicates the location of the cursor service.

Settings and Return Values

Sets or returns a Long value that can be set to one of the CursorLocationEnum values.

Remarks

This property allows you to choose between various cursor libraries accessible to the provider. Usually, you can choose between using a client-side cursor library or one that is located on the server.

This property setting affects connections established only after the property has been set. Changing the CursorLocation property has no effect on existing connections.

Cursors returned by the Execute method inherit this setting. Recordset objects will automatically inherit this setting from their associated connections.

This property is read/write on a Connection or a closed Recordset, and read-only on an open Recordset.

Remote Data Service Usage When used on a client-side Recordset or Connection object, the CursorLocation property can only be set to adUseClient.

You can create as many Recordset objects as needed.

When you open a Recordset, the current record is positioned to the first record (if any) and the BOF and EOF properties are set to False. If there are no records, the BOF and EOF property settings are True.

You can use the MoveFirst, MoveLast, MoveNext, and MovePrevious methods; the Move method; and the AbsolutePosition, AbsolutePage, and Filter properties to reposition the current record, assuming the provider supports the relevant functionality. Forward-only Recordset objects support only the MoveNext method. When you use the Move methods to visit each record (or enumerate the Recordset), you can use the BOF and EOF properties to determine if you've moved beyond the beginning or end of the RFilter Property

Task Evaluate the two types of updating that Recordset Object supports.

Notes

Self Assessment

Fill in the blanks:

1. The set of fields associated with a recordset (and with each individual record) is accessible through the ..... and Field object.
2. You use ..... objects to manipulate data from a provider.
3. ..... provides a static copy of a set of records for you to use to find data or generate reports.
4. Set the CursorType property prior to opening the Recordset to choose the cursor type, or pass a CursorType argument with the .....
5. If a ..... does not support the requested cursor type, it may return another cursor type.
6. Sets or returns a Long value that can be set to one of the ..... values.
7. To verify specific functionality of the returned cursor, use the ..... method.

12.2 Filter Property

Indicates a filter for data in a Recordset.

12.2.1 Settings and Return Values

Sets or returns a Variant value, which can contain one of the following:

- Criteria string — a string made up of one or more individual clauses concatenated with AND or OR operators.
- Array of bookmarks — an array of unique bookmark values that point to records in the Recordset object.
- A FilterGroupEnum value.

Remarks

Use the Filter property to selectively screen out records in a Recordset object. The filtered Recordset becomes the current cursor. Other properties that return values based on the current cursor are affected, such as AbsolutePosition, AbsolutePage, RecordCount, and PageCount. This is because setting the Filter property to a specific value will move the current record to the first record that satisfies the new value.

The criteria string is made up of clauses in the form FieldName-Operator-Value (for example, "LastName = 'Smith'"). You can create compound clauses by concatenating individual clauses with AND (for example, "LastName = 'Smith' AND FirstName = 'John'") or OR (for example, "LastName = 'Smith' OR LastName = 'Jones'"). Use the following guidelines for criteria strings:

- FieldName must be a valid field name from the Recordset. If the field name contains spaces, you must enclose the name in square brackets.
- Operator must be one of the following: <, >, <=, >=, <>, =, or LIKE.
- Value is the value with which you will compare the field values (for example, 'Smith', #8/24/95#, 12.345, or $50.00). Use single quotes with strings and pound signs (#) with dates.

Notes

For numbers, you can use decimal points, dollar signs, and scientific notation. If Operator is LIKE, Value can use wildcards. Only the asterisk (*) and percent sign (%) wild cards are allowed, and they must be the last character in the string. Value cannot be null.

Notes To include single quotation marks (') in the filter Value, use two single quotation marks to represent one. For example, to filter on O'Malley, the criteria string should be "col1 = 'O''Malley'". To include single quotation marks at both the beginning and the end of the filter value, enclose the string with pound signs (#). For example, to filter on '1', the criteria string should be "col1 = #'1'#".

- There is no precedence between AND and OR. Clauses can be grouped within parentheses. However, you cannot group clauses joined by an OR and then join the group to another clause with an AND, like this:

```
(LastName = 'Smith' OR LastName = 'Jones') AND FirstName = 'John'
```

- Instead, you would construct this filter as

```
LastName = 'Smith' AND FirstName = 'John') OR (LastName = 'Jones'
AND FirstName = 'John')
```

- In a LIKE clause, you can use a wildcard at the beginning and end of the pattern

Example: LastName Like '*mit*'

, or only at the end of the pattern

Example: LastName Like 'Smit*'

The filter constants make it easier to resolve individual record conflicts during batch update mode by allowing you to view, for example, only those records that were affected during the last UpdateBatch method call.

Setting the Filter property itself may fail because of a conflict with the underlying data (for example, a record has already been deleted by another user). In such a case, the provider returns warnings to the Errors collection but does not halt program execution. A run-time error occurs only if there are conflicts on all the requested records. Use the Status property to locate records with conflicts.

Setting the Filter property to a zero-length string ("") has the same effect as using the adFilterNone constant.

Whenever the Filter property is set, the current record position moves to the first record in the filtered subset of records in the Recordset. Similarly, when the Filter property is cleared, the current record position moves to the first record in the Recordset.

When a Recordset is filtered based on a field of some variant type (e.g., sql_variant), an error (DISP_E_TYPEMISMATCH or 80020005) will result if the subtypes of the field and filter values used in the criteria string do not match. For example, if a Recordset object (rs) contains a column (C) of the sql_variant type and a field of this column has been assigned a value of 1 of the I4 type, setting the criteria string of rs.Filter = "C='A'" on the field will produce the error at run time. However, rs.Filter = "C=2" applied on the same field will not produce any error although the field will be filtered out of the current record set.

Notes

Did u know? What are Filters?

Only Filters in the form of Criteria Strings (e.g. OrderDate > '12/31/1999') affect the contents of a persisted Recordset. Filters created with an Array of Bookmarks or using a value from the FilterGroupEnum will not affect the contents of the persisted Recordset. These rules apply to Recordsets created with either client-side or server-side cursors.

When you apply the adFilterPendingRecords flag to a filtered and modified Recordset in the batch update mode, the resultant Recordset is empty if the filtering was based on the key field of a single-keyed table and the modification was made on the key field values.

The resultant Recordset will be non-empty if one of the following is true:

- The filtering was based on non-key fields in a single-keyed table.
- The filtering was based on any fields in a multiple-keyed table.
- Modifications were made on non-key fields in a single-keyed table.
- Modifications were made on any fields in a multiple-keyed table.

The following table summarizes the effects of adFilterPendingRecords in different combinations of filtering and modifications. The left column shows the possible modifications; modifications can be made on any of the non-keyed fields, on the key field in a single-keyed table, or on any of the key fields in a multiple-keyed table. The top row shows the filtering criterion; filtering can be based on any of the non-keyed fields, the key field in a single-keyed table, or any of the key fields in a multiple-keyed table. The intersecting cells show the results: + means that applying adFilterPendingRecords results in a non-empty Recordset; - means an empty Recordset.

	Non keys	**Single Key**	**Multiple Keys**
Non keys	+	+	+
Single Key	+	-	N/A
Multiple Keys	+	N/A	+

Before using any functionality of a Recordset object, you must call the Supports method on the object to verify that the functionality is supported or available. You must not use the functionality when the Supports method returns false.

Example: you can use the MovePrevious method only if Recordset. Supports (adMovePrevious) returns true.

Otherwise, you will get an error, because the Recordset object might have been closed and the functionality rendered unavailable on the instance. If a feature you are interested in is not supported, Supports will return false as well. In this case, you should avoid calling the corresponding property or method on the Recrodset object.

Recordset objects can support two types of updating: immediate and batched. In immediate updating, all changes to data are written immediately to the underlying data source once you call the Update method. You can also pass arrays of values as parameters with the AddNew and Update methods and simultaneously update several fields in a record.

If a provider supports batch updating, you can have the provider cache changes to more than one record and then transmit them in a single call to the database with the UpdateBatch method. This

applies to changes made with the AddNew, Update, and Delete methods. After you call the UpdateBatch method, you can use the Status property to check for any data conflicts in order to resolve them.

Notes To execute a query without using a Command object, pass a query string to the Open method of a Recordset object. However, a Command object is required when you want to persist the command text and re-execute it, or use query parameters.

The Mode property governs access permissions.

The Fields collection is the default member of the Recordset object. As a result, the following two code statements are equivalent.

Debug.Print objRs.Fields.Item(0) ' Both statements print

Debug.Print objRs(0) ' the Value of Item(0).

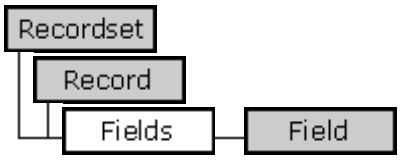
Self Assessment

Fill in the blanks:

8. An …… of unique bookmark values that point to records in the Recordset object.
9. The filtered Recordset becomes the …… cursor.
10. The criteria string is made up of …… in the form FieldName-Operator-Value
11. Whenever the …… is set, the current record position moves to the first record in the filtered subset of records in the Recordset.
12. The …… property governs access permissions.

12.3 Fields Collection

Contains all the Field objects of a Recordset or Record object.



Remarks

A Recordset object has a Fields collection made up of Field objects. Each Field object corresponds to a column in the Recordset. You can populate the Fields collection before opening the Recordset by calling the Refresh method on the collection.

The Fields collection has an Append method, which provisionally creates and adds a Field object to the collection, and an Update method, which finalizes any additions or deletions.

A Record object has two special fields that can be indexed with FieldEnum constants. One constant accesses a field containing the default stream for the Record, and the other accesses a field containing the absolute URL string for the Record.

Notes

Certain providers (for example, the Microsoft OLE DB Provider for Internet Publishing) may populate the Fields collection with a subset of available fields for the Record or Recordset. Other fields will not be added to the collection until they are first referenced by name or indexed by your code.

If you attempt to reference a nonexistent field by name, a new Field object will be appended to the Fields collection with a Status of adFieldPendingInsert. When you call Update, ADO will create a new field in your data source if allowed by your provider.

When a Recordset object is passed across processes, only the rowset values are marshalled, and the properties of the Recordset object are ignored. During unmarshalling, the rowset is unpacked into a newly created Recordset object, which also sets its properties to the default values.

The Recordset object is safe for scripting.

Task Give answers to the following questions:

1. What are the Filter Property Return Values?
2. Explain the The Recordset.Fields Collection.

12.3.1 Sorting of Recordset

We may use SQL to specify how to sort the data in the record set.

Sort the records on a specified fieldname ascending

```
<html>
<body>
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers ORDER BY CompanyName"
rs.Open sql, conn
%>
<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%></td>
```

Notes

```
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
</table>
</body>
</html>
Sort the records on a specified fieldname descending
<html>
<body>
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers ORDER BY CompanyName
DESC"
rs.Open sql, conn
%>
<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%> </td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
```

Notes

```
</table>
</body>
</html>
```

Let the user choose what column to sort on

```
<html>
<body>
<table border="1" width="100%" bgcolor="#fff5ee">
<tr>
<th align="left" bgcolor="#b0c4de">
<a href="demo_sort_3.asp?sort=companyname">Company</a>
</th>
<th align="left" bgcolor="#b0c4de">
<a href="demo_sort_3.asp?sort=contactname">Contact</a>
</th>
</tr>
<%
if request.querystring("sort")<>"" then
   sort=request.querystring("sort")
else
   sort="companyname"
end if
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs=Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname,Contactname FROM Customers ORDER BY " & sort
rs.Open sql,conn
do until rs.EOF
   response.write("<tr>")
   for each x in rs.Fields
     response.write("<td>" & x.value & "</td>")
   next
   rs.MoveNext
   response.write("</tr>")
loop
rs.close
conn.close
%>
```

Notes

```
</table>
</body>
</html>
```

Self Assessment

Fill in the blanks:

13. A Record object has two special fields that can be indexed with …… constants.
14. During……., the rowset is unpacked into a newly created Recordset object, which also sets its properties to the default values.
15. Each Field object corresponds to a …… in the Recordset.

Caselet

Google Runs 'A Little Bit Like a University'

If Google has underwear, who will buy it? This was one of the questions that Sergey Brin posed to students of an Israeli high school in 2003. "Hands shot up," writes David A. Vise in

The Google Story

, from Macmillan (www.panmacmillan.com) .

To Brin and Larry Page, the Google guys, named recently 'Men of the Year' by

Financial Times

, the project was 'one of the less technical' ones.

To check, you may use Google to briefly search for the brief, and see a March 2005 posting on http://blog.searchenginewatch.com confirming thus: "UnderGoos (naturally in beta) offers a full-line of Google-branded undergarments." Price, however, is £0.00 on www.google-store.com. "A project that is closer to our hearts is translation," Brin told the students.

The book takes one 'inside the hottest business, media and technology success of our time,' to know how Google, a start-up investment of $1 million, became a market value of $40 billion, all by word of mouth.

"With its colourful, childlike logo set against a background of pure white, Google's magical ability to produce speedy, relevant responses to queries hundreds of millions of times daily has changed the way people find information and stay abreast of news," writes Vise in the introduction.

Ten years ago, search was 'not pretty,' is how Rajeev Motwani, "a 30-year old professor who had been Sergey's advisor since his arrival at Stanford in 1993", found.

Vise writes about how Motwani had tested a search engine called Inktomi after it was developed at Berkeley, where he had received his Ph.D.

"He typed in `Inktomi' to see what would happen. Sure enough, said Motwani, `It wasn't there. It couldn't find itself." Not so with Google, which shows 800,000,000 results for itself in 0.17 seconds.

Contd...

Notes

"When Google went looking for someone to ramp up its computer network, Larry and Sergey hired a brain surgeon, Dr Jim Reese," informs the book. "Named Google's operations chief, Reese managed the company's burgeon collection of computer hardware."

Vise narrates the story of how the company "cobbled together a virtual supercomputer from cheap, commodity PCs." Back at the Googleplex, the garden-variety PCs got ripped apart, and all `unnecessary parts that would eat up computing power and resources' disposed of, to build streamlined computers, strung together with "software, wiring, and the special sauce that made Google lightning fast."

More than one lakh inexpensive PCs, stacked in refrigerator-size racks, remain 'strictly off-limits to outsiders', notes Vise. "PCs burn out and are not replaced. Instead other PCs take over."

Learn about the 20 per cent rule in Google from Krishna Bharat, a software engineer in the company's research group.

The rule stipulates that engineers spend at least 20 per cent of their time, or one day a week, working on whatever projects interested them.

"The 20 per cent rule was a way of encouraging innovation, and both Brin and Page saw this as essential to establishing and maintaining the right culture and creating a place where bright technologists would want to work and be motivated to come up with breakthrough ideas." 3M had something similar - the 15 per cent rule - many years earlier, notes Vise. A success product that had emerged from such a pursuit was Post-it Notes.

"Rather than having employees moonlighting as inventors at home - with the risk that an idea will either fail from lack of resources or succeed to the point that they quit to pursue it full-time - Google gives them both freedom and resources," observes Vise.

When speaking at the Israeli school, Brin had said, "We run Google a little bit like a university. We have lots of projects, about 100 of them. We like to have small groups of people, three or so people, working on projects. Some of them, for example, are related to molecular biology. Others involve building hardware... The only way you are going to have success is to have lots of failures first."

And Brin toys with the idea of plugging into brain 'a little version of Google', as you'd know from the final chapter, titled 'Googling your genes.'

Dr Craig Venter, who had decoded the human genome, is of the view that genetic information is going to be the leading edge. "Working with Google, we are trying to generate a gene catalogue to characterise all the genes on the planet and understand their evolutionary development. The massive computing power can be used "to analyse vast quantities of data with billions of parts" says Dr Alan E. Guttmacher, deputy director of the National Human Genome Research Institute, cited in the book.

"We are beginning to have incredible tools to understand the biology of human diseases in ways we never have before, and to come up with novel ways to prevent and treat them."

Fabulous read.

All about ADO.NET

SAHIL Malik, who has been working as a consultant in Microsoft technology for about a decade, and also leading the office of Emerging Technologies at the National Cancer Institute, has written

Contd...

Notes

Pro ADO.NET 2.0

from Dreamtech Press (www.wileydreamtech.com) . "ActiveX Data Objects (ADO) was the premier data access technology under the Microsoft umbrella before ADO.NET was introduced as an integral part of the .NET Framework," chronicles Malik.

"What sets ADO.NET apart from previous data access technologies is that it allows you to interact with your database in a completely disconnected data cache to work with data offline."

Disconnected data access is crucial for today's high-demand applications, notes the author.

The book has chapters on connecting to a data source, retrieving data in a connected fashion, DataSets, sorting and searching, updating data, and so on. Of value is the chapter on 'best practices' where Malik discusses the right tools. For instance, he reminds that data reader consumes less memory than a DataSet.

"A data reader is an object that allows you to access information on only a single row of data at a given time.

What this means is that, regardless of the size of a result set, traversing this result set with a data reader will only ever have a single record loaded in memory at a given time."

There are many flavours of transactions to choose from, writes Malik, listing out the same 'in an increasing order of management overhead and decreasing order of performance'.

The list begins with implicit transactions, which are automatically associated with any single SQL statement and ensure "the sanctity of the data during that statement's execution time period," and ends with "storing a snapshot of previous data, which acts as your `recovery contract' and a flag on the `in doubt' rows."

Helpful notes are strewn all over the book. One such reads, "Retrieving a large volume of data within the context of a single connection will always be faster than retrieving small portions and opening and closing the connection each time, because large-block retrieval causes less network roundtrips and incurs less latency."

Useful for the ADO techie.

12.4 Summary

- The Recordset object represents a set of records returned from a database query. It is used to examine and manipulate data within a database. Combined with the cursor service, it enables us to move through the records, find particular records that fit certain criteria, sort records in a particular order, and update records.
- A Recordset object allows us to access individual records and fields. The set of fields associated with a recordset (and with each individual record) is accessible through the Fields collection and Field object.
- A Recordset can either exist on its own or be attached to a Connection. The latter is a preferred option when we are creating several recordsets within a page, because it means that the connection to the data doesn't have to be opened each time we create a recordset. We can either update record in records in a recordset one record at a time, or we can batch a set of changes to various records, and then execute database update in one step.
- Recordset objects can also be disconnected from a data store, so that changes can be made to the data in an off-line state, and then updated when the recordset is reconnected to the database. This allows for the movement of entire recordsets from the server to client for update and manipulation.

Notes

- ADODB.Recordset is the ProgID that should be used to create a Recordset object. Existing applications that reference the outdated ADOR. Recordset ProgID will continue to work without recompiling, but new development should reference ADODB.Recordset.
- There are four different cursor types defined in ADO. They are dynamic cursor, keyset cursor, static cursor and forward-only cursor.
- Set the CursorType property prior to opening the Recordset to choose the cursor type, or pass a CursorType argument with the Open method.
- CursorType Property indicates the type of cursor used in a Recordset object. It set or returns a CursorTypeEnum value. The default value is adOpenForwardOnly.
- CursorLocation Property indicates the location of the cursor service. It sets or returns a Long value that can be set to one of the CursorLocationEnum values.
- Filter Property indicates a filter for data in a Recordset. It sets or returns a Variant value, which can contain one of the following (a) Criteria string: a string made up of one or more individual clauses concatenated with AND or OR operators (b) Array of bookmarks: an array of unique bookmark values that point to records in the Recordset object c) A FilterGroupEnum value
- Fields Collection contains all the Field objects of a Recordset or Record object.
- A Recordset object has a Fields collection made up of Field objects. Each Field object corresponds to a column in the Recordset. You can populate the Fields collection before opening the Recordset by calling the Refresh method on the collection.
- The Fields collection has an Append method, which provisionally creates and adds a Field object to the collection, and an Update method, which finalizes any additions or deletions.
- A Record object has two special fields that can be indexed with FieldEnum constants. One constant access a field containing the default stream for the Record, and the other accesses a field containing the absolute URL string for the Record.
- If you attempt to reference a nonexistent field by name, a new Field object will be appended to the Fields collection with a Status of adFieldPendingInsert. When you call Update, ADO will create a new field in your data source if allowed by your provider.
- When a Recordset object is passed across processes, only the rowset values are marshalled, and the properties of the Recordset object are ignored. During unmarshalling, the rowset is unpacked into a newly created Recordset object, which also sets its properties to the default values.

12.5 Keywords

ADODB.Recordset: It is the ProgID that should be used to create a Recordset object.

Append Method: It provisionally creates and adds a Field object to the collection.

CursorLocation Property: It indicates the location of the cursor service. It sets or returns a Long value that can be set to one of the CursorLocationEnum values.

CursorType Property: Indicates the type of cursor used in a Recordset object.

Dynamic Cursor: Allows you to view additions, changes, and deletions by other users.

Field Object: It corresponds to a column in the Recordset.

Fields Collection: It contains all the Field objects of a Recordset or Record object.

Filter Property: It indicates a filter for data in a Recordset. It sets or returns a Variant value, which can contain either criteria string, array of bookmarks or a FilterGroupEnum value.

Notes

Forward-only Cursor: Allows you to only scroll forward through the Recordset.

Keyset Cursor: Behaves like a dynamic cursor, except that it prevents you from seeing records that other users add, and prevents access to records that other users delete.

Static Cursor: Provides a static copy of a set of records for you to use to find data or generate reports. This is the only type of cursor allowed when you open a client-side Recordset object.

The Recordset Object: It represents a set of records returned from a database query. It is used to examine and manipulate data within a database.

Update method: It finalizes any additions or deletions.

12.6 Review Questions

1. Examine the default property of the Field object? Give examples
2. Explain how many field objects exist in the Fields collection?
3. What does the name property of the Field object return? Is it possible to both sort and filter a Record set?
4. Create two ASP pages, a form creation Web page (SelectPrice.asp) and a form processing script (ListStocksByPrice.asp). In SelectPrice.asp, the user should be shown a form into which he can enter a desired maximum price. When the for is submitted, ListStocksByPrice.asp will list all the stocks in the Portfolio table that cost strictly less than the price entered by the user.
5. The Recordset object is probably the most commonly used ADO object. Comment
6. Substantiate what are the types of cursors defined in ADO?
7. Analyze how the cursor type set and using which method it is set?
8. The cursor type can be set by the CursorType property or by the CursorType parameter in the Open method. Explain with the program.
9. The ADO Recordset object is used to hold a set of records from a database table. Explain
10. In ADO, this object is the most important and the one used most often to manipulate data from a database. Do you agree with this statement? Why or why not? Give reasons to support your answer.

Answers: Self Assessment

1. Fields collection
2. Recordset
3. Static cursor
4. Open method
5. provider
6. CursorLocationEnum
7. Supports
8. Array
9. Current
10. Clauses
11. Filter property
12. Mode
13. FieldEnum
14. unmarshalling
15. column

Notes

12.7 Further Readings

Books

Greg Buczek, *Instant ASP Scripts* (Paperback), McGraw-Hill Companies; 2nd Bk&Cdr edition (August 22, 2000)

John Kauffman, *Kevin Spencer*, Thearon Willis, *Beginning ASP Databases* (Paperback), Apress; 1 edition (August 26, 2003)

Online links

http://www.w3schools.com/ado/ado_ref_recordset.asp

http://msdn.microsoft.com/en-us/library/windows/desktop/ms681510(v=vs.85).aspx

Notes

Unit 13: ASP Cookies and Caching

CONTENTS

Objectives
Introduction
13.1 ASP Procedures
13.2 Cookies
13.2.1 Setting a Cookie
13.3 ASP File System
13.4 Send E-Mail
13.5 Caching
13.5.1 Fragment Caching, User Control Output Caching
13.5.2 Page Caching
13.5.3 Data Caching
13.6 Summary
13.7 Keywords
13.8 Review Questions
13.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Scan the ASP Procedures
- Describe Cookies
- Demonstrate the ASP file system
- Recognize sending e-mail
- Explain caching

Introduction

This unit on cookies will introduce you to what cookies are and how browsers use these cookies for maintaining state for developing critical interactive applications. You will learn about set-cookie and cookie fields, which are used to transfer information between the client and the server.

13.1 ASP Procedures

In ASP you can call a JavaScript procedure from a VBScript and vice versa.

Procedures

The ASP source code can contain procedures and functions:

Notes

Example:

```
<html>
<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>
</head>
<body>
<p>Result: <%call vbproc(3,4)%></p>
</body>
</html>
```

Show example »

Insert the <%@ language="language" %> line above the <html> tag to write the procedure/function in another scripting language:

Example:

```
<%@ language="javascript" %>
<html>
<head>
<%
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
%>
</head>
<body>
<p>Result: <%jsproc(3,4)%></p>
</body>
</html>
```

13.2 Cookies

One of the challenges of writing applications for the World Wide Web has been inability of the web to maintain state. That is, after a user sends a request to the server and a web page is returned, the server forgets all about the user and the page that has been downloaded. If the user clicks on a link, the server does not have background information about what page the user is coming from, and more importantly, if the user returns to the page at a later date, there is no information available to the server about the user's previous actions on the page.

Maintaining state can be important to developing complex interactive applications. However, browsers address this problem with cookies, which is a method of storing information locally in the browser and sending it to the server whenever the appropriate pages are requested by the user.

Notes

The term cookies has no special significance. It is just as name. When a user requests a page, an HTTP request is sent to the server. The request includes a header that defines several pieces of information, including the page being requested. The server returns an HTTP response that also includes a header.

Notes The header contains information about the document being returned, including its MIME type.

These headers all contain one or more fields of information in a basic format.

FieldName: Information

Cookie information is shared between the client browser and a server using fields in the HTTP headers. When the user requests a page for the first time, a cookie (or more than one cookie) can be stored in the browser by a set-cookie entry in the header of the response from the server. The set-cookie field includes the information to be stored in the cookie along with several optional pieces of information, including an expiry date, path, and server information and if the cookie requires security.

Then, when the user requests a page in the future, if a matching cookie is found among all the stored cookies, the browser sends a cookie field to the server in a request header. The header will contain the information stored in that cookie.

Did u know? How Cookie use the syntax?

The set-cookie and cookie fields use syntax to transfer significant information between client and server.

13.2.1 Setting a Cookie

Syntax

```
Set-cookie: NAME=value; EXPIRES=date; PATH=path; DOMAIN=domain; SECURE
```

The NAME=value is the only required piece of information that must be included in the set-cookie field. All other entries are optional.

Name	Description
NAME=value	Specifies the name of the cookie.
EXPIRES=date	Specifies the expiry date of the cookie. After this date the cookie will no longer be stored by the client or sent to the server. DATE takes the form WDY, DD-MON-YY HH:MM:SS GMT. By default, the value of expires is set to the end of current Navigator session.
PATH=path	Specifies the path portion of the URLs for which the cookie is valid. If the URL matches both the PATH and the DOMAIN, then the cookie is sent to the server in the request header. If left unset, the value of the PATH is the same as the document that set the cookie.
DOMAIN=domain	Specifies the domain portion of the URLs for which the cookie is valid. The default value for this attributes is the domain of the current document setting the cookie.
SECURE	Specifies that the cookie should only be transmitted over a secure link (i.e., HTTP servers using SSL protocol known as HTTPS server).

Notes

Self Assessment

Fill in the blanks:

1. Creating an ASP cookie is exactly the same process as creating an ……
2. The created cookie will store the value which contains the …… data.
3. To get the information we have stored in the cookie we must use the ……
4. A cookie's expiration can hold a date; this date will specify when the cookie will be ……
5. The …… is the only required piece of information that must be included in the set-cookie field.
6. When a user requests a page, an …… request is sent to the server.
7. A cookie is a …… given to a Web browser by a Web server.

13.3 ASP File System

The FileSystemObject Object

The FileSystemObject object is used to access the file system on a server.

This object can manipulate files, folders, and directory paths. It is also possible to retrieve file system information with this object.

The following code creates a text file (c:\test.txt) and then writes some text to the file:

```
<%
dim fs,fname
set fs=Server.CreateObject("Scripting.FileSystemObject")
set fname=fs.CreateTextFile("c:\test.txt",true)
fname.WriteLine("Hello World!")
fname.Close
set fname=nothing
set fs=nothing
%>
```

The FileSystemObject object's properties and methods are described below:

Properties

Property	Description
Drives	Returns a collection of all Drive objects on the computer

Methods

Method	Description
BuildPath	Appends a name to an existing path
CopyFile	Copies one or more files from one location to another
CopyFolder	Copies one or more folders from one location to another
CreateFolder	Creates a new folder

Contd...

Notes

CreateTextFile	Creates a text file and returns a TextStream object that can be used to read from, or write to the file
DeleteFile	Deletes one or more specified files
DeleteFolder	Deletes one or more specified folders
DriveExists	Checks if a specified drive exists
FileExists	Checks if a specified file exists
FolderExists	Checks if a specified folder exists
GetAbsolutePathName	Returns the complete path from the root of the drive for the specified path
GetBaseName	Returns the base name of a specified file or folder
GetDrive	Returns a Drive object corresponding to the drive in a specified path
GetDriveName	Returns the drive name of a specified path
GetExtensionName	Returns the file extension name for the last component in a specified path
GetFile	Returns a File object for a specified path
GetFileName	Returns the file name or folder name for the last component in a specified path
GetFolder	Returns a Folder object for a specified path
GetParentFolderName	Returns the name of the parent folder of the last component in a specified path
GetSpecialFolder	Returns the path to some of Windows' special folders
GetTempName	Returns a randomly generated temporary file or folder
MoveFile	Moves one or more files from one location to another
MoveFolder	Moves one or more folders from one location to another
OpenTextFile	Opens a file and returns a TextStream object that can be used to access the file

The FileSystem property returns the file system in use for a specified drive.

This property will return one of the following:

- FAT - for removable drives
- CDFS - for CD-ROM drives
- FAT, FAT32 or NTFS - for hard disks on Windows 2000 or Windows NT
- FAT or FAT32 - for hard disks on Windows 9x

Syntax

DriveObject.FileSystem

Example:

```
<%
dim fs,d
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
Response.Write("The file system in use is: " & d.FileSystem)
set d=nothing
set fs=nothing
%>
```

Notes

Output:

```
The file system in use is: NTFS
```

Self Assessment

Fill in the blanks:

8. …… returns the file name or folder name for the last component in a specified path.
9. …… copies one or more folders from one location to another.
10. …… returns the complete path from the root of the drive for the specified path.
11. …… returns the path to some of Windows' special folders

13.4 Send E-Mail

Using E-mail from shell Account

Let us review how E-Mail works, using an example. In this example, you are using a PC with Windows OS, which connects to the Internet using TCP/IP (i.e., PPP). Let us suppose you want to send a mail (or message) to two of your friends: Nitin in Kolkata and Puja in Germany. Nitin uses a Macintosh and also connects to the Net using PPP. Puja uses a shell account by connecting to a UNIX host computer. The following steps illustrate the example:

1. First using a Windows mail client, you compose the message on your own computer.
2. After you compose the message, address it to both Nitin and Puja.
3. Once the message is finished, you tell your program to send it on its way.
4. Now your client program contacts the mail server on your Internet host and, using the POP protocol, sends your message to the server.
5. In the next step, the server passes your message to the transport agent.
6. Now, it is the job of transport agent to look at the addresses in your message and connect to the appropriate computers over the Net.
7. First, the transport agent connects it on the host computer in Kolkata that receives mail for Nitin.
8. Once the connection is made, the two transport agents use the SMTP protocol to relay the message.
9. After the message is sent, your transport agent terminates the connection and forms a new connection with the transport agent on the appropriate computer in Germany.

Task In a group of four make a report on advantages of E-Mail in IT company.

10. Again, the two transport agents use SMTP to relay the message.
11. Once the message is sent, your transport agent terminates the connection. Its job is finished.

Notes

12. In Kolkata, Nitin turns on his computer to check the mail. He tells his Macintosh mail client to see if any new mail has arrived. Now it is the turn of his mail client to connect to the mail server on Nitin's host computer and using the POP protocol, asks the server to check Nitin's mailbox. Since server finds your message, so using POP, it sends the message to the client and places the message in his local mailbox (a file on the Mac) and tells him that new mail has arrived. Now, with the help of mail program, Nitin displays the message.
13. Similarly, in Germany, Puja has logged into her shell account on a Unix host. She runs her UNIX mail program which checks her mailbox and tells her new mail has arrived. Using appropriate command, Puja tells the mail program to show her your message.

Did u know? What are the components of E-Mail?

An email message consists of three components, the message envelope, the message header, and the message body.

Basic E-Mail Functions

E-Mail programs tend to work alike, even down to the keystrokes required to execute a command. Whichever E-Mail application you use, you do the same basic tasks. There are only a few basic functions in E-Mail, and almost all mailers handle them. They are:

- Read
- Compose (new messages)
- Reply (to messages you have received)
- Forward (messages you have received)
- Store (to organize all stored messages)
- Delete (to prevent mail from accumulating in your mailbox)
- Print (to produce a paper copy)
- Include or attach (other files in your work)
- Address book or aliases
- Sort your mail

E-Mail Clients

The software that you install on your PC to send and receive E-Mails is called E-Mail client. Once you have an E-Mail address, and a connection to the Internet, all you have to do is choose an E-Mail client and install it on your computer. A large number of commercial and shareware E-Mail packages exist. Windows comes with its own E-Mail client called Microsoft Exchange.

Notes Microsoft Office comes with Microsoft Outlook, a personal information manager that can also send and receive mail.

Full featured E-Mail client are also available as part of the Web browser suites that Netscape communicator and Microsoft Internet Explorer offer. Finally be aware that each of the on-line

Notes

services such as America on-line also provide an E-Mail system for their customers. When choosing your E-Mail package, you may want to keep the following in mind:

- Is the E-Mail client's interface user friendly?
- Does the E-Mail package have an electronic phone book, where you can keep a list of your important E-Mail addresses?
- Does the E-Mail package have the ability to encode and decode files attached to E-Mail messages?
- Does the E-Mail package have a spelling checker?

While all the items listed previously are not vital to make an E-Mail package usable, they are features that many E-Mail packages offers as a matter of course.

If your ISP has a Post Office Protocol (POP) server, you have a wide choice of E-Mail applications. You cannot use most E-Mail programs to get your E-Mail if you use AOL; instead, you use AOL's own program or its Web site. Before you can receive E-Mail, your ISP must know who you are, your client application must be set up, if someone needs to send you E-Mail.

Self Assessment

Fill in the blanks:

12. Windows comes with its own E-Mail client called ……
13. E-Mail programs tend to work alike, even down to the keystrokes required to execute a ……

13.5 Caching

Many visitors access the internet through a proxy. The visitors submit their request through their browser and the request is then sent to a proxy, which serves as a funnel for many computers making requests to the internet. One of the things proxy does is to store a cache of pages requested by all the users of the proxy. So instead of retrieving the same page many times form the internet, the proxy merely returns the cached page to the person making the request.

The cache control property is your way of instructing the proxy to cache or not to cache. The code must precede any HTML and take the following form:

Response.CacheControl= "public"

Or

Response.CacheControl= "private"

By default, the property is to private, which indicates that the content should not be buffered. If you do want the text of you ASP buffered, simply set the property to Private.

HTML pages and graphics can be stored in the cache. Response has two properties that can be used to determine how long an ASP page be cached.

These two properties are:

Response.Expires and

Response.ExpiresAbsolute

Notes

Task Analyze the advantages of caching?

13.5.1 Fragment Caching, User Control Output Caching

Often, caching an entire page is not feasible, because certain parts of the page are customized for the user. However, there may be other parts of the page that are common to the entire application. These are perfect candidates for caching, using fragment caching and user controls. Menus and other layout elements, especially ones that are dynamically generated from a data source, should be cached with this technique. If need be, the cached controls can be configured to vary based on the changes to its controls (or other properties) or any of the other variations supported by page level output caching.

Caution Hundreds of pages using the same controls can also share the cached entries for those controls, rather than keeping separate cached versions for each page.

Implementation

Fragment caching uses the same syntax as page level output caching, but applied to a user control (.ascx file) instead of to a web form (.aspx file). All of the attributes supported by the OutputCache directive on a web form are also supported for user controls except for the Location attribute. User controls also support an OutputCache attribute called VaryByControl, which will vary the caching of the user control depending on the value of a member of that control (typically a control on the page, such as a DropDownList). If VaryByControl is specified, VaryByParam may be omitted. Finally, by default each user control on each page is cached separately. However, if a user control does not vary between pages in an application and is named the same across all such pages, the Shared="true" parameter can be applied to it, which will cause the cached version(s) of the user control to be used by all pages referencing that control.

Example:

```
<%@ OutputCache Duration="60" VaryByParam="*" %>
```

This would cache the user control for 60 seconds, and would create a separate cache entry for every variation of querystring and for every page this control is placed on.

```
<%@ OutputCache Duration="60" VaryByParam="none"
   VaryByControl="CategoryDropDownList" %>
```

This would cache the user control for 60 seconds, and would create a separate cache entry for each different value of the CategoryDropDownList control, and for each page this control is placed on.

```
<%@ OutputCache Duration="60" VaryByParam="none" VaryByCustom="browser"
   Shared="true %>
```

Finally, this would cache the user control for 60 seconds, and would create one cache entry for each browser name and major version. The cache entries for each browser would then be shared by all pages referencing this user control (as long as all pages refer to the control with the same ID).

13.5.2 Page Caching

Page caching is an approach to caching where the entire action output of is stored as a HTML file that the web server can serve without going through Action Pack. This is the fastest way to cache

Notes

your content as opposed to going dynamically through the process of generating the content. Unfortunately, this incredible speed-up is only available to stateless pages where all visitors are treated the same. Content management systems – including weblogs and wikis – have many pages that are a great fit for this approach, but account-based systems where people log in and manipulate their own data are often less likely candidates.

Specifying which actions to cache is done through the caches_page class method:

class WeblogController < ActionController::Base

caches_page :show, :new

end

This will generate cache files such as weblog/show/5.html and weblog/new.html, which match the URLs used to trigger the dynamic generation. This is how the web server is able pick up a cache file when it exists and otherwise let the request pass on to Action Pack to generate it.

Expiration of the cache is handled by deleting the cached file, which results in a lazy regeneration approach where the cache is not restored before another hit is made against it. The API for doing so mimics the options from url_for and friends:

```
class WeblogController < ActionController::Base
  def update
     List.update(params[:list][:id], params[:list])
     expire_page :action => "show", :id => params[:list][:id]
     redirect_to :action => "show", :id => params[:list][:id]
  end
end
```

Notes Additionally, you can expire caches using Sweepers that act on changes in the model to determine when a cache is supposed to be expired.

13.5.3 Data Caching

Data retrieving from a repository can be quite a "heavy" task from a performance point of view, especially when the data repository is located far from the application server (e.g., web service call, RPC call, Remoting, etc.) or some specific data is accessed very often. So, in order to reduce the workload and time for data retrieving, you can use a caching functionality.

There are some rules that you should be aware of:

1. Use caching of data for a small period of time and avoid caching for the whole application lifecycle.
2. Try to cache the data that is likely to not be changed very often (e.g., dictionary elements).
3. Some data repositories can support notification events if the data is modified outside of the application (e.g., the data is just stored in a file).

Besides the obvious goals, data caching has some pitfalls (all of them are about potential situations when cached data can expire and application uses inconsistent data):

1. If the application is going to be scaled to a distributed environment (web farm, application cluster), then every machine will have its own copy of cached data. So, one of the machines can modify the data at any time.
2. Several applications can access the same data repository.

Notes

There are very many different implementations of the caching functionality, but all of them have a lot in common. I will use the "Microsoft Enterprise Library Caching Application Block" and will present you a way of simplifying the usage of the caching functionality.

My implementation of the caching functionality can be divided into two main parts: the cache manager utility and its usage. It heavily uses anonymous methods and generic methods (new features of .NET 2.0).

CacheService is a manager utility that is implemented as a singleton wrapper for underlying the MS caching API. It grants a few methods (GetData<t>,Add, Remove).

Self Assessment

Fill in the blanks:

14. Use caching of data for a small period of time and avoid caching for the whole ……..

15. Expiration of the cache is handled by deleting the …….. file.

Caselet

'E-commerce Proves a Big Boon for Entrepreneurs'

Did you know there's a market for Ramayana in Zambia? Neither did Mr Chinmay Tripathi, seller of religious books online, till a customer from that country placed an order for one of his products on his account on an online shopping portal.

Thanks to distances being bridged by e-commerce, the value of transactions across cyber space in India is estimated at ₹ 1,180 crore in 2004-05, and is projected to hit ₹ 2,300 crore in 2006-07, according to the Internet and Mobile Association of India.

The Internet is a big boon for new age entrepreneurs as this saves them the initial investment in a store and the subsequent management costs. Mr Tripathi is only a part-time programmer, who is busy with his full time business of managing his 120-plus registered accounts in shopping portals worldwide. He sells 200 to 300 religious books a month to customers in places ranging from Haridwar to Djakarta.

A few thousand of eBay's two million users in India make their living through this Web site, according to a spokesperson of eBay. Similarly, on Rediff.com's shopping portal, the business focus of 70-80 per cent of the company's total base of e-commerce partners is purely online, according to Mr Jasmeet Singh, Vice President - Product Marketing, Rediff.com. Mr Singh said e-commerce has grown positively in the last two years. He said there has been an increasing demand for convenience-driven services such as purchase of railway and airline tickets.

Another driving factor is the reduced cost of the products purchased online. The customer can save up to 30 per cent, said Mr Ambesh Khanna, who, along with Ms Reena Khanna, sells diamonds on ebay on a full-time basis. This is a because of the low day-to-day running costs for the seller. "We buy the diamonds locally (The Khannas are based in Delhi), and sell it through the Internet throughout India and also worldwide to countries like the US, Germany and Australia," said Mr Khanna. He said they have sold jewellery priced from ₹ 5,000 to ₹ 1.5 lakh online, generating revenues of about ₹ 4 lakh a month.

E-commerce also serves as a starting point for businesses. Mr Bomi Rupa, as a student in the US in 2003, saw in the Internet an opportunity to import low-end laptops from the US

Contd...

Notes

and sell it in India to make extra pocket money. "A laptop that goes for ₹ 20,000 in the US would easily go for ₹ 35,000 over here, so even including the 14 per cent import duty, the margins were good," he said.

Today, Mr Rupa owns Hightechsolutionss, a consumer electronics store in Mumbai that sells laptops, and is looking to expand his business. Unwilling to abandon his starting point, he still sells laptops on ebay, but only high-end laptops, that would fetch him ₹ 80,000 to ₹ 90,000.

Mr Singh of Rediff said that consumer electronics, which includes mobile phones, audio-video systems and digital cameras, is a large category, as is white goods, while traditional categories such as gifts, apparel, flowers and books continue to grow. He added that premium branded apparel for both men and women is also growing.

To make it easier for these individual sellers and SME entrepreneurs, ebay offers a service called 'eshops', where the seller can get his own url, logo and content, and that link will display only that seller's products, according to the company.

13.6 Summary

- In ASP you can call a JavaScript procedure from a VBScript and vice versa.
- In order to develop complex interactive applications it is important to maintain state, which browsers do with the help of cookies.
- Using cookies by browser is a method of maintaining information locally in the browser and sending it to the browser whenever the user requests the appropriate pages.
- Cookies information is shared between the client and the server.
- The set cookie syntax is Set-cookie: Name= value; EXPIRES=date; PATH=path; DOMAIN=domain; SECURE.
- The FileSystemObject object is used to access the file system on a server.
- The software that you install on your PC to send and receive E-Mails is called E-Mail client. Once you have an E-Mail address, and a connection to the Internet, all you have to do is choose an
- Page caching is an approach to caching where the entire action output of is stored as a HTML file that the web server can serve without going through Action Pack.

13.7 Keywords

Cookies: Cookies are small set of information stored locally by a browser on behalf of a uses to maintain state with server.

Expiry-date: It is the property that intimates the server how long the cookies will remain valid.

Post Office Protocol (POP)

Set-cookies Field: It is an attribute-value list that sets different properties of a cookie.

SSL: Secure Socket Layer.

TCP: Transmission Control Protocol.

13.8 Review Questions

1. In ASP you can call a JavaScript procedure from a VBScript and vice versa. Explain

Notes

2. One of the challenges of writing applications for the World Wide Web has been inability of the web to maintain state. Comment
3. Explain why maintaining state is important to developing complex interactive applications?
4. Did you know why cookies has no special significance? Give reasons
5. ASP is the tool you need if you're looking for a way to pull together HTML pages, script commands, and COM components. Examine
6. Data retrieving from a repository can be quite a "heavy" task from a performance point of view. Analyze
7. The FileSystem property returns the file system in use for a specified drive. Explain this statement with proper example.
8. The cache control property is your way of instructing the proxy to cache or not to cache. Explain
9. Substantiate the pitfalls of Data Caching.
10. Modern email operates across the Internet or other computer networks. Comment

Answers: Self Assessment

Fill in the blanks:

1. ASP Session
2. actual
3. ASP Request Object
4. destroyed
5. NAME=value
6. HTTP
7. Message
8. GetFileName
9. CopyFolder
10. GetAbsolutePathName
11. GetSpecialFolder
12. Microsoft Exchange
13. Command
14. application lifecycle
15. cached

13.9 Further Readings

Books

Bob Reselman, *Active Server Pages 3.0 by Example,* Que Publishing

David Buser, Chris Ullman, Brian Francis, *Dave Sussman, Beginning Active Server Pages 3.0,* John Wiley & Sons

Duane Wessels, *Web caching,* O'Reilly Media

Simon St. Laurent, *Cookies, (Paperback),* Computing Mcgraw-Hill.

Online links

http://en.wikipedia.org/wiki/Email

http://www.mnot.net/cache_docs/

Notes

Unit 14: Database Connectivity

CONTENTS

Objectives
Introduction
14.1 Open and Close a Connection
14.1.1 DSN Less
14.1.2 Closing the Connection
14.2 Reading Records from Database
14.2.1 Reading Records with ADO
14.3 Inserting, Updating and Deleting Database Records
14.3.1 Inserting Records
14.3.2 ASP Database Updating Existing Records
14.3.3 ASP Database — Deleting Information
14.4 Building Database Application using ADO
14.5 Summary
14.6 Keywords
14.7 Review Questions
14.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Recognize opening and closing of connection
- Describe reading from the database
- Demonstrate the inserting, deleting and updating the database records
- Build database application using active objects

Introduction

Inserting Records (Lock Types, Add New and Update); Updating Records, Deleting Records, The real power of data driven Websites comes from the capability to add and change data in your pages. To understand the concept of insertion, deletion and updation we will create a new table called tbluser which will have the fields First Name, Last Name, Email, Username and Password.

14.1 Open and Close a Connection

Once you have designed you database the next step will be to create a DSN less entry, UserDB1. To do this:

- Click on your "Start" Button, and go to Control Panel under Settings.
- Click on "32 ODBC", select "System DSN less"

Notes

- Click "Add" to add a DSN less entry, and then on "Microsoft Access Drive". If "Microsoft Access Driver" does not appear on the list, you possibly have not installed Microsoft Access 7's, 32 bit ODBC drivers.
- At the "Data Source Name" enter "UserDB1", then "Select" the database - you can use "Browse" to select the database users.mdb. With "users" directory created for the application the path to the database c:\inetsrv\wwwroot\users\users.md

Connecting to the Database - Method

This is done to maintain a connect throughout the user's session. The connection is closed when the session ends. This is controlled by the Global.asa file. Each ASP-Based intranet application can have one global.asa file located in the root directory of the application. The global.asa has four events - Application-Start , Session-Start, Application-End and Session-End. The connect to the database for the session is in the Session-Start event , with Session-End being the event used to close the connection.

Caution The language or script being used is VBScript and the ASP is to run on the server.

Connect string in Global.asa

Conn.Open "UserDB1", "userdblogin","userdbpassword"

"Conn" is connect to the database that can be used throughout the session to the DSN less entry UsersDB1 with a login of "userdblogin" and a password of "userdbpassword".

The first post-startup request is made to the web server for any *.asp file in an application causes the Global.asa to be read. So the moment a request is made to any *.asp in the directory in which the intranet application is stored a connection is established with the DSN less UserDB1. Following that the default document, in this case default.asp is processed.

Example: Connecting to database method

```
SUB Session_OnStart
'—— Open ADO connection to database
Conn.Open "UsersDB1", "userdblogin","userdbpassword"
END SUB
```

14.1.1 DSN Less

Requires no server setup, just a carefully constructed connection string as demonstrated below. DSN less connections demand that that you know the name of the file (i.e. file based databases like Access, Paradox, FoxPro, etc.) or the address of the data server (SQLserver, for example). Armed with appropriate information you could open a data source without a DSN! This is faster than a system DSN* since it saves a trip to read the registry each attempt.

To create a DSN-less connection with a Server.MapPath expression:

1. Upload the database file to the remote server.

   Make a note of its virtual path—for example, /jsmith/data/statistics.mdb.

2. Open one of your site's pages in UltraDev and choose Modify > Connections.

   The Connections dialog box appears.

Notes

3 Click New and select Custom Connection String from the pop-up menu.

The Custom Connection String dialog box appears.

4. Enter a name for the new connection.

5. Enter the connection string and use the Server.MapPath method to supply the DBQ parameter.

Example: Suppose the virtual path to your Microsoft Access database is /jsmith/data/statistics.mdb. The connection string can be expressed as follows if you're using VBScript as your scripting language:

```
Driver={Microsoft Access Driver (*.mdb)};DBQ=" & Server.MapPath("/jsmith/data/statistics.mdb")
```

There are several ways to define a connection to a database. A DSN Less Connection is defined by supplying the precise full file path to the physical database:

```
"DRIVER={Microsoft Access Driver (*.mdb)}; _
DBQ=D:\...\x.mdb;UID=admin;UserCommitSync=Yes; _
Threads=3;SafeTransactions=0;PageTimeout=5; _
MaxScanRows=8;MaxBufferSize=2048; _
```

Create a DSN-less Database Connection

The easiest way to connect to a database is to use a DSN-less connection. A DSN-less connection can be used against any Microsoft Access database on your website.

If you have a database called "northwind.mdb" located in a web directory like "c:/webdata/", you can connect to the database with the following ASP code:

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
%>
```

Notes Note, from the example above, that you have to specify the Microsoft Access database driver (Provider) and the physical path to the database on your computer.

Create an ODBC Database Connection

If you have an ODBC database called "northwind" you can connect to the database with the following ASP code:

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "northwind"
%>
```

Notes

With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

An ODBC Connection to an MS Access Database

Here is how to create a connection to a MS Access Database:

1. Open the ODBC icon in your Control Panel.
2. Choose the System DSN-less tab.
3. Click on Add in the System DSN-less tab.
4. Select the Microsoft Access Driver. Click Finish.
5. In the next screen, click Select to locate the database.
6. Click OK.

Notes Note that this configuration has to be done on the computer where your website is located. If you are running Personal Web Server (PWS) or Internet Information Server (IIS) on your own computer, the instructions above will work, but if your web site is located on a remote server, you have to have physical access to that server, or ask your web host to do this for you.

14.1.2 Closing the Connection

When done with the recordset don't forget to close it:

```
Conn.close
set
```

Self Assessment

Fill in the blanks:

1. Click "Add" to add a DSN less entry, and then on "……".
2. DSN less connections demand that that you know the name of the file or the …… of the data server.
3. The first …… request is made to the web server for any *.asp file in an application causes the Global.asa to be read.
4. The easiest way to connect to a database is to use a ……
5. Armed with appropriate information you could open a data source without a ……
6. Enter the connection string and use the Server.MapPath method to supply the ……
7. A DSN Less Connection is defined by supplying the precise full file path to the …… database.

Notes

14.2 Reading Records from Database

Databases and ASP (Communicating with a Database using ActiveX Data Objects (ADO); Connecting to a Database (The Connection Object, Using a System DSN, Using a DSN-less Connection, Opening the Connection, Closing the connection, Properties of the connection); Reading data from a Database (The Recordset Object, Using adovbs.inc, Reading and Displaying the contents of a Database Table).

Internet programming is one of the most exciting new fields to develop in recent years, and will continue growing in importance, especially for Visual Basic programmers. In response, Microsoft is pushing hard to integrate several of its key technologies, in order to allow developers to create Internet solutions that scale well and that are quick to develop. Server-side technologies are where the new advances in Internet technology will likely be made – at least in the near future, which is about as far as we can ever see in today's rapidly changing technological landscape.

Active Server Pages (ASP) is a server-side technology that accomplishes this very feat: an ASP script is a combination of HTML and ASP objects that work together to produce a pure HTML file. The client never sees the script, which resides on the server, they just see the output. ASP is basically a collection of objects that handle all aspects of a Web site, from communicating with the client, to managing multiple connections. ASP is firmly grounded in Visual Basic programming concepts. In fact, Visual Basic is rapidly emerging as a premier language for developing Web solutions, as you will appreciate by the end of this article. My concern here is a DSN-less connection.

A DSN-less connection is operationally identical to a DSN except that the server doesn't have to access the registry to access the parameters since they're all specified in the connection string itself.

Did u know? Which parameter is required by ODBC connection?

The only parameter required by all ODBC connection strings is Driver which specifies which driver to use. Unfortunately this isn't enough to get connected to anything and additional information is required by each driver.

Now we will see how we read a database using ASP(communicating with a database Active X Data Objects(ADO),connecting to a database using a DSN-less connection, its opening ,closing, properties &of course reading the data, displaying it using adovbs.inc.

14.2.1 Reading Records with ADO

Recordset Object

The recordset object provides a relatively simple interface o read records from a database.

ADO was introduced and thrived in a time-frame when connected and (mostly) two-tier applications were moving to a more layered and Web-based architecture. The overall ADO object model design reflects this transition.

The Web forced the adoption of a disconnected model where more and more information is downloaded and cached on the client. Within ADO, you find two souls hidden just under the skin of the Recordset object.

The Recordset can leverage server cursors and do what it is supposed to do being connected to the data source all the time. At the same time, another facet of the same object can work disconnected and build up a correct representation of records in two ways. Either it can fetch

Notes

rows from the data source and then drop the connection, or it can read all the needed information from a client disk file, including an XML file with a fixed schema.

The disconnected facilities of ADO have been bolted on the Recordset object interface, paying careful attention to make connected and disconnected functionalities available through equivalent APIs. As a result, the Recordset became hard to wield, rather overwhelming and with significant housekeeping.

Anyway, in ADO you can get data in three ways:

1. Through full-time connection using server cursors.
2. By the means of a fast, read-only, forward-only mechanism specifically thought to read through and process a set of records.
3. Getting a static snapshot of data, processing them disconnected from the data source and submitting changes at a later time.

Whatever way you work, you always use the Recordset object and choose the operating mode by selecting the proper cursor type and location. (See the ADO documentation for more details.)

An obvious drawback is that no matter how optimized and well-designed the Recordset object is, you load in memory more stuff than you actually need. In addition, it adds an extra layer of code to process input parameters and possibly fixes their values.

Notes Given the overall programming interface of ADO, this is an absolute necessity as well as a performance hit.

ADO.NET simplifies the data reading infrastructure by applying the old "divide et impera" motto. It introduces two new objects that don't have particularly familiar names but expose certainly well-known functionalities. They are the Data Reader and the DataSet.

The Data Reader object is the ADO.NET counterpart of the read-only, forward-only default ADO cursor. The DataSet is a container that can be programmatically filled with static snapshot of data. In this sense, it can be seen as a repository of disconnected recordsets. There's no Recordset component in the ADO.NET object model, but the DataTable object is the .NET double of a disconnected recordset.

In ADO.NET there's no explicit support for server cursors even though, in a future version, you can expect to see ADO.NET to support server cursors when the data provider is SQL Server 7.0 or later.

If you need server cursors tody, just import the ADO type library in your .NET application and code through it. To learn more on this good, bad and ugly, see the previous column, ADO Rocks and Rolls in .NET Applications.

ADO.NET is designed around XML to work seamlessly in highly interoperable and disconnected scenarios. From this point of view, it is certainly more appropriate than ADO for the breed of applications that the majority of people are writing and planning today. It simplifies the coding and optimizes the working of data access components that have Web-based clients.

It behaves well also when you have a Web-enabled database server such as SQL Server 2000 with the XMLhttp support.

Despite the actual syntax differences, the programming philosophy behind ADO and ADO.NET is aligned as much as possible to reduce the learning curve for data access developers and the quantity of new concepts they have to familiarize with.

Notes

Task ADO.NET is designed around XML to work seamlessly in highly interoperable and disconnected scenarios. Analyze what author wants to say from this statement?

Signs of an ADO Presence

More often than not, you can see signs of a supernatural ADO presence in several ADO.NET code snippets. Want an example? Let's consider a piece of ADO code that, although in slightly different flavors, runs buried in the body of thousands of ASP pages and middle-tier components.

```
Set oCN = Server.CreateObject("ADODB.Connection")
oCN.Open strConn
Set oCMD = Server.CreateObject("ADODB.Command")
Set oCMD.ActiveConnection = oCN
Set oRS = oCMD.Execute(strCmd)
```

In this fragment, you explicitly create a Connection and a Command object, and when this executes, you are returned a new Recordset object. The Connection object contains information about the desired cursor type and location. If you choose the adOpenStatic type of cursor, and necessarily the client-side location, you can safely close the connection and walk through the records. Otherwise, you keep the connection open until you finish navigating through the fetched rows.

Once you have the recordset, you scroll it using a loop as follows:

This type of code won't work as-is in ADO.NET. However, the ADO.NET DataReader object lets you write code that, at least functionally speaking, is nearly identical.

First off, you create a special object to govern the execution of the command. The base .NET class is DBCommand. You normally don't use this class; you'd use one of the more specialized .NET classes like ADOCommand and Visual basic or any user-defined derived class. DBCommand represents a command that the data source can understand.

```
While Not oRS.EOF
Response.Write(oRS("lastname") & "<BR>")
oRS.MoveNext
Wend
```

Example: Specific VBScript EXAMPLES and code samples of MS Access visual basic employed to perform Access form functions and operations:

- How to calculate person's age VB scripts
- Set status fields of master record based on value of detail record-VBA programming examples
- Programming visual basic-compare original value to new value of field in the after update event
- Microsoft Access from field validation examples-dates,date ranges and numbers using VBA
- Send Email outlook VBA code example
- Read outlook Email from Microsoft Access

Notes

- Reusable query definitions-pass from field variable to query
- Concatenate multiple records single text field
- Change names/address from upper case to proper case Access VB example.

Self Assessment

Fill in the blanks:

8. The …… can leverage server cursors and do what it is supposed to do being connected to the data source all the time.
9. The …… is a container that can be programmatically filled with static snapshot of data. In this sense, it can be seen as a repository of disconnected recordsets.
10. The Connection object contains information about the desired …… and location.
11. The base .NET class is ……

14.3 Inserting, Updating and Deleting Database Records

14.3.1 Inserting Records

Data driven sites must be capable of editing as well as addition and deletion of data. When we make a database accessible over the WEB, there are many people accessing it at once. Accessing is OK till the people are just reading, when they try to manipulate it, it may cause problems. To prevent this the first person who tries to change the record puts a "LOCK" on it (lock basically means that another person cannot edit). While the lock is on no one can change the records. As soon as the user is finished the lock is removed.

In the current scenario we need not only reading access but also the writing access. The adLockReadOnly is not sufficient. Currently the lock that is put is "adLockOptimistic" (Optimistic locking means that records are locked by the provider when update is called).

AddNew and UpDate

There are two methods of Recordset object that you will need to make changes to the database : AddNew and Update. AddNew creates a new record in the recordset. The new record is not added to the database till the Update method is called. After AddNew is called, the new record becomes the current record, and it remains the current record even after Update is called.

Example:

```
ObjRS.AddNew
ObjRS("Name")= "Nikhil"
ObjRS("Email")= xyz@asd.com
ObjRS.Update
```

In this example AddNew creates the new record and sets it as the current record which helps in assigning the values. Update command add the record to the database.

While we are editing record and call the AddNew function. So the following has the effect of creating one new record, adding it to the table, creating a second new record, and it to the table:

Notes

```
ObjRS.AddNew
ObjRS("Name") = "Nikhil"
ObjRS("Email") = xyz@asd.com
ObjRS.AddNew
ObjRS("Name") = "Aman"
ObjRS("Email") = "asd@xyz.com"
ObjRS.Update
```

Another way of initializing the variables using AddNew is

Syntax :— objRS.AddNew fields, values

Fields and values are either single values or arrays with the same number of elements.

For example objRS.AddNew Array("Name", "Email"), Array("Nikhil", xyz@asd.com)

Adding a Record Using AddNew and Update

```
<%@ Language=VBScript %>
<! - - #include virtual=*/adovbs.inc" —> %>
<! - - #include file=*DatabaseConnect.asp" —>
<HTML>
<BODY>
<%
Dim objRS
Set objRS = Server.CreateObject("ASOSB.Recordset")
objRS.Open "tblUsers",objConn, , AdLockOptimistic, adCmdTable
objRS.AddNew
objRS("FirstName") = "Nikhil"
ObjRS("Email") = xyz@asd.com
objRS("Username") = "aaa"
objRS("Password") = "pwd"
objRS.Update
objRS.MoveFirst
%>
<B> Entries in Table</B>
<P>
<TABLE>
<TR>
<TD> Name </TD>
<TD> Email</TD>
<TD> Username</TD>
```

Notes

```
25.  </TR>
26.  <% Do While Not objRS.EQF %>
27.  <TR>
28.  <TD><%=objRS("LastName) %>, <%=objRS("FirstName") %></TD>
29.  <TD><%=objRS("Email") %></TD>
30.  <TD><%=objRS("Username") %></TD>
31.  </TR>
32.  <% objRS.MoveNext
33.  loop
34.  objRS.Close
35.  Set objRS = Nothing
36.  objConn.Close
37.  Set objConn = Nothing
38.  %>
39.  </Table>
40.  </BODY>
41.  </HTML>
```

Explanation of the above program

Line 2 contains the constants that we require. Line 3 connects it to the database. Line 7 to Line 9 create and open the Recordset object that will be used. As we know that if we want to make changes in the Recordset we cannot use the default lock type. Line 9 specifies the Optimistic Locking. Now we can add new record. Line 10 invokes AddNew, which creates a new empty record. Line 11 till Line 14 set all the field values. Finally, line 15 saves the new record to the table using Update. Line 16 resets the cursor to the beginning of the recordset using Movefirst. This starts from the beginning of the recordset and displaying its contents in the way as stated in the program.

Adding the User to the Table

Lines 8 to Lines 10 check to make sure that all the fields have specified values. If not, a message describing the problem is printed and a link is provided to go back to the previous page (lines 11 to 13).

Lines 15 to 17 open the recordset. Line 18 check to see whether the current record's username id the same as the username entered in the form. If the username are found to be same the message is printed that is given in line 20. Similarly the email is checked and if it is found to be same a message given in line 26 is printed. This means that one registration per email.

Notes If any one of the username or email is found to be same the Boolean variable is set to true (lines 22-30). When this happens the loop is terminated. If both of them are unique then AddNew is called, the field values are set and update is called.

Notes

14.3.2 ASP Database Updating Existing Records

If your database application did not have the ability to update or modify existing database entries, then it would be severely limiting in its potential. For example, imagine having a customer database in which you did not have the ability to change the contact information for a customer. You would not be able to update the customer's mailing address, phone number, or email address. How about an inventory management system that had an inability to update the price or quantity of an item in stock?

For today's mainstream database applications, database maintenance is a necessity. To that end, there are two main methods of updating information utilizing ADO.

Update command is not only used to add a new record but also used to make changes to the existing records. Instead of calling AddNew move to the record you want to change. Set the field value with the assignment operator as done in the earlier programs. When the values are assigned Update method should be called.

objRS.("username") = "Anshul"

objRS("email") = sdf@asd.com

objRS.Update

the first two lines change the values of the username and email fields of the current record. Then, calling Update save those changes to the database.

Some times you need to Cancel the Update done in the previous records. The following program will tell how the cancellation command will work

objRS("username") = "nikhil"

objRS("email") = "sdf@asd.com

objRS.CancelUpdate

CancelUpdate here undo the preceding lines. Mostly the CancelUpdate command is used with IF…. The program basically becomes

objRS("password") = Request("Pass")

objRS("email") = Request("Email")

If objRS ("password") = "" then

objRS.CancelUpdate

else

objRS.Update

End if

This will cancel the changes made to the record if the password is an empty string. Otherwise, the changes are saved using Update.

ASP Database The Update Method in More Detail

Similar to the addnew method, there are two optional parameters that can be used. These are the fieldlist and the valuelist parameters. These parameters work in the same manner as the ones used by the addnew method.

The fieldlist parameter can be a single field name, or an array of field names, or the numeric (ordinal) position of the fields in the new record. For both the single name and array of names, each name must be enclosed within a pair of double quotes.

Notes

The valuelist parameter is a single value or an array of values for the fields that you want to populate in the new record. If the fieldlist parameter is an array, then valuelist must also be an array. Further, the valuelist must have the exact same number of members and be in the same order as the fieldlist.

Example: Consider these examples:

```
<%
set objRS = Server.CreateObject("ADODB.recordset")
RS.open "tbluser", conn
' Method 1
objRS.Update "FirstName", "Rich"
' Method 2
objRS.Update Array("FirstName", "LastName"), Array("Rich","Smith)
' Method 3
myFieldList = Array("FirstName", "LastName")
myValueList = Array("Rich", "Smith")
objRRS.Update myFieldList, myValueList
objRS.close
set objRS = nothing
objconn.close
set objconn = nothing
%>
```

The above examples echo the update method's ability to update information within recordsets quite similarly to the addnew method.

14.3.3 ASP Database — Deleting Information

Using the recordset object, there is a "delete" method to delete the current record. When using the delete method keep in mind that the current record is deleted and is still the current record in the recordset. If you attempt to retrieve values from the record, an error will occur. This deleted record will hold its place in the recordset until you move to another record.

Program for Deleting Records from Database

```
1.  <%@ Language=VBScript %>
2.  <% Option Explicit %>
3.  <!- - #include virtual='/adovbs.inc"- - >
4.  <!- - #include file="DatabaseConnect.asp"- - >
5.  <HTML>
6.  <BODY>
7.  <% Dim objRS
8.  Set objRS = Server.CreateObject("ADODB.Recordset)
9.  objRRS.Open "tblUsers",objConn, adLockOptimistic, adCmdTable %>
```

Notes

```
10. <B>Entries in table</B>
11. <P>
12. <Table>
13. <TR>
14. <TD>Name</TD>
15. <TD>Email</TD>
16. <TD>Username</TD>
17. </TR>
18. <% Do While Not objRS.EQF %>
19. <TR>
20. <TD><%=objRS("LastName") %>, <%=objRS("Firstname") %></TD>
21. <TD><%=objRS("Email") %></TD>
22. <%=objRS("Username") %></TD>
23. </TR>
24. <% objRS.MoveNext
25. Loop
26. objRS.MoveFirst
27. objRS.Delete
28. objRS.MoveFirst  %>
29. </TABLE>
30. <P>
31. <TABLE>
32. <% Do While Not objRS.EQF %>
33. <TR>
34. <TD><%=objRS("LastName") %>, <%=objRS("Firstname") %></TD>
35. <TD><%=objRS("Email") %></TD>
36. <%=objRS("Username") %></TD>
37. </TR>
38. <% objRS.MoveNext
39. Loop
40. objRS.Close
41. Set objRS = Nothing
42. objConn.close
43. Set objConn = Nothing
44. %>
45. </Table>
46. </BODY>
47. </HTML>
```

Lines 10 through 25 first print out the table as it is initially. It then goes to the first record(line 26), deletes it(line 27), and prints out the new table (lines 31 through 39)

Self Assessment

Fill in the blanks:

12. While the lock is on no one can change the ……..
13. If the fieldlist parameter is an array, then valuelist must also be an ……..

14.4 Building Database Application using ADO

ActiveX Data Objects (ADO)

The bridge between the data providers and data consumers is through data sources created using Microsoft ActiveX Data Objects (ADO), which is the primary method in Visual Basic to access data in any data source, both relational and non-relational. For backward compatibility and project maintenance, Remote Data Objects (RDO) and Data Access Objects (DAO) are still supported.

Data Sources and Data Controls

On the client side, several new data sources are available, including the Data Environment, a graphical designer that allows you to quickly create ADO Connections and Commands to access your data. The Data Environment designer provides a dynamic programmatic interface to the data access objects in your project. In addition, the Data Environment provides advanced data shaping services — the ability to create hierarchies of related data, aggregates, and automatic groupings, all without code.

The new ADO Data control is similar to the intrinsic data control and Remote Data control, except that it uses ADO to access data. You can now use an ADO Recordset as a data source for your controls and objects in Visual Basic.

In Visual Basic you can now create your own data sources either as user controls or classes, to encapsulate business rules or proprietary data structures. The class module now features the DataSourceBehavior property and the GetDataMember event, which allow you to configure a class as a data source.

Dynamic Data Binding

The ability to dynamically bind a data source to a data consumer is now possible in Visual Basic. At run time, you can now set the DataSource property of a data consumer (such as the DataGrid control) to a data source (such as the ADO Data control).

Notes This capability, unavailable in previous versions of Visual Basic, allows you to create applications, which can access a multitude of data sources.

Presenting Data to the End User

Visual Basic offers a variety of rich ways to present data to your end users. ADO/OLE DB-based versions of all the data bound controls are included in Visual Basic:

Notes

- The DataList and DataCombo controls are the ADO/OLE DB equivalents of DBList and DBCombo controls.
- The DataGrid is the successor to DBGrid.
- The Chart control is now data bound.
- A new version of the FlexGrid control, called the Hierarchical FlexGrid, supports the hierarchical abilities of the Data Environment.
- The new DataRepeater control functions as a scrolling container of data bound user controls where each control views a single record.

The Data Report is a new ActiveX designer that creates reports from any data source, including the Data Environment. With the Data Report designer, formatted reports can be viewed online, printed, or exported to text or HTML pages.

Self Assessment

Fill in the blanks:

14. The new ADO Data control is similar to the intrinsic data control and …….
15. The ability to dynamically bind a data source to a data consumer is now possible in …….

Caselet

Let's Talk Tech

WITH marauding hordes, Genghis Khan terrorised whole populations. His motto: "It's not enough that I succeed, everyone else must fail." But he is long dead. Yet his philosophy lives on in the software world, writes Karen Southwick in Everyone Else Must Fail, from Crown Business (www.crownbusiness.com). She provides "the unvarnished truth about Oracle and Larry Ellison". The blurb speaks of how, inside Oracle, "Ellison has time and again systematically purged key operating, sales, and marketing people who got too powerful for his comfort." What is his style? "Freewheeling version of capitalism, the kind practised by the nineteenth century robber barons who ran their companies as private fiefdoms." The book raises a question: "Whether Oracle's products and the reliance placed in them by so many are too important to be subject to the whims of one man." Is there a warning "about an ingenious man's tendency to be his own company's worst enemy"?

The introduction notes how he has "come a long way from the college dropout who started at the bottom rung financially and socially." He is "by turns brilliant and intolerant, inspiring and chilling, energetic and disinterested." Ellison is "one of the most intriguing, dominant, and misguided leaders of a major twenty-first-century corporation." Don't forget that more than half of the Fortune 100 cited Oracle as the preferred database vendor, or that Ellison owns nearly one-fourth of Oracle's stock. He is "the ultimate narcissist," as one business psychologist said. "Ellison may be the last of his kind, but he is unforgettable." He complains "about the way the press tears down heroes, comparing the media to lions at the ancient Roman Colosseum." Yet he takes gleeful joy "in creating controversies." The author analyses: "Because of his childhood, Ellison feels vulnerable whenever he feels himself growing dependent on someone else. He can't stand the thought of abandonment, so he abandons other people before they can do it to him."

Contd...

Notes

Ellison has gone over to the dark side of the Silicon Valley infatuation with power and wealth, notes the concluding chapter, titled On the Edge. His world is solipsistic. But don't count him out too soon. "He has been a wildly entertaining performer," finishes the author, but sighs: "How much more he could have been."

Let's talk IT out

What's common between politics and computers? Everybody talks about both and yet unfortunately few understand. So, Mohammed Azam has taken 'a dialogue oriented approach' to IT. His book Computer Literacy Kit, from Eswar Press (www.eswar.com) , is aimed at "providing a wholesome learning experience for the entire family." Being conversational in style, there are many questions throughout the book, and these find answers from the author's many characters. For instance, "Who were the first buyers of personal computers?" Hobbyists, who knew electronics and software, bought the first lot of PCs. "Apple Computers realised that users did not like the idea of messing around with a lot of wires specially with electricity running in them and unveiled a model that was fully built. The users had to merely take it home and connect it to their TV and start work."

Questions often come in torrents: Such as, what is a platter, what is a cylinder, why is the hard disk sealed, how does the read/write assembly work, how is data recorded on magnetic tape, how is the storage of a tape measured, and so forth. Also, there are short poems. "The computer will tell you with a beep or chime/That you pressed the wrong key this time." Or, "The Operating System plays the host/Taking over after the POST." Yet another, "Command and syntax you need not cram/But to run Windows you need plenty of RAM." Try this one on virus: "A virus is actually an intelligent string of bytes/But it is malignant and it sometimes bites/Some rename and some even corrupt a file/Some are a nuisance, harmless and not vile."

The book provides an elaborate glossary with entries such as "a.out: The default name of the executable file produced by the Unix assembler, link editor, and C compiler" and "Daisy chain: The linking of items one after another. In word processing, daisy chain printing means to print documents one after another." To keep the conversation alive, there are illustrations throughout the book. Good read for starters.

Net coding

In the near future, we will be dealing with distributed applications, fragments of which run on different systems, in heterogeneous networks, under different operating systems; and the computer itself would lose its traditional look, and take any shape, from cubic units built into the walls to small devices such as wristwatches. This is the scenario that Sergei Dunaev paints in Advanced Internet Programming Technologies and Applications, from Eswar PressThe book is a guide for developing Net applications and e-com solutions. "Readers learn how to create and use objects such as applets, scriplets, servlets, XML-constructions, JSP, ASP pages and so on," states the back cover. "JavaBeans/CORBA and ActiveX/DCOM are described in detail."

What software developers encounter every day are "two basic technologies," notes the author in the first chapter. One of these is ActiveX/DCOM, used on Intel platforms using Windows OS, while the parallel technology is called JavaBeans/CORBA, which does not depend on either the platform or the OS. DCOM, which is no diploma in commerce, but distributed component object model, also called COM `with a longer wire' because it allows `registration of remote objects'. ActiveX serves a unique purpose — that of providing operations for program components inside composite program containers that include Web browsers and other document viewers. JavaBeans components are "obliged to advertise their characteristics", and the "clearing of these characteristics by other components is called introspection."

Contd...

Notes

Now what is CORBA? "When we say CORBA, we actually mean CORBA/IIOP," that is Common Object Request Broker Architecture/Internet Inter-ORB. This is a technology "meant for distributed information objects that can closely interact with each other within a managing program, which essentially consists of these objects itself." There is lot more in this `advanced' book for the eager beaver.

14.5 Summary

- The connection is closed when the session ends. This is controlled by the Global.asa file.
- Each ASP-Based intranet application can have one global.asa file located in the root directory of the application.
- Databases and ASP (Communicating with a Database using ActiveX Data Objects (ADO); Connecting to a Database (The Connection Object, Using a System DSN, Using a DSN-less Connection, Opening the Connection, Closing the connection, Properties of the connection); Reading data from a Database (The Recordset Object, Using adovbs.inc, Reading and Displaying the contents of a Database Table).
- Data driven sites must be capable of editing as well as addition and deletion of data.
- When we make a database accessible over the WEB, there are many people accessing it at once. Accessing is OK till the people are just reading, when they try to manipulate it, it may cause problems.
- To prevent this the first person who tries to change the record puts a "LOCK" on it (lock basically means that another person cannot edit).
- While the lock is on no one can change the records. As soon as the user is finished the lock is removed.
- The bridge between the data providers and data consumers is through data sources created using Microsoft ActiveX Data Objects (ADO), which is the primary method in Visual Basic to access data in any data source, both relational and non-relational.
- For backward compatibility and project maintenance, Remote Data Objects (RDO) and Data Access Objects (DAO) are still supported.

14.6 Keywords

ADO: ActiveX Data Objects

ASP: Active Server Pages

DAO: Data Access Objects

RDO: Remote Data Objects

14.7 Review Questions

1. Describe the actions performed by the AddNew method of the Recordset object.
2. Substantiate the arguments which AddNew can accept?
3. Why could not you use the default locking type?
4. What does CancelUpdate do? How do you delete a record?
5. Write a page that takes the titles of the links in the table and lists them in a drop-down box. The user may then click one of the titles and then a submit button, and be taken to a page

Notes

where he may edit the title, URL, or description of the specified record. Create a third page to actually save the changes to the database.

6. The connection is closed when the session ends. Explain
7. The global.asa has four events - Application-Start, Session-Start, Application-End and Session-End. Comment
8. Once you have designed you database the next step will be to create a DSN less entry, UserDB1. Explain with a suitable program
9. ADO.NET simplifies the data reading infrastructure by applying the old "divide et impera" motto. Discuss
10. Explain why Data driven sites must be capable of editing as well as addition and deletion of data?

Answers: Self Assessment

Fill in the blanks:

1. Microsoft Access Drive
2. address
3. post-startup
4. DSN-less connection
5. DSN
6. DBQ parameter
7. Physical
8. Recordset
9. DataSet
10. cursor type
11. DBCommand
12. Records
13. Array
14. Remote Data control
15. Visual Basic

14.8 Further Readings

Books

A Keyton Weissinger, *ASP in a Nutshell*, 2nd Edition [ILLUSTRATED] (Paperback), O'Reilly Media; 2 edition (January 1, 2000)

Bob Reselman, Active Server Pages 3.0 by Example, Que Publishing

David Buser, Chris Ullman, Brian Francis, Dave Sussman, *Beginning Active Server Pages 3.0,* John Wiley & Sons

John W. Gosney, *ASP Programming for the Absolute Beginner* (For the Absolute Beginner) (Paperback), Course Technology PTR; 1 edition (July 15, 2002)

Keith Morneau, Jill Batistick, Active server pages, Course Technology

Scott Mitchell, Designing *Active Server pages*, O'Reilly Media

Online links

http://www.takempis.com/adointro.asp

http://en.wikipedia.org/wiki/ActiveX_Data_Objects