



**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed to be University Established Under Section 3 of UGC Act 1956)

**Coimbatore – 641 021.**

(For the candidates admitted from 2017 onwards)

**DEPARTMENT OF COMMERCE (CA)**

Syllabus	Semester 6
<b>17CCU601A</b>	
<b>INTERNET AND WEB DESIGNNING</b>	<b>L T P C</b>
	<b>4 - - 4</b>

**SCOPE :** Internet and Web Design enables the students with the application of concept of Internet and designing of websites. It equips the students with the skill in the application of scripting languages like HTML, Java script and ASP, Net.

**OBJECTIVES:**

- To know the working of Internet, uses of search engines and procedures to develop a web page.
- To make the student expertise in creating web page. And to ensure that the student know the concepts of Internet and design a web page.

**UNIT- I**

**Introduction to Internet:** Internet basics – World wide Web – browser portability – Features of Internet Explorer. Introduction to HTML ; Mark up languages – common Tags – Headers – text styling – Linking – Images – formatting text – special characters – Horizontal rules and Line text – unrecorded lists – nested unordered lists- basic HTML tables – Intermediate HTML tables.

**UNIT- II**

**Formatting** – Basic HTML forms – creating and using Image maps – meta tags – frameset – nested frameset. Introduction to DHTML: cascading Style Sheet – Inline style – creating style sheets with style elements – conflicting styles – Linking external style sheets – Position elements – Backgrounds – element dimension – text flow – Box models.

**UNIT-III**

**Introduction to Java Script** – Operators – Arithmetic Operators – Precedence of Operators – Relational Operators – Control Structures – Assignment Operators – Increment and Decrement Operators – For loops – Switch – Do While – Break – Continues – Arrays – Functions .

**UNIT- IV**

**Active Server Page (ASP):** Introduction – How ASP work – five objects of ASP – Client Side Scripting Vs Server Side Scripting – Server Side ActiveX component – Session Tracking and Cookies – file system objects.

**UNIT-V**

**Introduction to XML :** Introduction – The Syntax of XML – XML Document Structure – Document Type Definitions – Namespaces – XML Schemas – Displaying Raw XML Documents – Displaying XML Documents with CSS – XML Processors.

**Suggested Readings :****Text Books**

1. H.M.Deitel and T.R.Nieto, (2000). *Internet and World Wide Web How to Program*. New Delhi, Pearson education.
2. Ivan Bayross, (2000). *Web enables Commercial application development using HTML, DHTML, Java Script, Perl, CGI*. New Delhi, BPB Publications.

**Reference Books**

1. Robert .W.Sebesta, (2007). *Programming the World Wide Web* [3<sup>rd</sup> Edition]. New Delhi, Pearson Education.
2. Shelly Powers (2008) *Dynamic Web Publishing*. New Delhi, Techmedia.
3. Scot Johnson.(2010). *Using Active Server Pages*. New Delhi. Que Publications.
4. J. Jaworski (2000). *Mastering JavaScript* [2<sup>nd</sup> Edition]. New Delhi, BPB Publications.
5. Thomas Powell. (2002). *HTML Computer Reference*[4th Edition] New Delhi, Tata Mcgraw Hill



**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed to be University Established Under Section 3 of UGC Act 1956)  
**Coimbatore – 641 021.**

**Lecture Plan**

**DEPARTMENT OF COMMERCE (Computer Applications)**

**STAFF NAME: S.Boopathiraja and C.Jothish**

**SUBJECT NAME: Internet and Web Designing**

**SUB CODE: 17CCU601A**

**SEMESTER: VI**

**CLASS: III B. Com CA**

S.NO	LECTURE DURATION PERIOD	TOPICS TO BE COVERED	SUPPORT MATERIAL/PAGE NOS
<b>UNIT-I</b>			
1.	1	Internet basics ,World wide Web	<b>1,2,5</b>
2.	1	browser portability, Features of Internet Explorer	<b>1,2,5</b>
3.	1	Introduction to HTML ; Mark up languages	<b>1,2,5</b>
4.	1	common Tags ,Headers, text styling, Linking Images	<b>1,2,5</b>
5.	1	formatting text, special characters , Horizontal rules and Line text	<b>1,2,5</b>
6.	1	unrecorded lists , nested unordered lists	<b>1,2,5</b>
7.	1	Basic HTML tables, Intermediate HTML tables.	<b>1,2,5</b>
8.	1	Recap of Unit I	
<b>UNIT-II</b>			
1	1	Basic HTML forms	<b>1,2,5</b>
2	1	creating and using Image maps	<b>1,2,5</b>
3	1	meta tags frameset nested frameset	<b>1,2,5</b>
4	1	Introduction to DHTML: cascading Style Sheet	<b>1,2,5</b>
5	1	Inline style creating style sheets with style elements, conflicting styles, Linking external style sheets	<b>1,2,5</b>
6	1	Position elements ,Backgrounds, element dimension	<b>1,2,5</b>
7	1	text flow, Box models.	<b>1,2,5</b>
8	1	Recap of Unit II	
<b>UNIT-III</b>			
1	1	Introduction to Java Script – Operators	<b>4</b>
2	1	Arithmetic Operators, Precedence of Operators, Relational Operators	<b>4</b>
3	1	Control Structures, Assignment Operators – Increment and Decrement Operators	<b>4</b>
4	1	For loops	<b>4</b>
5	1	Switch, Do While, Break,	<b>4</b>

		Continues	
6	1	Arrays	4
7	1	Functions	4
8	1	Recap of Unit III	4
UNIT-VI			
1	1	Active Server Page (ASP): Introduction	3
2	1	How ASP work	3
3	1	five objects of ASP	3
4	1	Client Side Scripting Vs Server Side Scripting	3
5	1	Server Side ActiveX component	3
6	1	Session Tracking and Cookies	3
7	1	File system objects.	3
8	1	Recap of Unit IV	
UNIT-V			
1	1	Introduction to XML : Introduction	2,3
2	1	The Syntax of XML	2,3
3	1	XML Document Structure	2,3
4	1	Document Type Definitions – Namespaces	2,3
5	1	XML Schemas	2,3
6	1	Displaying Raw XML Documents	2,3
7	1	Displaying XML Documents with CSS – XML Processors	2,3
8	1	Recap of Unit V	

#### TEXT BOOKS

1. H.M.Deitel and T.R.Nieto, (2000). *Internet and World Wide Web How to Program*. New Delhi, Pearson education.
2. Ivan Bayross, (2000). *Web enables Commercial application development using HTML, DHTML, Java Script, Perl, CGI*. New Delhi, BPB Publications.

#### REFERENCES

1. Robert .W.Sebesta, (2007). *Programming the World Wide Web* [3rd Edition]. New Delhi, Pearson Education
2. Shelly Powers (2008) *Dynamic Web Publishing*. New Delhi, Techmedia.
3. Scot Johnson.(2010). *Using Active Server Pages*. New Delhi. Que Publications.
4. J. Jaworski (2000). *Mastering JavaScript* [2nd Edition]. New Delhi, BPB Publications.
5. Thomas Powell. (2002). *HTML Computer Reference*[4th Edition] New Delhi, Tata Mcgraw Hill

**UNIT-I****SYLLABUS**

**Introduction to Internet:** Internet basics – World Wide Web – browser portability – Features of Internet Explorer. Introduction to HTML: Mark up languages – Common Tags – Headers – Text Styling – Linking – Images – Formatting Text – Special Characters – Horizontal Rules and Line Text – Unordered List – Nested Unordered lists – Basic HTML Tables – Intermediate HTML Tables.

**1.1 Introduction to Internet**

- Internet can be defined as an electronic medium, which is connecting the world with the help of the computers and it brought the world in a roof called as the Global Village.
- Internet consists of large amount of data that can be accessed by the various users and because of this it is also referred to as the “Information Superhighway” of the world.
- With the help of the internet can easily be in touch with anyone in the whole world by sending electronic mail, by chatting, etc., travel booking can be made very easily, one can order books or buy anything online in simple terms it can be said that internet provides a very strong connection or network between computers globally, bringing people and their working close to each other.
- The network of networks – ‘The Internet’ is not a personal property of any one i.e. it is not owned by anyone, which allows individuals and the various organizations to get connected to any other server or any other user.
- Internet has become such an important and defining tool in today’s competitive and market oriented environment that it helps a lot in getting business and making money.

**Internet History**

The history of the Internet begins with the development of electronic computers in the 1950s. Initial concepts of wide area networking originated in several computer science laboratories in the United States, United Kingdom, and France. The U.S. Department of Defense awarded contracts as early as the 1960s, including for the development of the ARPANET project, directed by Robert Taylor and managed by Lawrence Roberts. The first message was sent over the ARPANET in 1969 from computer science Professor Leonard Kleinrock's laboratory

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

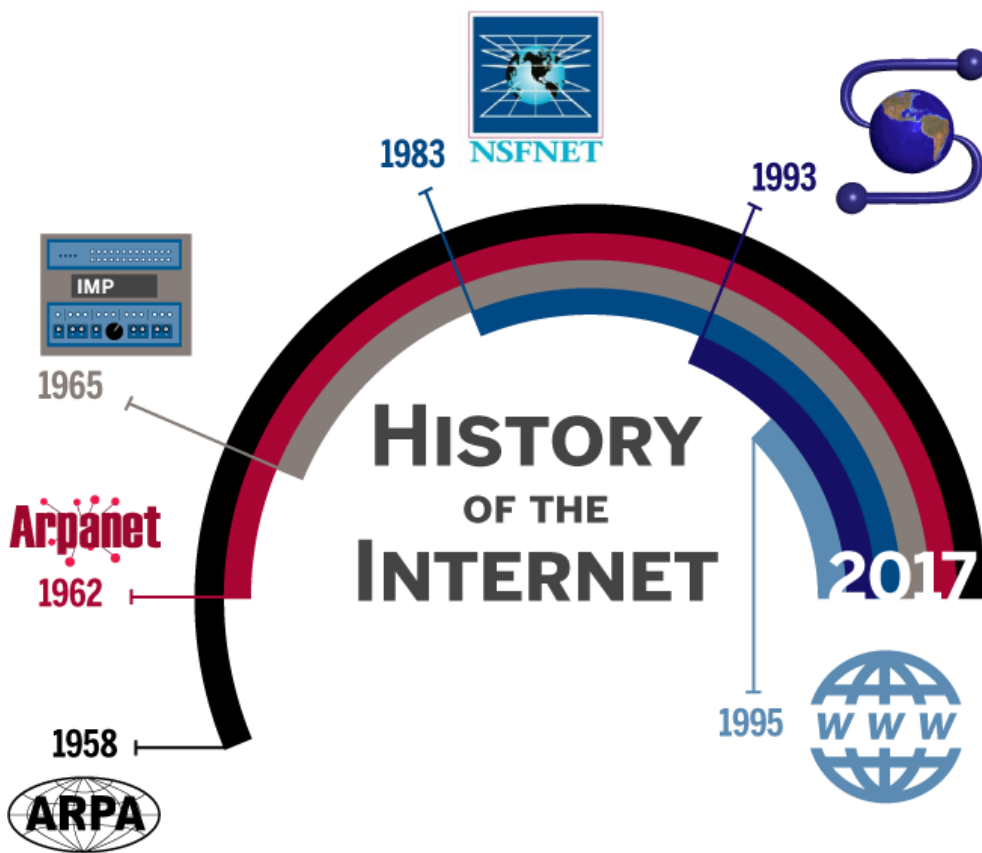
at University of California, Los Angeles (UCLA) to the second network node at Stanford Research Institute (SRI).

Packet switching networks such as the NPL network, ARPANET, Tymnet, Merit Network, CYCLADES, and Telenet, were developed in the late 1960s and early 1970s using a variety of communications protocols. Donald Davies first demonstrated packet switching in 1967 at the National Physics Laboratory (NPL) in the UK, which became a testbed for UK research for almost two decades. The ARPANET project led to the development of protocols for internetworking, in which multiple separate networks could be joined into a network of networks.

The Internet protocol suite (TCP/IP) was developed by Robert E. Kahn and Vint Cerf in the 1970s and became the standard networking protocol on the ARPANET, incorporating concepts from the French CYCLADES project directed by Louis Pouzin. In the early 1980s the NSF funded the establishment for national supercomputing centers at several universities, and provided interconnectivity in 1986 with the NSFNET project, which also created network access to the supercomputer sites in the United States from research and education organizations. Commercial Internet service providers (ISPs) began to emerge in the very late 1980s. The ARPANET was decommissioned in 1990. Limited private connections to parts of the Internet by officially commercial entities emerged in several American cities by late 1989 and 1990, and the NSFNET was decommissioned in 1995, removing the last restrictions on the use of the Internet to carry commercial traffic.

In the 1980s, research at CERN in Switzerland by British computer scientist Tim Berners-Lee resulted in the World Wide Web, linking hypertext documents into an information system, accessible from any node on the network. Since the mid-1990s, the Internet has had a revolutionary impact on culture, commerce, and technology, including the rise of near-instant communication by electronic mail, instant messaging, voice over Internet Protocol (VoIP) telephone calls, two-way interactive video calls, and the World Wide Web with its discussion forums, blogs, social networking, and online shopping sites. The research and education community continues to develop and use advanced networks such as JANET in the United Kingdom and Internet2 in the United States. Increasing amounts of data are transmitted at higher and higher speeds over fiber optic networks operating at 1 Gbit/s, 10 Gbit/s, or more.

The Internet's takeover of the global communication landscape was almost instant in historical terms: it only communicated 1% of the information flowing through two-way telecommunications networks in the year 1993, already 51% by 2000, and more than 97% of the telecommunicated information by 2007. Today the Internet continues to grow, driven by ever greater amounts of online information, commerce, entertainment, and social networking. However, the future of the global internet may be shaped by regional differences in the world.



***Early research and development:***

- 1965: NPL network planning starts
- 1966: Merit Network founded
- 1966: ARPANET planning starts
- 1967: NPL network packet switching pilot experiment
- 1969: ARPANET carries its first packets
- 1970: Network Information Center (NIC)

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

- 1971: Tymnet packet-switched network
- 1972: Merit Network's packet-switched network operational
- 1972: Internet Assigned Numbers Authority (IANA) established
- 1973: CYCLADES network demonstrated
- 1974: Telenet packet-switched network
- 1976: X.25 protocol approved
- 1978: Minitel introduced
- 1979: Internet Activities Board (IAB)
- 1980: USENET news using UUCP
- 1980: Ethernet standard introduced
- 1981: BITNET established

### *Merging the networks and creating the Internet:*

- 1981: Computer Science Network (CSNET)
- 1982: TCP/IP protocol suite formalized
- 1982: Simple Mail Transfer Protocol (SMTP)
- 1983: Domain Name System(DNS)
- 1983: MILNET split off from ARPANET
- 1985: First .COM domain name registered
- 1986: NSFNET with 56 kbit/s links
- 1986: Internet Engineering Task Force (IETF)
- 1987: UUNET founded
- 1988: NSFNET upgraded to 1.5 Mbit/s (T1)
- 1988: OSI Reference Model released
- 1988: Morris worm
- 1989: Border Gateway Protocol (BGP)
- 1989: PSINet founded, allows commercial traffic
- 1989: Federal Internet Exchanges (FIXes)
- 1990: GOSIP (without TCP/IP)
- 1990: ARPANET decommissioned



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

- 1990: Advanced Network and Services (ANS)
- 1990: UUNET/Altnet allows commercial traffic
- 1990: Archie search engine
- 1991: Wide area information server (WAIS)
- 1991: Gopher
- 1991: Commercial Internet eXchange (CIX)
- 1991: ANS CO+RE allows commercial traffic
- 1991: World Wide Web(WWW)
- 1992: NSFNET upgraded to 45 Mbit/s (T3)
- 1992: Internet Society(ISOC) established
- 1993: Classless Inter-Domain Routing (CIDR)
- 1993: InterNIC established
- 1993: AOL added USENETaccess
- 1993: Mosaic web browserreleased
- 1994: Full text web search engines
- 1994: North American Network Operators' Group(NANOG) established

### ***Commercialization, privatization, broader access leads to the modern Internet:***

- 1995: New Internet architecture with commercial ISPs connected at NAPs
- 1995: NSFNETdecommissioned
- 1995: GOSIP updated to allow TCP/IP
- 1995: very high-speed Backbone Network Service(vBNS)
- 1995: IPv6 proposed
- 1996: AOL changes pricing model from hourly to monthly
- 1998: Internet Corporation for Assigned Names and Numbers (ICANN)
- 1999: IEEE 802.11bwireless networking
- 1999: Internet2/Abilene Network
- 1999: vBNS+ allows broader access
- 2000: Dot-com bubblebursts
- 2001: New top-level domain names activated

- 2001: Code Red I, Code Red II, and Nimda worms
- 2003: UN World Summit on the Information Society (WSIS) phase I
- 2003: National LambdaRailfounded
- 2004: UN Working Group on Internet Governance (WGIG)
- 2005: UN WSIS phase II
- 2006: First meeting of the Internet Governance Forum
- 2010: First internationalized country code top-level domains registered
- 2012: ICANN begins accepting applications for new generic top-level domain names
- 2013: Montevideo Statement on the Future of Internet Cooperation
- 2014: NetMundial international Internet governance proposal
- 2016: ICANN contract with U.S. Dept. of Commerce ends, IANA oversight passes to the global Internet community on October 1st

### 1.1.1 : Internet Basics

Internet has been the most useful technology of the modern times which helps us not only in our daily lives, but also our personal and professional lives developments. The internet helps us achieve this in several different ways.

For the students and educational purposes the internet is widely used to gather information so as to do the research or add to the knowledge of various subjects. Even the business professionals and the professionals like doctors, access the internet to filter the necessary information for their use. The internet is therefore the largest encyclopedia for everyone, in all age categories. The internet has served to be more useful in maintaining contacts with friends and relatives who live abroad permanently.

The **Internet** is a global system of interconnected computer networks that use the standard Internet protocol suite (TCP/ IP) to serve billions of users worldwide. It is a *network of networks* that consists of millions of private, public, academic, business, and government networks, of local to global scope, that are linked by a broad array of electronic, wireless and optical networking technologies. The Internet carries a vast range of information resources and services, such as the inter- linked hypertext documents of the World Wide Web

(WWW) and the infrastructure to support electronic mail.

**Advantages of Internet:**

- **E-mail:** Email is now an essential communication tools in business. With e-mail you can send and receive instant electronic messages, which works like writing letters. Your messages are delivered instantly to people anywhere in the world, unlike traditional mail that takes a lot of time. Email is free, fast and very cheap when compared to telephone, fax and postal services.
- **24 hours a day - 7 days a week:** Internet is available, 24x7 days for usage.
- **Information:** Information is probably the biggest advantage internet is offering. There is a huge amount of information available on the internet for just about every subject, ranging from government law and services, trade fairs and conferences, market information, new ideas and technical support. You can almost find any type of data on almost any kind of subject that you are looking for by using search engines like google, yahoo, msn, etc.
- **Online Chat:** You can access many 'chat rooms' on the web that can be used to meet new people, make new friends, as well as to stay in touch with old friends. You can chat in MSN and yahoo websites.
- **Services:** Many services are provided on the internet like net banking, job searching, purchasing tickets, hotel reservations, guidance services on array of topics engulfing every aspect of life.
- **Communities:** Communities of all types have sprung up on the internet. Its a great way to meet up with people of similar interest and discuss common issues.
- **E-commerce:** Along with getting information on the Internet, you can also shop online. There are many online stores and sites that can be used to look for products as well as buy them using your credit card. You do not need to leave your house and can do all your shopping from the convenience of your home. It has got a real amazing and wide range of products from household needs, electronics to entertainment.
- **Entertainment:** Internet provides facility to access wide range of Audio/Video songs,

plays films. Many of which can be downloaded. One such popular website is YouTube.

- **Software Downloads:** You can freely download innumerable, softwares like utilities, games, music, videos, movies, etc from the Internet.

### **Limitations of Internet**

- **Theft of Personal information:** Electronic messages sent over the Internet can be easily snooped and tracked, revealing who is talking to whom and what they are talking about. If you use the Internet, your personal information such as your name, address, credit card, bank details and other information can be accessed by unauthorized persons. If you use a credit card or internet banking for online shopping, then your details can also be 'stolen'.

**Negative effects on family communication:** It is generally observed that due to more time spent on Internet, there is a decrease in communication and feeling of togetherness among the family members.

- **Internet addiction:** There is some controversy over whether it is possible to actually be addicted to the Internet or not. Some researchers, claim that it is simply people trying to escape their problems in an online world.
- **Children using the Internet** has become a big concern. Most parents do not realize the dangers involved when their children log onto the Internet. When children talk to others online, they do not realize they could actually be talking to a harmful person. Moreover, pornography is also a very serious issue concerning the Internet, especially when it comes to young children. There are thousands of pornographic sites on the Internet that can be easily found and can be a detriment to letting children use the Internet.
- **Virus threat:** Today, not only are humans getting viruses, but computers are also. Computers are mainly getting these viruses from the Internet. Virus is a program which disrupts the normal functioning of your computer systems. Computers attached to internet are more prone to virus attacks and they can end up into crashing your whole hard disk.
- **Spamming:** It is often viewed as the act of sending unsolicited email. This multiple or vast emailing is often compared to mass junk mailings. It needlessly obstruct the entire

system. Most spam is commercial advertising, often for dubious products, get-rich-quick schemes, or quasi-legal services. Spam costs the sender very little to send — most of the costs are paid for by the recipient or the carriers rather than by the sender

### **1.1.2 : World Wide Web**

WWW stands for World Wide Web. A technical definition of the World Wide Web is : all the resources and users on the Internet that are using the Hypertext Transfer Protocol HTTP.

A broader definition comes from the organization that Web inventor Tim Berners-Lee helped found, the World Wide Web Consortium W3C.

The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge.

In simple terms, The World Wide Web is a way of exchanging information between computers on the Internet, tying them together into a vast collection of interactive multimedia resources.

Internet and Web is not the same thing: Web uses internet to pass over the information.

#### **WWW Operation**

WWW works on client- server approach. Following steps explain how the web works:

1. User enters the URL (say, <http://www.tutorialspoint.com>) of the web page in the address bar of web browser.
2. Then browser requests the Domain Name Server for the IP address corresponding to [www.tutorialspoint.com](http://www.tutorialspoint.com).
3. After receiving IP address, browser sends the request for web page to the web server using HTTP protocol which specifies the way the browser and web server communicate.
4. Then web server receives request using HTTP protocol and checks its search for the requested web page. If found it returns it back to the web browser and close the HTTP connection.
5. Now the web browser receives the web page, It interprets it and display the contents of web page in web browser's window.

**Protocols used in Internet**

There are various protocols used in the internet for data transfer within web browser and web server. The protocols used in internet are as follows

1. TCP/IP Protocol
2. UDP Protocol
3. FTP Protocol
4. Telnet Protol
5. HTTP Protocol

**1. Transmission Control Protocol TCP**

TCP is a connection oriented protocol and offers end-to-end packet delivery. It acts as backbone for connection. It exhibits the following key features:

- Transmission Control Protocol TCP corresponds to the Transport Layer of OSI Model.
- TCP is a reliable and connection oriented protocol.
- TCP offers:
  - Stream Data Transfer.
  - Reliability.
  - Efficient Flow Control
  - Full-duplex operation.
  - Multiplexing.
- TCP offers connection oriented end-to-end packet delivery.
- TCP ensures reliability by sequencing bytes with a forwarding acknowledgement number that indicates to the destination the next byte the source expects to receive.
- It retransmits the bytes not acknowledged within a specified time period.

**TCP Services**

TCP offers following services to the processes at the application layer:

- Stream Delivery Service
- Sending and Receiving Buffers
- Bytes and Segments
- Full Duplex Service

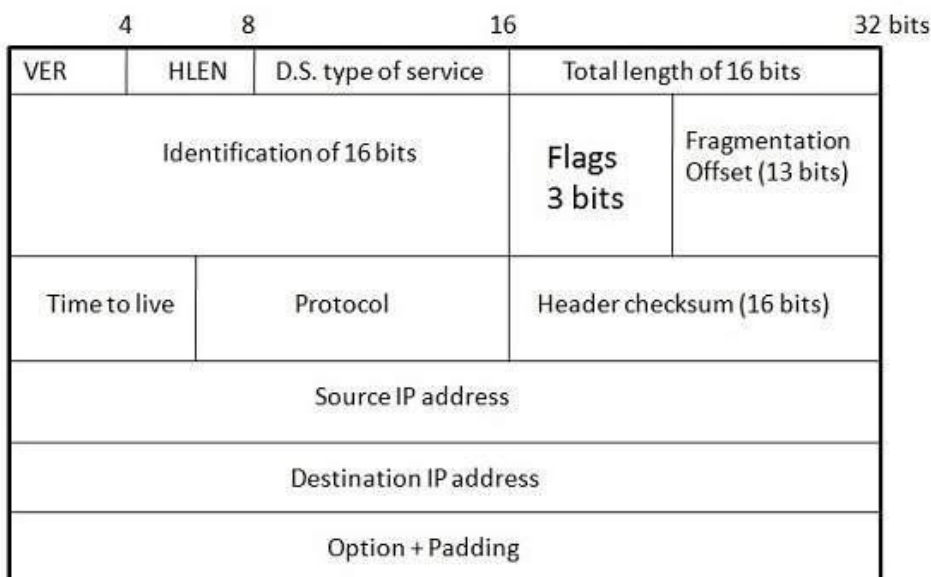
- Connection Oriented Service
- Reliable Service

## **2. Internet Protocol IPIP**

Internet Protocol is **connectionless** and **unreliable** protocol. It ensures no guarantee of successfully transmission of data.

In order to make it reliable, it must be paired with reliable protocol such as TCP at the transport layer.

Internet protocol transmits the data in form of a datagram as shown in the following diagram:



## **3. User Datagram Protocol UDP**

Like IP, UDP is connectionless and unreliable protocol. It doesn't require making a connection with the host to exchange data. Since UDP is unreliable protocol, there is no mechanism for ensuring that data sent is received.

UDP transmits the data in form of a datagram. The UDP datagram consists of five parts as shown in the following diagram:

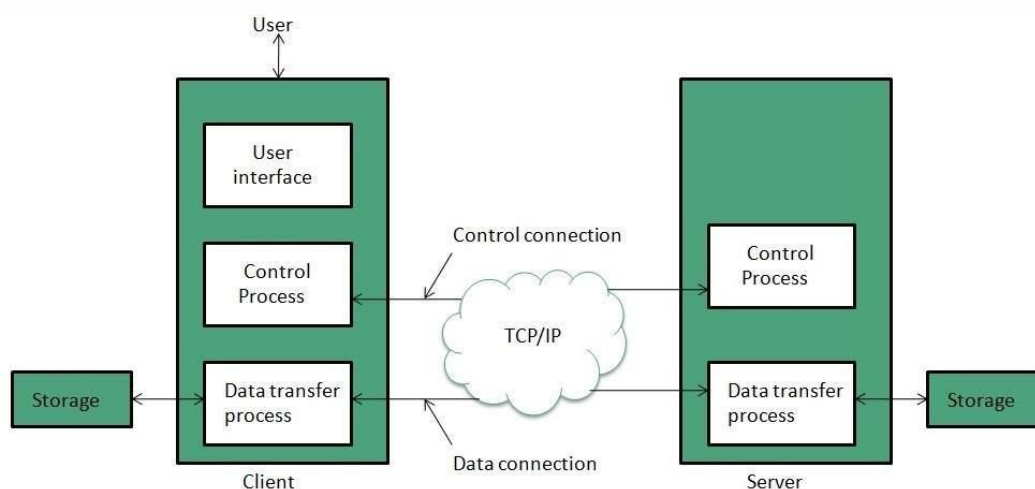
Source Port	Destination Port
Length	UDP checksum
Data	

- UDP is used by the application that typically transmit small amount of data at one time.
- UDP provides protocol port used i.e. UDP message contains both source and destination port number, that makes it possible for UDP software at the destination to deliver the message to correct application program.

#### **4. File Transfer Protocol FTP**

FTP is used to copy files from one host to another. FTP offers the mechanism for the same in following manner:

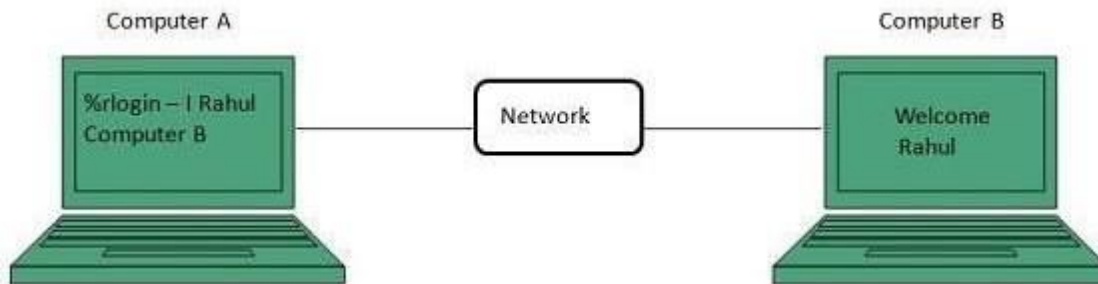
- FTP creates two processes such as Control Process and Data Transfer Process at both ends i.e. at client as well as at server.
- FTP establishes two different connections: one is for data transfer and other is for control information.
- **Control connection** is made between **control processes** while **Data Connection** is made between
- FTP uses **port 21** for the control connection and **Port 20** for the data connection.





### **5.Telnet**

Telnet is a protocol used to log in to remote computer on the internet. There are a number of Telnet clients having user friendly user interface. The following diagram shows a person is logged in to computer A, and from there, he remote logged into computer B.



### **6.Hyper Text Transfer Protocol HTTP**

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

#### **HTTP Request**

HTTP request comprises of lines which contains:

- Request line
- Header Fields
- Message body

#### **Key Points**

- The first line i.e. the **Request line** specifies the request method i.e. **Get** or **Post**.
- The second line specifies the header which indicates the domain name of the server from where index.htm is retrieved.

#### **HTTP Response**

Like HTTP request, HTTP response also has certain structure. HTTP response contains:

- Status line
- Headers
- Message body

### **1.1.3 : Browser Portability**

A web browser (commonly referred to as a browser) is a software application for accessing information on the World Wide Web. Each individual web page, image, and video is identified by a distinct URL, enabling browsers to retrieve and display them on the user's device.

A web browser is not the same thing as a search engine, though the two are often confused. For a user, a search engine is just a website, such as google.com, that stores searchable data about other websites. But to connect to and display websites on their device, a user needs to have a web browser installed.

The most popular web browsers are Chrome, Firefox, Safari, Internet Explorer, and Edge.

Ensuring a consistent look and feel on client-side browsers is one of the great challenges of developing web-based applications. Currently, a standard does not exist to which software developers must adhere when creating web browsers. Although browsers share a common set of features, each browser might render pages differently. Browsers are available in many versions and on many different platforms (Microsoft Windows, Apple Macintosh, Linux, UNIX, etc.). Vendors add features to each new version that sometimes result in cross-platform incompatibility issues. Clearly it is difficult to develop web pages that render correctly on all versions of each browser. In this book we develop web applications that execute on both the Internet Explorer 7 and Firefox 2 browsers.

### **1.1.4 : Features of Internet Explorer**

Even though not many websites cover *Internet Explorer* extensively and write interesting articles and tutorials about it, this browser still manages to hold the title of the most used browser. One of the reasons for it is that it provides some interesting functionality that other browsers don't have or have not fully adopted. Here is a list of 5 great features *Internet Explorer* has and other browser do not.

- **Visual Search Providers**

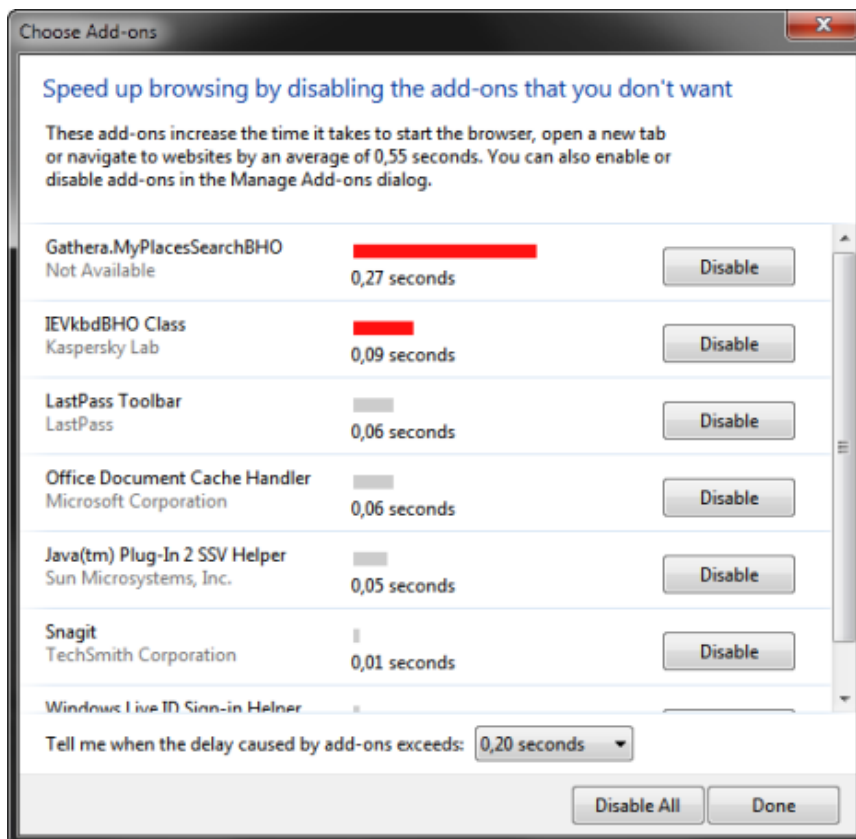
One of the features in *Internet Explorer* is the ability to install and use visual search providers. They allow you not only to see suggestions for your searches but they are also accompanied by a visual preview that includes a picture and some text description.



Very useful and looks great. This feature while doing the research for Google and Other 7 Search Providers for Internet Explorer 9. Unfortunately though, this feature is not used by many search providers and I would like to see more of it both in *Internet Explorer* and other browsers.

- **Stats about the Performance Impact of Add-ons**

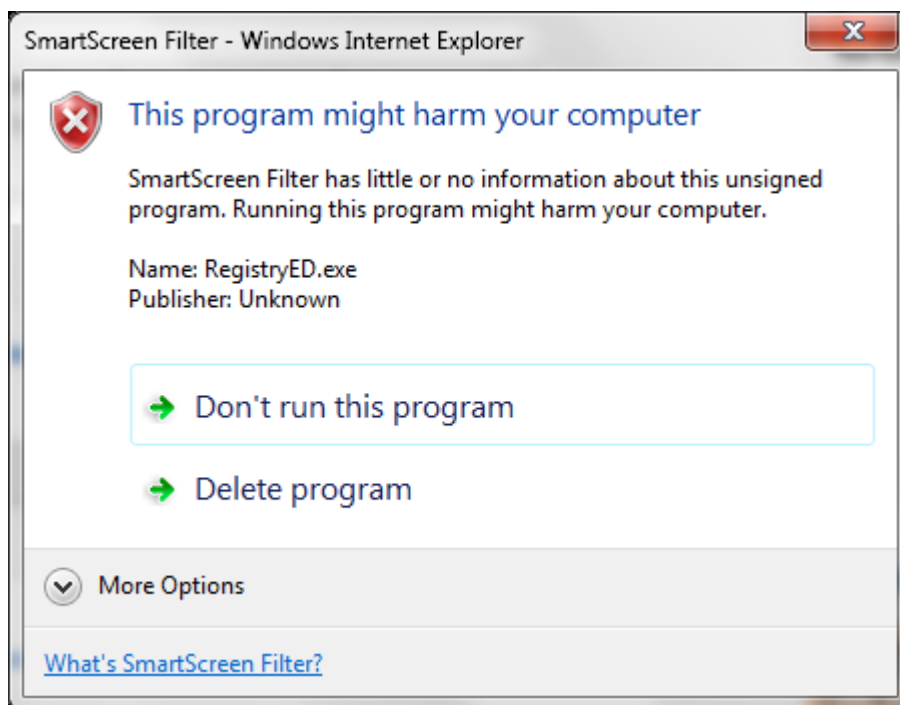
As highlighted in the article Internet Explorer 9 - How to Improve the Load & Navigation Speed, the browser monitors the performance impact of all add-ons and encourages users to disable those that slow things down.



More browsers would have a similar behavior and share in a transparent way how much each add-on contributes to slowing the browsing experience.

- **Improved Security**

If you follow the tech news on a regular basis, you definitely learned about *Internet Explorer* being declared the most secure browser or, at least an important contender to the title. The last debate, found here: [Browser Study Sheds Light On Firefox's Insecurity \(And Google Approves This Message\)](#), considers Google Chrome and *Internet Explorer* as being the top two contenders for the title.

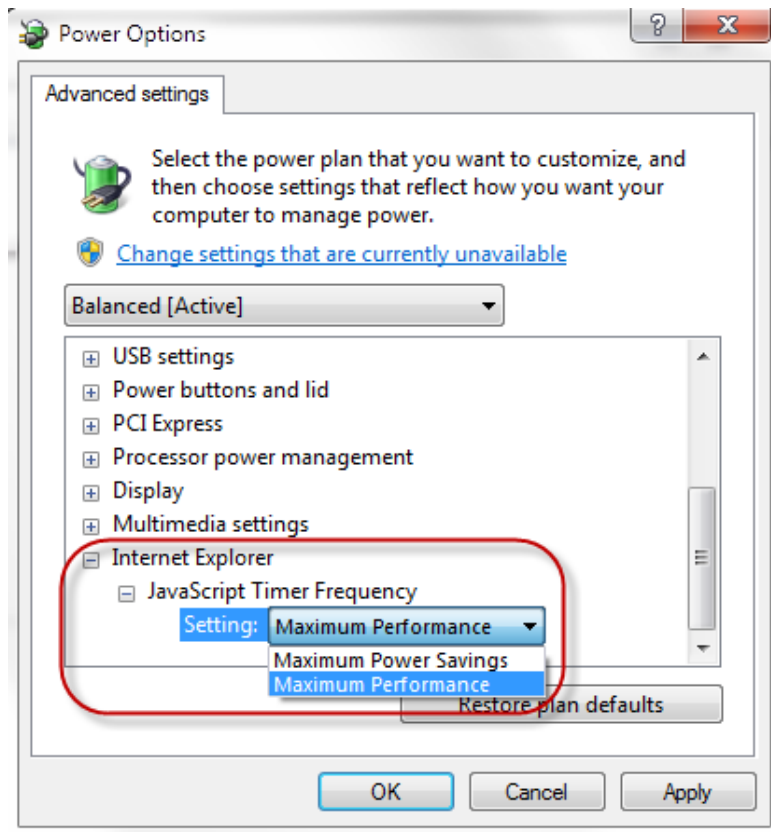


Leaving this debate aside, during the tests we made for the Security for Everyone series, we noticed that *Internet Explorer* definitely provides better identification and reporting of malicious websites and downloads, when compared to other browsers and even some security suites. Another great feature is that *Internet Explorer* allows users to easily report malicious websites, directly from the main interface of the browser, thus helping increase security for all users.

We wish to see more improvements in this area from all major browsers, including *Internet Explorer*.

- **Power Saving Features**

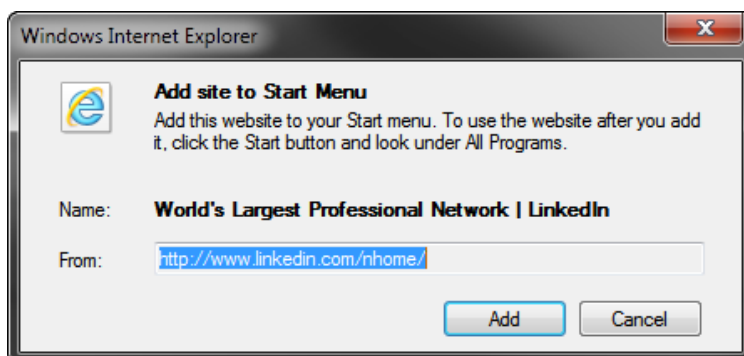
You might not be aware of this, but *Internet Explorer* allows you to squeeze a bit more juice from your battery, when running unplugged. You get less hardware acceleration while browsing, without making things terribly slow and a bit more battery time.



Here is how it works: Save Battery Power While Browsing the Web in Internet Explorer 9.

- **Pinning Websites to the Windows Taskbar or the Start Menu**

This feature is not new and it has been adopted by some browsers, such as Google Chrome. However, I consider it useful and I think all browsers should offer such capabilities to their users.



### 1.1.5 : Introduction to HTML

- HTML is the standard markup language used for creating Web pages.
- HTML stands for Hyper Text Markup Language.
- HTML describes the structure of Web pages using markup.
- HTML elements are the building blocks of HTML pages.
- HTML elements are represented by tags.
- HTML programs are developed in word pad or note pad and the code is executed in any browsers.
- HTML programs are portable and are platform independent.
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on.
- Browsers do not display the HTML tags, but use them to render the content of the page.

#### Sample HTML Program

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

#### Program Explanation

<! \_\_\_\_ > : Comment line

<html> \_\_\_\_ </html>: the element is the root element of an HTML page

<head> \_\_\_\_ </head>: the element contains meta information about the document

<title> \_\_\_\_ </title>: the element specifies a title for the document

<body> element contains the visible page content

The <h1> element defines a large heading

The <p> element defines a paragraph

### 1.1.6 : Markup languages

A markup language is a computer language that uses tags to define elements within a document. It is human-readable, meaning markup files contain standard words, rather than typical programming syntax. While several markup languages exist, the two most popular are HTML and XML.

HTML is a markup language used for creating webpages. The contents of each webpage are defined by HTML tags. Basic page tags, such as <head>, <body>, and <div> define sections of the page, while tags such as <table>, <form>, <image>, and <a> define elements within the page. Most elements require a beginning and end tag, with the content placed between the tags. The beginning and end tags acts as an container for the content placed between the tags.

For example,

```
<a href="http://www.techterms.com">TechTerms.com</a>
```

### Common Tags

#### Basic HTML tags

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Defines information about the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment

#### Formatting tags

Tag	Description
<acronym>	Not supported in HTML5. Use <abbr> instead. Defines an acronym
<abbr>	Defines an abbreviation or an acronym
<address>	Defines contact information for the author/owner of a document/article
<b>	Defines bold text
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<bdo>	Overrides the current text direction
<big>	Not supported in HTML5. Use CSS instead. Defines big text
<blockquote>	Defines a section that is quoted from another source
<center>	Not supported in HTML5. Use CSS instead. Defines centered text
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<del>	Defines text that has been deleted from a document

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

<dfn>	Represents the defining instance of a term
<em>	Defines emphasized text
<font>	Not supported in HTML5. Use CSS instead. Defines font, color, and size for text
<i>	Defines a part of text in an alternate voice or mood
<ins>	Defines a text that has been inserted into a document
<kbd>	Defines keyboard input
<mark>	Defines marked/highlighted text
<meter>	Defines a scalar measurement within a known range (a gauge)
<pre>	Defines preformatted text
<progress>	Represents the progress of a task
<q>	Defines a short quotation
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<s>	Defines text that is no longer correct
<samp>	Defines sample output from a computer program
<small>	Defines smaller text
<strike>	Not supported in HTML5. Use <del> or <s> instead. Defines strikethrough text
<strong>	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<template>	Defines a template
<time>	Defines a date/time
<tt>	Not supported in HTML5. Use CSS instead. Defines teletype text
<u>	Defines text that should be stylistically different from normal text
<var>	Defines a variable
<wbr>	Defines a possible line-break

### Forms and Input tags

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<button>	Defines a clickable button
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

### Frames tags

Tag	Description
<frame>	Not supported in HTML5. Defines a window (a frame) in a frameset
<frameset>	Not supported in HTML5. Defines a set of frames



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

<noframes>	Not supported in HTML5. Defines an alternate content for users that do not support frames
<iframe>	Defines an inline frame

### Images tags

Tag	Description
<img>	Defines an image
<map>	Defines a client-side image-map
<area>	Defines an area inside an image-map
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content
<picture>	Defines a container for multiple image resources
<svg>	Defines a container for SVG graphics

### Audio / Video tags

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements (<video>, <audio> and picture)
<track>	Defines text tracks for media elements (<video> and <audio>)
<video>	Defines a video or movie

### Links tags

Tag	Description
<a>	Defines a hyperlink
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<nav>	Defines navigation links

### Lists tags

Tag	Description
<ul>	Defines an unordered list
<ol>	Defines an ordered list
<li>	Defines a list item
<dir>	Not supported in HTML5. Use <ul> instead. Defines a directory list
<dl>	Defines a description list
<dt>	Defines a term/name in a description list
<dd>	Defines a description of a term/name in a description list

### Tables tags

Tag	Description
<table>	Defines a table
<caption>	Defines a table caption
<th>	Defines a header cell in a table

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

<tr>	Defines a row in a table
<td>	Defines a cell in a table
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table
<col>	Specifies column properties for each column within a <colgroup> element
<colgroup>	Specifies a group of one or more columns in a table for formatting

### Styles and Semantics

Tag	Description
<style>	Defines style information for a document
<div>	Defines a section in a document
<span>	Defines a section in a document
<header>	Defines a header for a document or section
<footer>	Defines a footer for a document or section
<main>	Specifies the main content of a document
<section>	Defines a section in a document
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<summary>	Defines a visible heading for a <details> element
<data>	Links the given content with a machine-readable translation

### Meta Info tags

Tag	Description
<head>	Defines information about the document
<meta>	Defines metadata about an HTML document
<base>	Specifies the base URL/target for all relative URLs in a document
<basefont>	Not supported in HTML5. Use CSS instead. Specifies a default color, size, and font for all text in a document

### Programming tags

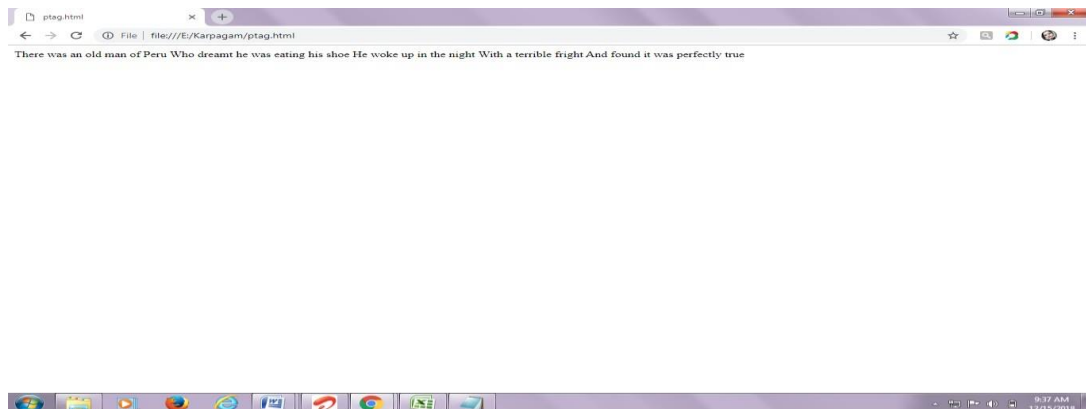
Tag	Description
<script>	Defines a client-side script
<noscript>	Defines an alternate content for users that do not support client-side scripts
<applet>	Not supported in HTML5. Use <embed> or <object> instead. Defines an embedded applet
<embed>	Defines a container for an external (non-HTML) application
<object>	Defines an embedded object
<param>	Defines a parameter for an object

### <Pre> Tag

The PRE element is used for *pre-formatted text*. What this means is that the text between the opening and closing tags is displayed exactly as shown in the HTML file. Normally, any line breaks etc. in the HTML file are ignored when the page is displayed: instead the browser breaks where it finds a <BR>, starts a new paragraph at a <P>, etc. One main use for <PRE> is when you have a large chunk of text in a plain text file which you want to display without having to convert all the paragraphs and tables to HTML formatting. Example:

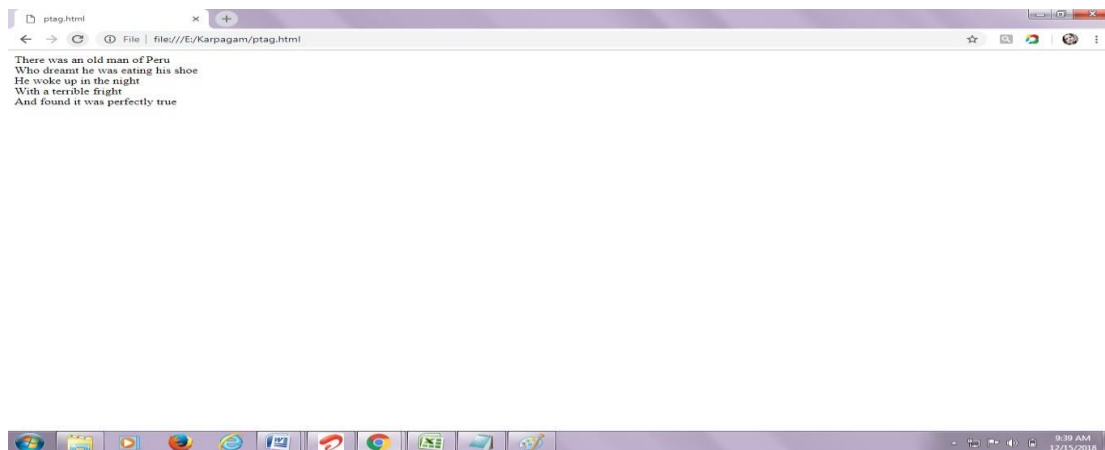
Consider the following code:

```
<P>There was an old man of Peru  
Who dreamt he was eating his shoe  
    He woke up in the night  
    With a terrible fright  
And found it was perfectly true</P>
```



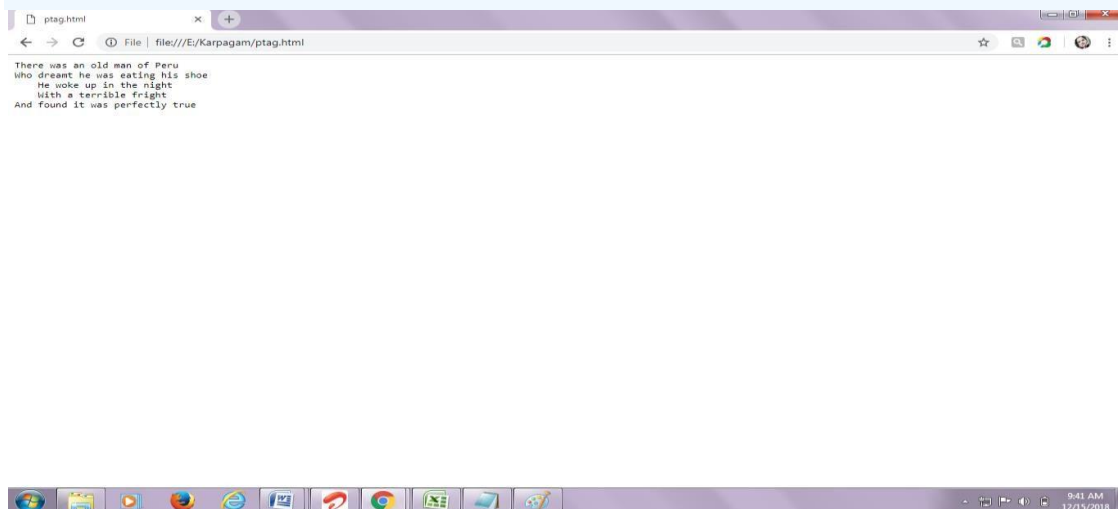
The spacing and line breaks in the HTML code are ignored. To get the right result we would normally use formatting tags:

```
<P>There was an old man of Peru<BR> Who dreamt he was eating his shoe<BR> He woke  
up in the night<BR> With a terrible fright<BR> And found it was perfectly true</P>
```



However, we can use the <PRE> element:

```
<PRE>There was an old man of Peru
Who dreamt he was eating his shoe
    He woke up in the night
    With a terrible fright
And found it was perfectly true</PRE>
```



## 1.1.7 : Text Styling

The color property defines the text color for an HTML element:

### Example

```
<h1 style="color:blue;">This is a heading</h1>
```

```
<p style="color:red;">This is a paragraph.</p>
```

**This is a heading**

This is a paragraph.

### **HTML Fonts**

The font-family property defines the font to be used for an HTML element:

#### **Example**

```
<h1 style="font-family:verdana;">This is a heading</h1>
```

```
<p style="font-family:courier;">This is a paragraph.</p>
```

**This is a heading**

This is a paragraph.

### **HTML Text Size**

The font-size property defines the text size for an HTML element:

#### **Example**

```
<h1 style="font-size:300%;">This is a heading</h1>
```

```
<p style="font-size:160%;">This is a paragraph.</p>
```

# **This is a heading**

This is a paragraph.

### **HTML Text Alignment**

The text-align property defines the horizontal text alignment for an HTML element:

#### **Example**

```
<h1 style="text-align:center;">Centered Heading</h1>
```

```
<p style="text-align:center;">Centered paragraph.</p>
```

**Centered Heading**

Centered paragraph.

### 1.1.8 : Linking

The HTML <link> tag is used for defining a link to an external document. It is placed in the <head> section of the document.

#### **Example**

```
<html>
<head>
  <title>HTML link Tag</title>
  <link rel = "stylesheet" href = "stylenew.css">
</head>
<body>
  <div id = "contentinfo">
    <p>Welcome to our website. We provide tutorials on various subjects.</p>
  </div>
</body>
</html>
```

Here is the css file *stylenew.css*

```
#contentinfo p {
  line-height: 20px;
  margin: 30px;
  padding-bottom: 20px;
  text-align: justify;
  width: 140px;
  color: red;
}
```

This will produce the following result –

Welcome to our website. We provide tutorials on various subjects.

#### **Attributes**

The HTML <link> tag also supports the following additional attributes –

Attribute	Value	Description
Charset	charset	Defines the character encoding of the linked document.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: III BCom CA


COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

Href	URL	Specifies the URL of the resource document.
Hreflang	language	Language code of the destination URL
Media	screen tty tv projection handheld print braille aural all	Specifies the device the document will be displayed on
Rel	alternate appendix bookmark chapter contents copyright glossary help home index next prev section start stylesheet subsection	Describes the relationship between the current document and the destination URL.
Rev	alternate appendix bookmark chapter	Describes a reverse between the destination URI and the current document.

	contents copyright glossary help home index next prev section start stylesheet subsection	
sizes 	HeightxWidth	Specifies the size of the linked resource.
Target	blank _self _top _parent	Specifies the target frame to load the page into.
Type	mimetype	The MIMETYPE of content at the link destination

### 1.1.9 : Images

Images can improve the design and the appearance of a web page.

#### Example

```

```

#### **HTML Image**





### **HTML Images Syntax**

In HTML, images are defined with the <img> tag.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The src attribute specifies the URL (web address) of the image:

```

```

### **The alt Attribute**

The alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

The value of the alt attribute should describe the image:

### **Example**

```

```

### **Alternative text**

The alt attribute should reflect the image content, so users who cannot see the image gets an understanding of what the image contains:



If a browser cannot find an image, it will display the value of the alt attribute:

### **Image Size - Width and Height**

You can use the style attribute to specify the width and height of an image.

### **Example**

```

```

Alternatively, you can use the width and height attributes:

### **Image Size**

Use the style attribute to specify the width and height of an image:



### **1.1.10 : Formatting Text**

If you use a word processor, you must be familiar with the ability to make text bold, italicized, or underlined; these are just three of the ten options available to indicate how text can appear in HTML and XHTML.

#### **Bold Text**

Anything that appears within `<b>...</b>` element, is displayed in bold as shown below –

#### **Example**

`<html>`

```
<head>
  <title>Bold Text Example</title>
</head>
<body>
  <p>The following word uses a <b>bold</b> typeface.</p>
</body>
</html>
```

### **Italic Text**

Anything that appears within `<i>...</i>` element is displayed in italicized as shown below –

#### **Example**

```
<html>
  <head>
    <title>Italic Text Example</title>
  </head>
  <body>
    <p>The following word uses an <i>italicized</i> typeface.</p>
  </body>
</html>
```

### **Underlined Text**

Anything that appears within `<u>...</u>` element, is displayed with underline as shown below –

#### **Example**

```
<html>
  <head>
    <title>Underlined Text Example</title>
  </head>
  <body>
    <p>The following word uses an <u>underlined</u> typeface.</p>
  </body>
</html>
```

### **Strike Text**

Anything that appears within `<strike>...</strike>` element is displayed with strikethrough, which is a thin line through the text as shown below –

#### **Example**

```
<html>
```

```
<head>
  <title>Strike Text Example</title>
</head>
<body>
  <p>The following word uses a <strike>strikethrough</strike> typeface.</p>
</body>
</html>
```

### Superscript Text

The content of a **<sup>...</sup>** element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed half a character's height above the other characters.

#### Example

```
<html>
<head>
  <title>Superscript Text Example</title>
</head>
<body>
  <p>The following word uses a <sup>superscript</sup> typeface.</p>
</body>
</html>
```

### Subscript Text

The content of a **<sub>...</sub>** element is written in subscript; the font size used is the same as the characters surrounding it, but is displayed half a character's height beneath the other characters.

#### Example

```
<html>
<head>
  <title>Subscript Text Example</title>
</head>
<body>
  <p>The following word uses a <sub>subscript</sub> typeface.</p>
</body>
</html>
```

### Inserted Text

Anything that appears within **<ins>...</ins>** element is displayed as inserted text.

#### Example

```
<html>
  <head>
    <title>Inserted Text Example</title>
  </head>
  <body>
    <p>I want to drink <del>cola</del> <ins>wine</ins></p>
  </body>
</html>
```

### Deleted Text

Anything that appears within **<del>...</del>** element, is displayed as deleted text.

### Example

```
<html>
  <head>
    <title>Deleted Text Example</title>
  </head>
  <body>
    <p>I want to drink <del>cola</del> <ins>wine</ins></p>
  </body>
</html>
```

### Larger Text

The content of the **<big>...</big>** element is displayed one font size larger than the rest of the text surrounding it as shown below –

### Example

```
<html>
  <head>
    <title>Larger Text Example</title>
  </head>
  <body>
    <p>The following word uses a <big>big</big> typeface.</p>
  </body>
</html>
```

### Smaller Text

The content of the **<small>...</small>** element is displayed one font size smaller than the rest of the text surrounding it as shown below –

### Example



```
<html>
<head>
  <title>Smaller Text Example</title>
</head>
<body>
  <p>The following word uses a <small>small</small> typeface.</p>
</body>
</html>
```

### 1.1.11 : Special Characters

Some characters are not available on a standard keyboard. In HTML they can be indicated by special character entities. The special character entities start with an ampersand &, and end with a semicolon;. Between these is a string of letters which usually indicates the character to be shown.

There is also an alternative system of referring to special characters by number but these are less easy to remember. On the other hand it can be more reliable if you are using the less common special characters.

The following is a list of some of the more common special character entities. Not all of them will display in all browsers. If you see, in the right-hand box, not a special character but just a repeat of the code in the left-hand box, your browser has failed to recognize the code.

Character entity	Special character
&nbsp;	Non-breaking space
&amp;	& (ampersand)
&lt;	<
&gt;	>
&copy;	© (copyright symbol)
&reg;	® (Reg. trademark symbol)
&ne;	≠ (not-equals sign; a range of mathematical symbols are available)
&plusmn;	± (plus-minus)
&deg;	° (degree)
&quot;	" (Quotation mark)
&ndash;	– (n-dash)
&mdash;	— (m-dash)
&para;	¶ (paragraph symbol as used in word-processors)
&pound;	£ (pound sterling symbol)
&euro;	€ (Euro currency symbol)
&frac14;	¼ (fraction 1/4)
&frac12;	½ (fraction 1/2)
&frac34;	¾ (fraction 3/4)
&eacute;	é
&Eacute;	É
&egrave;	è
&agrave;	à

<b>&amp;ccedil;</b>	ç
<b>&amp;ecirc;</b>	ê
<b>&amp;acirc;</b>	â
<b>&amp;ucirc;;</b>	û
<b>&amp;auml;</b>	ä
<b>&amp;ouml;</b>	ö
<b>&amp;uuml;</b>	ü
<b>&amp;scaron;</b>	š
<b>&amp;ntilde;</b>	ñ
<b>&amp;szlig;</b>	ß (the German character, not to be confused with beta (see below))
<b>&amp;beta;</b>	β (lower-case Beta; the whole Greek alphabet is accessible like this)

### 1.1.12 : Horizontal Rules and Line Text

The HTML `<hr>` tag is used for creating a horizontal line. This is also called Horizontal Rule in HTML.

#### **Definition and Usage**

The `<hr>` tag defines a thematic break in an HTML page (e.g. a shift of topic).

The `<hr>` element is used to separate content (or define a change) in an HTML page.

#### **Example**

```
<h1>HTML</h1>
```

```
<p>HTML is a language for describing web pages .... </p>
```

```
<hr>
```

```
<h1>CSS</h1>
```

```
<p>CSS defines how to display HTML elements ....</p>
```

#### **Attributes**

Attribute	Value	Description
<u>align</u>	left center right	Not supported in HTML5. Specifies the alignment of a <code>&lt;hr&gt;</code> element
<u>noshade</u>	noshade	Not supported in HTML5. Specifies that a <code>&lt;hr&gt;</code> element should render in one solid color (noshaded), instead of a shaded color
<u>size</u>	<i>pixels</i>	Not supported in HTML5.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

		Specifies the height of a <hr> element
<u>width</u>	<i>pixels</i> %	Not supported in HTML5. Specifies the width of a <hr> element

<html>

<head>

<style>

hr {

display: block;

margin-top: 0.5em;

margin-bottom: 0.5em;

margin-left: auto;

margin-right: auto;

border-style: inset;

border-width: 1px;

}

</style>

</head>

<body>

<p>A hr element is displayed like this:</p>

<hr>

<p>Change the default CSS settings to see the effect.</p>

</body>

</html>



### 1.1.13 : Lists

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements. Lists may contain –

- **<ul>** – An unordered list. This will list items using plain bullets.
- **<ol>** – An ordered list. This will use different schemes of numbers to list your items.
- **<dl>** – A definition list. This arranges your items in the same way as they are arranged in a dictionary.

#### **HTML Unordered Lists**

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML **<ul>** tag. Each item in the list is marked with a bullet.

#### **Example**

<html>

```
<head>
  <title>HTML Unordered List</title>
</head>

<body>
  <ul>
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ul>
</body>

</html>
```

This will produce the following result –

- Beetroot
- Ginger
- Potato
- Radish

### **The type Attribute**

You can use **type** attribute for <ul> tag to specify the type of bullet you like. By default, it is a disc. Following are the possible options –

```
<ul type = "square">
<ul type = "disc">
<ul type = "circle">
```

### **Example**

Following is an example where we used <ul type = "square">

```
<html>
  <head>
    <title>HTML Unordered List</title>
  </head>
```

```
<body>
  <ul type = "square">
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ul>
</body>
</html>
```

This will produce the following result –

- Beetroot
- Ginger
- Potato
- Radish

### **Example**

Following is an example where we used <ul type = "disc"> –

```
<html>
  <head>
    <title>HTML Unordered List</title>
  </head>
  <body>
    <ul type = "disc">
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ul>
  </body>
</html>
```

This will produce the following result –

- Beetroot
- Ginger

- Potato
- Radish

**Example**

Following is an example where we used `<ul type = "circle">` –

```
<html>
<head>
  <title>HTML Unordered List</title>
</head>
<body>
  <ul type = "circle">
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ul>
</body>
</html>
```

This will produce the following result –

- Beetroot
- Ginger
- Potato
- Radish

**HTML Ordered Lists**

If you are required to put your items in a numbered list instead of bulleted, then HTML ordered list will be used. This list is created by using `<ol>` tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with `<li>`.

**Example**

```
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
```

---

```
<ol>
  <li>Beetroot</li>
  <li>Ginger</li>
  <li>Potato</li>
  <li>Radish</li>
</ol>
</body>
</html>
```

This will produce the following result –

1. Beetroot
2. Ginger
3. Potato
4. Radish

### **The type Attribute**

You can use **type** attribute for <ol> tag to specify the type of numbering you like. By default, it is a number. Following are the possible options –

<ol type = "1"> - Default-Case Numerals.

<ol type = "I"> - Upper-Case Numerals.

<ol type = "i"> - Lower-Case Numerals.

<ol type = "A"> - Upper-Case Letters.

<ol type = "a"> - Lower-Case Letters.

### **Example**

Following is an example where we used <ol type = "1">

```
<html>
  <head>
    <title>HTML Ordered List</title>
  </head>
  <body>
    <ol type = "1">
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
```

```
<li>Radish</li>
</ol>
</body>
</html>
```

This will produce the following result –

1. Beetroot
2. Ginger
3. Potato
4. Radish

**Example**

Following is an example where we used `<ol type = "I">`

```
<html>
<head>
  <title>HTML Ordered List</title>
</head>
<body>
  <ol type = "I">
    <li>Beetroot</li>
    <li>Ginger</li>
    <li>Potato</li>
    <li>Radish</li>
  </ol>
</body>
</html>
```

This will produce the following result –

- I. Beetroot
- II. Ginger
- III. Potato
- IV. Radish

**Example**

Following is an example where we used `<ol type = "i">`

```
<html>
```

---

```
<head>

<title>HTML Ordered List</title>

</head>

<body>

  <ol type = "i">

    <li>Beetroot</li>

    <li>Ginger</li>

    <li>Potato</li>

    <li>Radish</li>

  </ol>

</body>

</html>
```

This will produce the following result –

- i. Beetroot
- ii. Ginger
- iii. Potato
- iv. Radish

**Example**

Following is an example where we used <ol type = "A" >

```
<html>

<head>

  <title>HTML Ordered List</title>

</head>

<body>

  <ol type = "A">

    <li>Beetroot</li>

    <li>Ginger</li>

    <li>Potato</li>

    <li>Radish</li>

  </ol>

</body>

</html>
```



This will produce the following result –

- A. Beetroot
- B. Ginger
- C. Potato
- D. Radish

**Example**

Following is an example where we used `<ol type = "a">`

```
<html>
  <head>
    <title>HTML Ordered List</title>
  </head>
  <body>
    <ol type = "a">
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ol>
  </body>
</html>
```

This will produce the following result –

- a. Beetroot
- b. Ginger
- c. Potato
- d. Radish

**The start Attribute**

You can use **start** attribute for `<ol>` tag to specify the starting point of numbering you need.

Following are the possible options –

```
<ol type = "1" start = "4"> - Numerals starts with 4.
<ol type = "I" start = "4"> - Numerals starts with IV.
<ol type = "i" start = "4"> - Numerals starts with iv.
<ol type = "a" start = "4"> - Letters starts with d.
```

<ol type = "A" start = "4"> - Letters starts with D.

**Example**

Following is an example where we used <ol type = "i" start = "4" >

```
<html>
  <head>
    <title>HTML Ordered List</title>
  </head>
  <body>
    <ol type = "i" start = "4">
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ol>
  </body>
</html>
```

This will produce the following result –

- iv. Beetroot
- v. Ginger
- vi. Potato
- vii. Radish

**HTML Definition Lists**

HTML and XHTML supports a list style which is called **definition lists** where entries are listed like in a dictionary or encyclopedia. The definition list is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following three tags.

- <dl> – Defines the start of the list
- <dt> – A term
- <dd> – Term definition
- </dl> – Defines the end of the list

**Example**

```
<html>
  <head>
    <title>HTML Definition List</title>
  </head>
  <body>
    <dl>
      <dt><b>HTML</b></dt>
      <dd>This stands for Hyper Text Markup Language</dd>
      <dt><b>HTTP</b></dt>
      <dd>This stands for Hyper Text Transfer Protocol</dd>
    </dl>
  </body>
</html>
```

This will produce the following result –

**HTML**

This stands for Hyper Text Markup Language

**HTTP**

This stands for Hyper Text Transfer Protocol

**1.1.14 : Basic HTML Tables**

The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.

The HTML tables are created using the **<table>** tag in which the **<tr>** tag is used to create table rows and **<td>** tag is used to create data cells. The elements under **<td>** are regular and left aligned by default

**Example**

```
<html>
  <head>
    <title>HTML Tables</title>
  </head>
```

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

    <tr>
      <td>Row 2, Column 1</td>
      <td>Row 2, Column 2</td>
    </tr>
  </table>

</body>
</html>
```

Here, the **border** is an attribute of <table> tag and it is used to put a border across all the cells. If you do not need a border, then you can use border = "0".

Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

### **Table Heading**

Table heading can be defined using <th> tag. This tag will be put to replace <td> tag, which is used to represent actual data cell. Normally you will put your top row as table heading as shown below, otherwise you can use <th> element in any row. Headings, which are defined in <th> tag are centered and bold by default.

### **Example**

```
<html>

<head>
  <title>HTML Table Header</title>
</head>

<body>
  <table border = "1">
    <tr>
      <th>Name</th>
      <th>Salary</th>
    </tr>
    <tr>
      <td>Ramesh Raman</td>
```

```
<td>5000</td>
</tr>

<tr>
  <td>Shabbir Hussein</td>
  <td>7000</td>
</tr>
</table>
</body>

</html>
```

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

### **Cellpadding and Cellspacing Attributes**

There are two attributes called *cellpadding* and *cellspacing* which you will use to adjust the white space in your table cells. The *cellspacing* attribute defines space between table cells, while *cellpadding* represents the distance between cell borders and the content within a cell.

### **Example**

```
<html>

<head>
  <title>HTML Table Cellpadding</title>
</head>

<body>
  <table border = "1" cellpadding = "5" cellspacing = "5">
    <tr>
      <th>Name</th>
      <th>Salary</th>
    </tr>
    <tr>
      <td>Ramesh Raman</td>
      <td>5000</td>
    </tr>
    <tr>
      <td>Shabbir Hussein</td>
      <td>7000</td>
    </tr>
  </table>
```

&lt;/body&gt;

&lt;/html&gt;

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

**Colspan and Rowspan Attributes**

You will use **colspan** attribute if you want to merge two or more columns into a single column. Similar way you will use **rowspan** if you want to merge two or more rows.

**Example**

&lt;html&gt;

&lt;head&gt;

&lt;title&gt;HTML Table Colspan/Rowspan&lt;/title&gt;

&lt;/head&gt;

&lt;body&gt;

&lt;table border = "1"&gt;

&lt;tr&gt;

&lt;th&gt;Column 1&lt;/th&gt;

&lt;th&gt;Column 2&lt;/th&gt;

&lt;th&gt;Column 3&lt;/th&gt;

&lt;/tr&gt;

&lt;tr&gt;

&lt;td rowspan = "2"&gt;Row 1 Cell 1&lt;/td&gt;

&lt;td&gt;Row 1 Cell 2&lt;/td&gt;

&lt;td&gt;Row 1 Cell 3&lt;/td&gt;

&lt;/tr&gt;

&lt;tr&gt;

&lt;td&gt;Row 2 Cell 2&lt;/td&gt;

&lt;td&gt;Row 2 Cell 3&lt;/td&gt;

&lt;/tr&gt;

&lt;tr&gt;

&lt;td colspan = "3"&gt;Row 3 Cell 1&lt;/td&gt;

&lt;/tr&gt;

&lt;/table&gt;

&lt;/body&gt;

&lt;/html&gt;

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

### Tables Backgrounds

You can set table background using one of the following two ways –

- **bgcolor** attribute – You can set background color for whole table or just for one cell.
- **background** attribute – You can set background image for whole table or just for one cell.

### Example

```
<html>
```

```
<head>
```

```
<title>HTML Table Background</title>
```

```
</head>
```

```
<body>
```

```
<table border = "1" bordercolor = "green" bgcolor = "yellow">
```

```
<tr>
```

```
<th>Column 1</th>
```

```
<th>Column 2</th>
```

```
<th>Column 3</th>
```

```
</tr>
```

```
<tr>
```

```
<td rowspan = "2">Row 1 Cell 1</td>
```

```
<td>Row 1 Cell 2</td>
```

```
<td>Row 1 Cell 3</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Row 2 Cell 2</td>
```

```
<td>Row 2 Cell 3</td>
```

```
</tr>
```

```
<tr>
```

```
<td colspan = "3">Row 3 Cell 1</td>
```

```
</tr>
```

```
</table>
```

```
</body>
```

&lt;/html&gt;

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

**Table Height and Width**

You can set a table width and height using **width** and **height** attributes. You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

**Example**

&lt;html&gt;

&lt;head&gt;

&lt;title&gt;HTML Table Width/Height&lt;/title&gt;

&lt;/head&gt;

&lt;body&gt;

&lt;table border = "1" width = "400" height = "150"&gt;

&lt;tr&gt;

&lt;td&gt;Row 1, Column 1&lt;/td&gt;

&lt;td&gt;Row 1, Column 2&lt;/td&gt;

&lt;/tr&gt;

&lt;tr&gt;

&lt;td&gt;Row 2, Column 1&lt;/td&gt;

&lt;td&gt;Row 2, Column 2&lt;/td&gt;

&lt;/tr&gt;

&lt;/table&gt;

&lt;/body&gt;

&lt;/html&gt;

**Table Caption**

The **caption** tag will serve as a title or explanation for the table and it shows up at the top of the table. This tag is deprecated in newer version of HTML.

**Example**

&lt;html&gt;

&lt;head&gt;

&lt;title&gt;HTML Table Caption&lt;/title&gt;



```
</head>
```

```
<body>
```

```
<table border = "1" width = "100%">
```

```
<caption>This is the caption</caption>
```

```
<tr>
```

```
<td>row 1, column 1</td><td>row 1, columnn 2</td>
```

```
</tr>
```

```
<tr>
```

```
<td>row 2, column 1</td><td>row 2, columnn 2</td>
```

```
</tr>
```

```
</table>
```

```
</body>
```

```
</html>
```

### **Table Header, Body, and Footer**

Tables can be divided into three portions – a header, a body, and a foot. The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content holder of the table.

The three elements for separating the head, body, and foot of a table are –

- **<thead>** – to create a separate table header.
- **<tbody>** – to indicate the main body of the table.
- **<tfoot>** – to create a separate table footer.

A table may contain several `<tbody>` elements to indicate *different pages* or groups of data. But it is notable that `<thead>` and `<tfoot>` tags should appear before `<tbody>`

### **Example**

```
<html>
```

```
<head>
```

```
<title>HTML Table</title>
```

```
</head>
```

```
<body>
```

```
<table border = "1" width = "100%">
```

```
<thead>
```

```
<tr>
```

```
<td colspan = "4">This is the head of the table</td>
```

```
</tr>
```

```
</thead>
```

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: I

BATCH-2017-20

```
<tfoot>
  <tr>
    <td colspan = "4">This is the foot of the table</td>
  </tr>
</tfoot>
```

```
<tbody>
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
    <td>Cell 3</td>
    <td>Cell 4</td>
  </tr>
</tbody>
```

```
</table>
</body>
```

```
</html>
```

This is the head of the table
-------------------------------

This is the foot of the table
-------------------------------

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

**Department of Commerce (CA)**

**SUBJECT: INTERNET AND WEB DESIGNING**

**SEMESTER: VI**

**SUBJECT CODE: 17CCU601A**

**CLASS : III B.COM CA**

**UNIT:I**

S. N O	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	HTML stands for_____.	Hypertext Markup Language	Hyperlink Markup Language	Hypertext Markedup Language	Hypertext Markup Language	Hypertext Markup Language
2	_____are used to markup HTML elements.	HTML tags	HR tags	Head tags	SGML tags	HTML tags
3	A_____element is used to define style information.	Title	Set	Style	Character	Style
4	How many levels of heading elements are in HTML?	6	3	7	8	6
5	_____tag used to break up sections of a document.	Line break	Horizontal line	Preserve	Heading	Horizontal line
6	_____tag set a color for the background of the page.	Bg color	Br color	Bg	Color	Bg color
7	How many categories are in core attributes?	3	4	5	6	4

8	_____ tag is used to create an unordered list.	<OL>	<UL>	<DL>	<LI>	<UL>
9	Unordered list is used for the purpose of_____.	Symbols and bullets	Alphabet and numeric	Numeric and symbols	Heading and symbols	Symbols and bullets
10	<DT> tag is used for_____ in definition list.	Style	Links	Paragraph	Heading	Heading
11	HTML supports a list style which is called as_____.	Definition list	Nested list	Unordered list	Ordered list	Definition list
12	Each item in the list is marked with_____.	Bullets	Numeric	Hash sign	Pound	Bullets
13	<P> tag is used for_____.	Font color	Alignment	Formatting	Links	Formatting
14	URL is abbreviated as_____.	Uniform Resource Location	Uniform Resource locator	Unified Resource Locator	Uniform Research Locator	Uniform Resource locator
15	_____ tag used to create a current position in the HTML document.	 	<hr>	<b>	<p>	<hr>
16	HTML is known as_____ language.	Platform dependent	Platform independent	Place dependent	Place Independent	Platform independent
17	The Internet was originally _____ project agency.	NSF	NSA	APRA	ARPA	ARPA
18	HTML is used to create_____.	Machine language program	High level program	Web page	Web server	Web page
19	Internet explorer falls under_____.	Operating System	Compiler	Browser	IP address	Browser
20	WWW stands for_____.	World Wide Web	World Wide Word	World Wide Wood	World Wide Width	World Wide Web
21	ALL HTML tags are enclosed in what ?	# and #	? and !	< and >	{ and }	< and >
22	To create HTML page, you need	web browser and text editor	web browser	text editor	Web server	web browser and text editor

23	<a> and </a> are the tags used for	Adding image	Aligning text	Audio-voiced text	Adding links to your page	Adding links to your page
24	To add a plain color background to your web page, use which of the following ?	<body bgcolor="36,24,35">	<body color="Green">	<body bgcolor="Green">	<body bgcolor="Green">	<body bgcolor="Green">
25	The BODY tag is usually used after	HTML tag	EM tag	TITLE tag	HEAD tag	HTML tag
26	Choose the correct HTML tag to make a text italic	<i>	<italic>	<it>	<il>	<i>
27	What does the   tag add to your webpage ?	Line break	pragraph break	long break	break	pragraph break
28	Adding a border to your image helps the visitor to recognize it as _____.	frame	link	picture	tag	picture
29	The first tag inside <TABLE> tag is	<HEAD>	<CAPTION>	<TH>	<TD>	<CAPTION>
30	Which tag tells the browser where the page starts and stops ?	html	body	head	title	html
31	In HTML, tags that include both on and off tag are called	comment tag	document tag	container tag	command tag	container tag
32	Which tag will you add to specify a font for your whole page ?	<defaultfont>	<targetfont>	<basefont>	<font>	<defaultfont>
33	The tag used for creating hypertext and hypermedia links is	<HR>	 	<PRE>	<A>	<A>
34	The main container for <TR>, <TD> and <TH> is	<DATA>	<GROUP>	<TABLE>	<Th>	<TABLE>
35	What is the correct HTML for creating a hyperlink ?	<a>https://w.w.gkseries.com</a>	<a name="https://w.w.gkseries.com">Gkseries.com</a>	<a url="https://w.w.gkseries.com">Gkseries.com</a>	<a href="https://w.w.gkseries.com">Gkseries.com</a>	<a href="https://w.w.gkseries.com">Gkseries.com</a>
36	All normal webpages consists of	Top and bottom	Body and	Head and body	Body and html	Head and body

			frameset			
37	<! Is a	Comment tag	Underlined tag	Underlined with italic tag	common tag	Comment tag
38	Choose the correct HTML tag for the largest heading	<h1>	<h6>	<h3>	<h2>	<h1>
39	To created a bulleted list, use	<ol>	<ul>	<dl>	<li>	<ul>
40	_____is a network in which the computers are connected directly.	LAN	WAN	NET	ARPANET	LAN
41	_____ is connected to satellites links.	WAN	LAN	WWW	None	WAN
42	Which of the following is the “Backbone” of the network?	LAN	WAN	NET	Network	LAN
43	LANs are connected by the special purpose of computers are called _____.	Routers	Resources	Networks	None	Routers
44	_____ is used to access resources.	Clients	Servers	Client - Servers	Internet	Clients
45	In web language, a client program is called a _____.	Browser	Address	Server	All the above	Browser
46	To send and receive electronic mail, one should familiar with_____.	mail client	client – server	internet	e-mail address	mail client
47	Within the internet, each separate computer is called _____.	Host	Node	Both 1 & 2	None	Host
48	Programmers write their programs using _____.	Protocols	Script	Both 1 & 2	None	Protocols
49	TCP stands for _____.	Transmission Control Protocol	Transiting Control Protocol	Transmission Control Process	Transmission Control Programs	Transmission Control Protocol
50	IP stands for_____.	Internet Protocol	I – Protocol	Internet – P	None	Internet Protocol

51	_____ will divide the message into a number of _____.	TCP, Packets	IP, Packets	TCP, Host	IP, Host	TCP, Packets
52	The data containing links to another data is called _____.	Hypertext	TCP	E-mail address	None	Hypertext
53	The name Usenet is a contraction of _____.	User network	User's network	Network	None	User's network
54	Which one of the following has main-menu and sub- menus?	Gopher	Usenet	Web	Telnet	Gopher
55	The FTP service allows to _____ from one computer to another.	Copy files	Send data	Connect	All the above	Copy files
56	FTP stands for_____.	File Transfer Protocol	File Transaction Protocol	Fixed Transfer Protocol	File Transiting Protocol	File Transfer Protocol
57	Anonymous "ftp" is one of the most important in _____.	Internet Services	Web Services	Gopher Services	Usenet Services	Internet Services
58	The role of_____ is to make the whole system manageable.	Archie	Gopher	Website	Usenet	Archie
59	The service allowing us to login and use a remote computer is called _____.	Telnet	Usenet	IP	None	Telnet
60	Menu – Based information is a _____.	Gopher	Archie	Usenet	Mail	Gopher

**UNIT-II****SYLLABUS**

**Formatting:** Basic HTML forms – creating and using image maps – meta tags – frameset – nested frameset. **Introduction to HTML:** cascading Style Sheet – Inline Style – Creating Style Sheet with style elements – conflicting styles – Linking External Style Sheets – Positioning Elements – Backgrounds – Element Dimension – text flow – box models.

**2.1 Basic HTML Forms**

HTML Forms are required, when you want to collect some data from the site visitor. For example, during user registration you would like to collect information such as name, email address, credit card, etc.

A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML **<form>** tag is used to create an HTML form and it has following syntax –

**<form>**

.

*form elements*

.

**</form>**

An HTML form contains **form elements**.

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

**The <input> Element**

The **<input>** element is the most important form element.

The **<input>** element can be displayed in several ways, depending on the **type** attribute.



Here are some examples:

Type	Description
<code>&lt;input type="text"&gt;</code>	Defines a one-line text input field
<code>&lt;input type="radio"&gt;</code>	Defines a radio button (for selecting one of many choices)
<code>&lt;input type="submit"&gt;</code>	Defines a submit button (for submitting the form)

### 1. Text Input

`<input type="text">` defines a one-line input field for **text input**:

```
<html>
```

```
<body>
```

```
<h2>Text Input</h2>
```

```
<form>
```

First name:<br>

```
<input type="text" name="firstname">
```

```
<br>
```

Last name:<br>

```
<input type="text" name="lastname">
```

```
</form>
```

```
<p>Note that the form itself is not visible.</p>
```

```
<p>Also note that the default width of a text input field is 20 characters.</p>
```

```
</body>
```

```
</html>
```

Produces the output as,

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: II

BATCH-2017-20

### Text Input

First name:

Last name:

Note that the form itself is not visible.

Also note that the default width of a text input field is 20 characters.

Sr.No	Attribute & Description
1	<b>type</b>  Indicates the type of input control and for text input control it will be set to <b>text</b> .
2	<b>name</b>  Used to give a name to the control which is sent to the server to be recognized and get the value.
3	<b>value</b>  This can be used to provide an initial value inside the control.
4	<b>size</b>  Allows to specify the width of the text-input control in terms of characters.
5	<b>maxlength</b>  Allows to specify the maximum number of characters a user can enter into the text box.

### 2. Radio Button Input

`<input type="radio">` defines a **radio button**.

Radio buttons let a user select ONE of a limited number of choices:

`<html>`

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: II

BATCH-2017-20

```
<body>
<h2>Radio Buttons</h2>
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
</body>
</html>
```

Produces the output as,

### **Radio Buttons**

- ☒ Male
- ☐ Female
- ☐ Other

Sr.No	Attribute & Description
1	<b>type</b>  Indicates the type of input control and for checkbox input control it will be set to radio.
2	<b>name</b>  Used to give a name to the control which is sent to the server to be recognized and get the value.
3	<b>value</b>  The value that will be used if the radio box is selected.
4	<b>Checked</b>  Set to <i>checked</i> if you want to select it by default.

### 3. The Submit Button

`<input type="submit">` defines a button for **submitting** the form data to a **form-handler**.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's **action** attribute:

```
<html>
```

```
<body>
```

```
<h2>HTML Forms</h2>
```

```
<form action="/action_page.php">
```

```
  First name:<br>
```

```
  <input type="text" name="firstname" value="Mickey">
```

```
  <br>
```

```
  Last name:<br>
```

```
  <input type="text" name="lastname" value="Mouse">
```

```
  <br><br>
```

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

```
</form>
```

```
<p>If you click the "Submit" button, the form-data will be sent to a page called
```

```
"/action_page.php".</p>
```

```
</body>
```

```
</html>
```

Produces the output as,

#### **HTML Forms**

First name:

Last name:

If you click the "Submit" button, the form-data will be sent to a page called

"/action\_page.php".

### **The Action Attribute**

The **action** attribute defines the action to be performed when the form is submitted.

Normally, the form data is sent to a web page on the server when the user clicks on the submit button.

In the example above, the form data is sent to a page on the server called `"/action_page.php"`.

This page contains a server-side script that handles the form data:

```
<form action="/action_page.php">
```

If the **action** attribute is omitted, the action is set to the current page.

### **The Target Attribute**

The **target** attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.

The default value is `"_self"` which means the form will be submitted in the current window.

To make the form result open in a new browser tab, use the value `"_blank"`:

```
<form action="/action_page.php" target="_blank">
```

Other legal values are `"_parent"`, `"_top"`, or a name representing the name of an `iframe`.

### **The Method Attribute**

The **method** attribute specifies the HTTP method (**GET** or **POST**) to be used when submitting the form data:

```
<form action="/action_page.php" method="get">
```

The default method when submitting form data is **GET**.

However, when **GET** is used, the submitted form data will be visible in the page address field.

```
<form action="/action_page.php" method="post">
```

Always use **POST** if the form data contains sensitive or personal information. The **POST** method does not display the submitted form data in the page address field.

## **4. The <select> Element**

The `<select>` element defines a **drop-down list**:

```
<html>
```

<body>

<h2>The select Element</h2>

<p>The select element defines a drop-down list:</p>

<form action="/action\_page.php">

<select name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="fiat">Fiat</option>

<option value="audi">Audi</option>

</select>

<br><br>

<input type="submit">

</form>

</body>

</html>

Produces the output as,

**The select Element**

The select element defines a drop-down list:



Submit

The <option> elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

**Allow Multiple Selections:**

Use the `multiple` attribute to allow the user to select more than one value:

```
<html>
<body>
<h2>Allow Multiple Seletcions</h2>
<p>Use the multiple attribute to allow the user to select more than one value.</p>
<form action="/action_page.php">
  <select name="cars" size="4" multiple>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>
<p>Hold down the Ctrl (windows) / Command (Mac) button to select multiple options.</p>
</body>
</html>
```

Produces the output as,

**Allow Multiple Seletcions**

Use the multiple attribute to allow the user to select more than one value.



Volvo  
Saab  
Fiat  
Audi



Submit

Hold down the Ctrl (windows) / Command (Mac) button to select multiple options.

### **The <textarea> Element**

The `<textarea>` element defines a multi-line input field (**a text area**):

```
<html>
<body>
<h2>Textarea</h2>
<p>The textarea element defines a multi-line input field.</p>
<form action="/action_page.php">
  <textarea name="message" rows="10" cols="30">The cat was playing in the
garden.</textarea>
  <br>
  <input type="submit">
</form>
</body>
</html>
```

Produces the output as,

### **Textarea**

The textarea element defines a multi-line input field.



Submit

The rows attribute specifies the visible number of lines in a text area.

The cols attribute specifies the visible width of a text area.



## 2.2 Creating and using image maps

The <map> tag is used to define a client-side image-map. An image-map is an image with clickable areas.

The required name attribute of the <map> element is associated with the <img>'s usemap attribute and creates a relationship between the image and the map.

The <map> element contains a number of <area> elements, that defines the clickable areas in the image map.

```
<html>

<body>

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



<map name="planetmap">

  <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">

  <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm">

  <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">

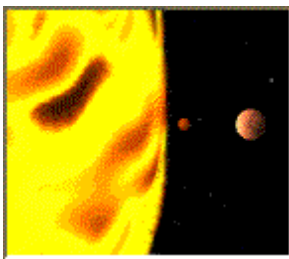
</map>

</body>

</html>
```

### **Produces the output as follows**

Click on the sun or on one of the planets to watch it closer:



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: II

BATCH-2017-20

Attribute	Value	Description
<u>alt</u>	<i>text</i>	Specifies an alternate text for the area. Required if the href attribute is present
<u>coords</u>	<i>coordinates</i>	Specifies the coordinates of the area
<u>download</u>	<i>filename</i>	Specifies that the target will be downloaded when a user clicks on the hyperlink
<u>href</u>	<i>URL</i>	Specifies the hyperlink target for the area
<u>hreflang</u>	<i>language_code</i>	Specifies the language of the target URL
<u>media</u>	<i>media query</i>	Specifies what media/device the target URL is optimized for
<u>nohref</u>	<i>value</i>	Not supported in HTML5. Specifies that an area has no associated link
<u>rel</u>	alternate author bookmark help license next nofollow norereferrer prefetch prev search tag	Specifies the relationship between the current document and the target URL
<u>shape</u>	default rect circle	Specifies the shape of the area

	poly	
<u>target</u>	_blank _parent _self _top <i>framename</i>	Specifies where to open the target URL
<u>type</u>	<i>media_type</i>	Specifies the media type of the target URL

## 2.3 Meta Tags

HTML lets you specify metadata - additional important information about a document in a variety of ways. The META elements can be used to include name/value pairs describing properties of the HTML document, such as author, expiry date, a list of keywords, document author etc.

The **<meta>** tag is used to provide such additional information. This tag is an empty element and so does not have a closing tag but it carries information within its attributes.

You can include one or more meta tags in your document based on what information you want to keep in your document but in general, meta tags do not impact physical appearance of the document so from appearance point of view, it does not matter if you include them or not.

### **Adding Meta Tags to Your Documents**

You can add metadata to your web pages by placing **<meta>** tags inside the header of the document which is represented by **<head>** and **</head>** tags. A meta tag can have following attributes in addition to core attributes –

Sr.No	Attribute & Description
1	<b>Name</b> Name for the property. Can be anything. Examples include, keywords, description, author, revised, generator etc.
2	<b>content</b> Specifies the property's value.

3	<b>scheme</b> Specifies a scheme to interpret the property's value as declared in the content attribute as declared in the content attribute.
4	<b>http-equiv</b> Used for http response message headers. For example, http-equiv can be used to refresh the page or to set a cookie. Values include content-type, expires, refresh and set-cookie.

**Specifying Keywords**

You can use <meta> tag to specify important keywords related to the document and later these keywords are used by the search engines while indexing your webpage for searching purpose.

**Example**

Following is an example, where we are adding HTML, Meta Tags, Metadata as important keywords about the document.

```
<html>
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<p>All meta information goes in the head section...</p>
</body>
</html>
```

This will produce the following result –

**All meta information goes in the head section...**

**2.4 Frameset**

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in

the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

The <frameset> tag is not supported in HTML5.

The <frameset> tag defines a frameset.

The <frameset> element holds one or more <frame> elements. Each <frame> element can hold a separate document.

The <frameset> element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

### **Creating Frames**

---

To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines, how to divide the window into frames. The **rows** attribute of <frameset> tag defines horizontal frames and **cols** attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

### **Example**

Following is the example to create three vertical frames –

```
<html>

<frameset cols="25%,*,25%">

  <frame src="frame_a.html">

  <frame src="frame_b.html">

  <frame src="frame_c.html">

</frameset>

</html>
```

#### **frame\_a.html**

```
<html>
```

```
<body bgcolor="pink">
```

```
<h1>Frame A</h1>
```

```
<p>Note: The frameset, frame, and noframes elements are not supported in  
HTML5.</p>
```

```
</body>
```

```
</html>
```

**frame\_b.html**

```
<html>
```

```
<body bgcolor="blue">
```

```
<h1>Frame B</h1>
```

```
</body>
```

```
</html>
```

**frame\_c.html**

```
<html>
```

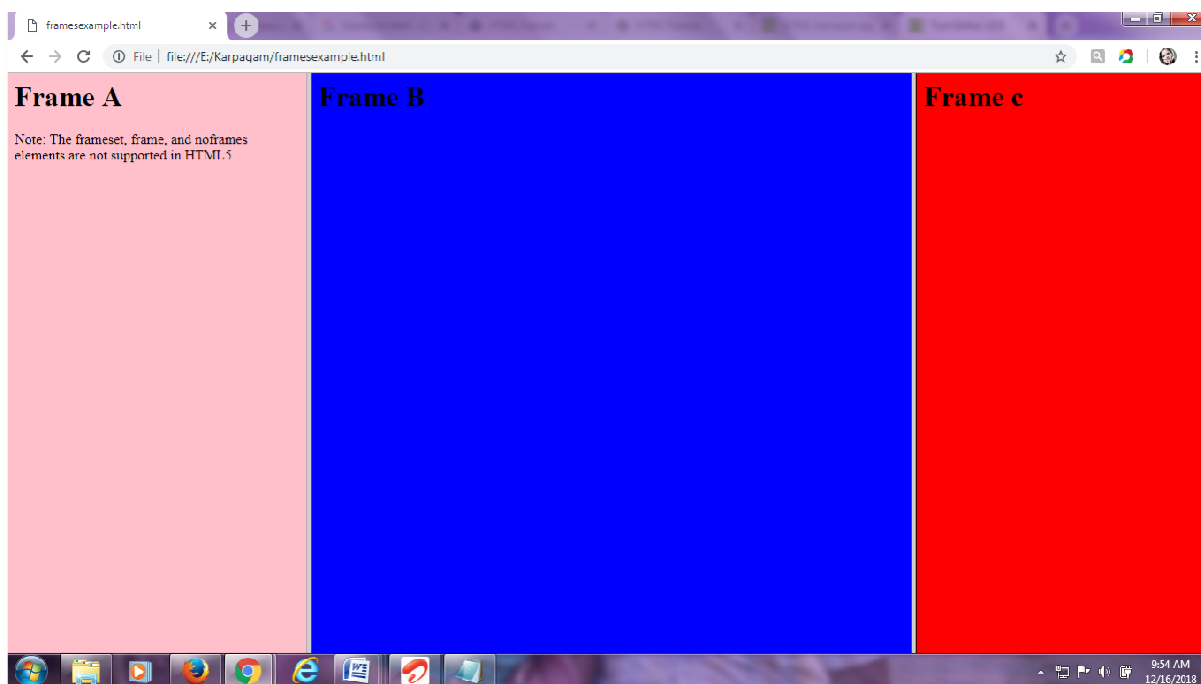
```
<body bgcolor="red">
```

```
<h1>Frame c</h1>
```

```
</body>
```

```
</html>
```

This will produce the following result –



### The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag –

Sr.No	Attribute & Description
1	<b>cols</b> Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways – Absolute values in pixels. For example, to create three vertical frames, use <i>cols = "100, 500, 100"</i> . A percentage of the browser window. For example, to create three vertical frames, use <i>cols = "10%, 80%, 10%"</i> . Using a wildcard symbol. For example, to create three vertical frames, use <i>cols = "10%, *, 10%"</i> . In this case wildcard takes remainder of the window. As relative widths of the browser window. For example, to create three vertical frames, use <i>cols = "3*, 2*, 1*"</i> . This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
2	<b>Rows</b> This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use <i>rows = "10%, 90%"</i> . You can specify the height of each row in the same way as explained above for columns.
3	<b>Border</b> This attribute specifies the width of the border of each frame in pixels. For example, <i>border = "5"</i> . A value of zero means no border.
4	<b>frameborder</b>

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: II

BATCH-2017-20

	This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 yesyes or 0 nono. For example frameborder = "0" specifies no border.
5	<b>framespacing</b> This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames.

### The <frame> Tag Attributes

Following are the important attributes of <frame> tag –

Sr.No	Attribute & Description
1	<b>src</b> This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2	<b>name</b> This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3	<b>frameborder</b> This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 yesyes or 0 nono.
4	<b>marginwidth</b> This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10".
5	<b>marginheight</b> This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight = "10".
6	<b>noresize</b> By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize = "noresize".
7	<b>scrolling</b> This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.
8	<b>longdesc</b> This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc = "framedescription.htm"



## 2.5 Nested Frameset

When two different types of frameset like horizontal and vertical frames created for a same webpage is said to a nested frameset.

### Example

#### **nestedframesetexample.html**

```
<html>

<frameset rows="50%,50%">

  <frame src="frame_a.html">

  <frameset cols="25%,75%">

    <frame src="frame_b.html">

    <frame src="frame_c.html">

  </frameset>

</frameset>

</html>
```

#### **frame\_a.html**

```
<html>

<body bgcolor="pink">

<h1>Frame A</h1>

<p>Note: The frameset, frame, and noframes elements are not supported in
HTML5.</p>

</body>

</html>
```

#### **frame\_b.html**

```
<html>

<body bgcolor="blue">
```

```
<h1>Frame B</h1>
```

```
</body>
```

```
</html>
```

**frame\_c.html**

```
<html>
```

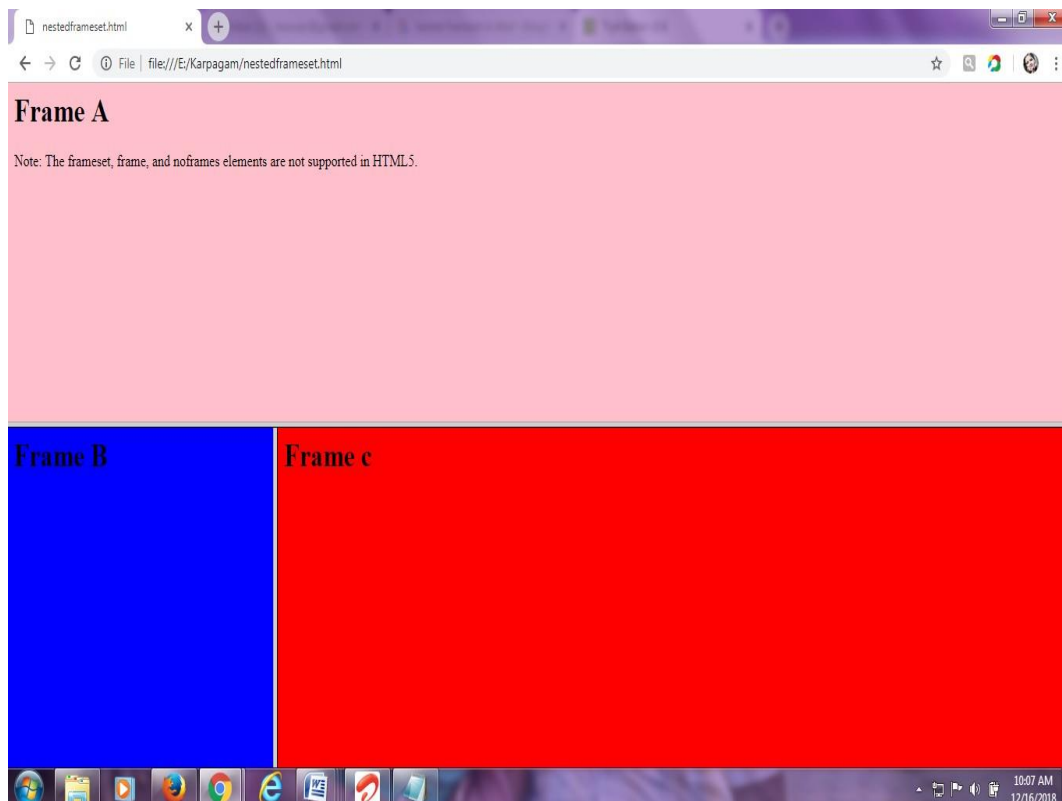
```
<body bgcolor="red">
```

```
<h1>Frame c</h1>
```

```
</body>
```

```
</html>
```

Produces the output as follows:



## 2.6 Introduction to DHTML

DHTML stands for Dynamic HTML, it is totally different from HTML. The browsers which support the dynamic HTML are some of the versions of Netscape Navigator and Internet Explorer of version higher than 4.0. The DHTML is based on the properties of the HTML, javascript, CSS, and DOM (Document Object Model which is used to access individual elements of a document) which helps in making dynamic content. It is the combination of HTML, CSS, JS, and DOM. The DHTML make use of Dynamic object model to make changes in settings and also in properties and methods. It also makes uses of Scripting and it is also part of earlier computing trends.

DHTML allows different scripting languages in a web page to change their variables, which enhance the effects, looks and many others functions after the whole page have been fully loaded or under a view process, or otherwise static HTML pages on the same. But in true ways, there is noting that as dynamic in DHTML, there is only the enclosing of different technologies like CSS, HTML, JS, DOM, and different sets of static languages which make it as dynamic.

DHTML is used to create interactive and animated web pages that are generated in real-time, also known as dynamic web pages so that when such a page is accessed, the code within the page is analyzed on the web server and the resulting HTML is sent to the client's web browser.

**HTML:** HTML stands for Hypertext Markup Language and it is a client-side markup language. It is used to build the block of web pages.

**Javascript:** It is a Client-side Scripting language. Javascript is supported by most of the browser, also have cookies collection to determine the user needs.

**CSS:** The abbreviation of CSS is Cascading Style Sheet. It helps in the styling of the web pages and helps in designing of the pages. The CSS rules for DHTML will be modified at different levels using JS with event handlers which adds a significant amount of dynamism with very little code.

**DOM:** It is known as a dynamic Object Model which act as the weakest links in it. The only defect in it is that most of the browser does not support DOM. It is a way to manipulate the static contents.

**Key Features**

Following are the some major key features of DHTML:

- Tags and their properties can be changed using DHTML.
- It is used for real-time positioning.
- Dynamic fonts can be generated using DHTML.
- It is also used for data binding.
- It makes a webpage dynamic and be used to create animations, games, applications along with providing new ways of navigating through websites.
- The functionality of a webpage is enhanced due to the usage of low-bandwidth effect by DHTML.
- DHTML also facilitates the use of methods, events, properties, and codes.

**Usages of DHTML**

DHTML makes a webpage dynamic but Javascript also does, the question arises that what different does DHTML do? So the answer is that DHTML has the ability to change a webpages look, content and style once the document has loaded on our demand without changing or deleting everything already existing on the browser's webpage. DHTML can change the content of a webpage on demand without the browser having to erase everything else, i.e. being able to alter changes on a webpage even after the document has completely loaded.

**Advantages**

- Size of the files are compact in compared to other interactional media like Flash or Shockwave, and it downloads faster.
- It is supported by big browser manufacturers like Microsoft and Netscape.
- Highly flexible and easy to make changes.
- Viewer requires no extra plug-ins for browsing through the webpage that uses DHTML, they do not need any extra requirements or special software to view it.
- User time is saved by sending less number of requests to the server. As it is possible to modify and replace elements even after a page is loaded, it is not required to create separate pages for changing styles which in turn saves time in building pages and also reduces the number of requests that are sent to the server.

- It has more advanced functionality than a static HTML. it is capable of holding more content on the web page at the same time.

**Disadvantages**

- It is not supported by all the browsers. It is supported only by recent browsers such as Netscape 6, IE 5.5, and Opera 5 like browsers.
- Learning of DHTML requires a lot of pre-requisites languages such as HTML, CSS, JS, etc should be known to the designer before starting with DHTML which is a long and time-consuming in itself.
- Implementation of different browsers are different. So if it worked in one browser, it might not necessarily work the same way in another browser.
- Even after being great with functionality, DHTML requires a few tools and utilities that are some expensive. For example, the DHTML text editor, Dreamweaver. Along with it the improvement cost of transferring from HTML to DHTML makes cost rise much higher.

**Difference between HTML and DHTML**

- HTML is a markup language while DHTML is a collection of technologies.
- HTML is used to create static webpages while DHTML is capable of creating dynamic webpages.
- DHTML is used to create animations and dynamic menus but HTML not used.
- HTML sites are slow upon client-side technologies whereas DHTML sites are comparatively faster.
- Web pages created using HTML are rather simple and have no styling as it uses only one language whereas DHTML uses HTML, CSS, and Javascript which results in a much better and way more presentable webpage.
- HTML cannot be used as server side code but DHTML used as server side code.
- DHTML needs database connectivity but not in case of HTML.
- Files in HTML are stored using .htm or .html extension while DHTML uses .dhtm extension.
- HTML requires no processing from the browser but DHTML does.

## 2.7 Cascading Style Sheet

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

### Advantages of CSS

- **CSS saves time** – You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** – If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** – CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** – Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

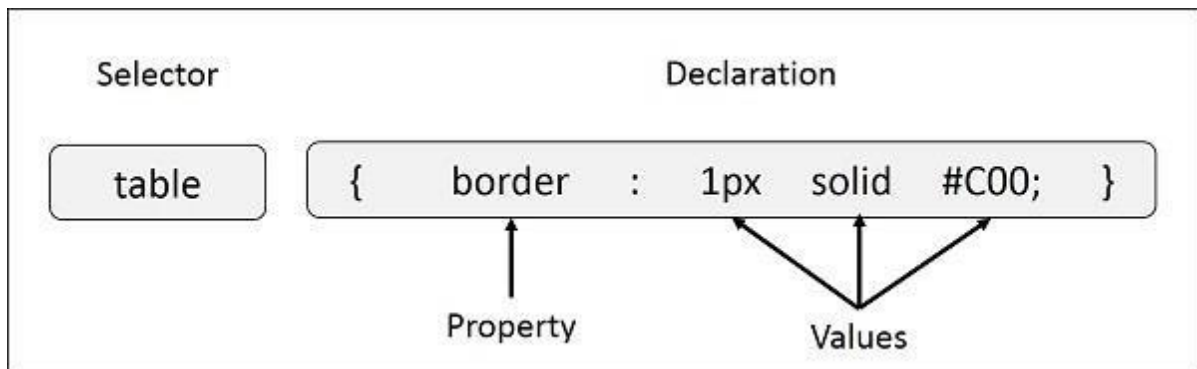
## CSS Syntax

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts –

- **Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc.
- **Property** – A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.
- **Value** – Values are assigned to properties. For example, *color* property can have value either *red* or *#F1F1F1* etc.

You can put CSS Style Rule Syntax as follows –

***selector { property: value }***



### Example

**`table{ border :1px solid #C00; }`**

Here `table` is a selector and `border` is a property and given value `1px solid #C00` is the value of that property.

### The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings –

```
h1 {  
    color: #36CFFF;  
}
```

### The Universal Selectors

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type –

```
* {  
  color: #000000;  
}
```

This rule renders the content of every element in our document in black.

### The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to `<em>` element only when it lies inside `<ul>` tag.

```
ul em {  
  color: #000000;  
}
```

### The Class Selectors

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {  
  color: #000000;  
}
```

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example –

```
.black {  
  color: #000000;  
}
```

This rule renders the content in black for only `<h1>` elements with class attribute set to black. Consider the following example –

```
<p class = "center bold">
```

**This para will be styled by the classes center and bold.**

```
</p>
```

### The ID Selectors

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.



```
#black {  
    color: #000000;  
}
```

This rule renders the content in black for every element with id attribute set to black in our document. You can make it a bit more particular. For example –

```
h1#black {  
    color: #000000;  
}
```

This rule renders the content in black for only <h1> elements with id attribute set to black. The true power of id selectors is when they are used as the foundation for descendant selectors, For example –

```
#black h2 {  
    color: #000000;  
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie within tags having id attribute set to black.

### **The Child Selectors**

You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example –

```
body > p {  
    color: #000000;  
}
```

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like <div> or <td> would not have any effect of this rule.

### **The Attribute Selectors**

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of text –

```
input[type = "text"] {  
    color: #000000;  
}
```

The advantage to this method is that the `<input type = "submit" />` element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

- `p[lang]` – Selects all paragraph elements with a lang attribute.
- `p[lang="fr"]` – Selects all paragraph elements whose lang attribute has a value of exactly "fr".
- `p[lang~="fr"]` – Selects all paragraph elements whose lang attribute contains the word "fr".
- `p[lang|="en"]` – Selects all paragraph elements whose lang attribute contains values that are exactly "en", or begin with "en-".

### Multiple Style Rules

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example –

```
h1 {  
  color: #36C;  
  font-weight: normal;  
  letter-spacing: .4em;  
  margin-bottom: 1em;  
  text-transform: lowercase;  
}
```

Here all the property and value pairs are separated by a semicolon (;). You can keep them in a single line or multiple lines. For better readability, we keep them in separate lines.

For a while, don't bother about the properties mentioned in the above block. These properties will be explained in the coming chapters and you can find complete detail about properties in CSS References.

### Grouping Selectors

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example –

```
h1, h2, h3 {  
  color: #36C;
```

```
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine the various id selectors together as shown below –

```
#content, #footer, #supplement {
    position: absolute;
    left: 510px;
    width: 200px;
}
```

### Ways to insert style sheets in CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

## 2.8 Inline Style Sheet

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

```
<html>

<body>

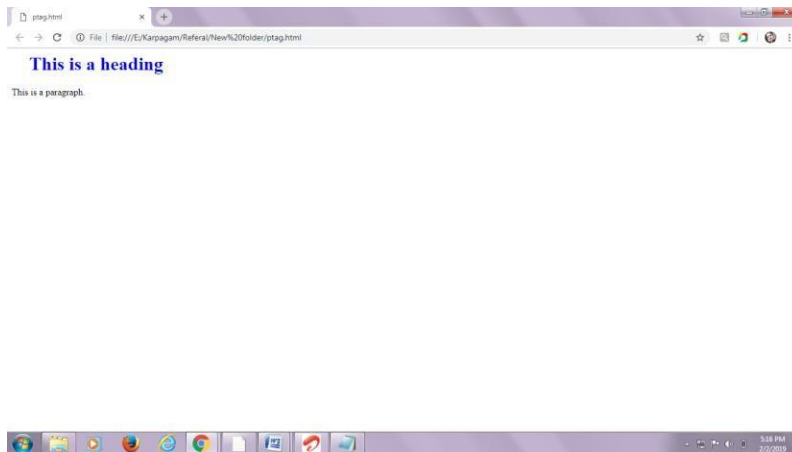
<h1 style="color:blue;margin-left:30px;">This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>
```

### Output



## 2.9 Creating Style Sheets with Style Elements

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

```
<html>

<head>

<style>

body {

    background-color: linen;

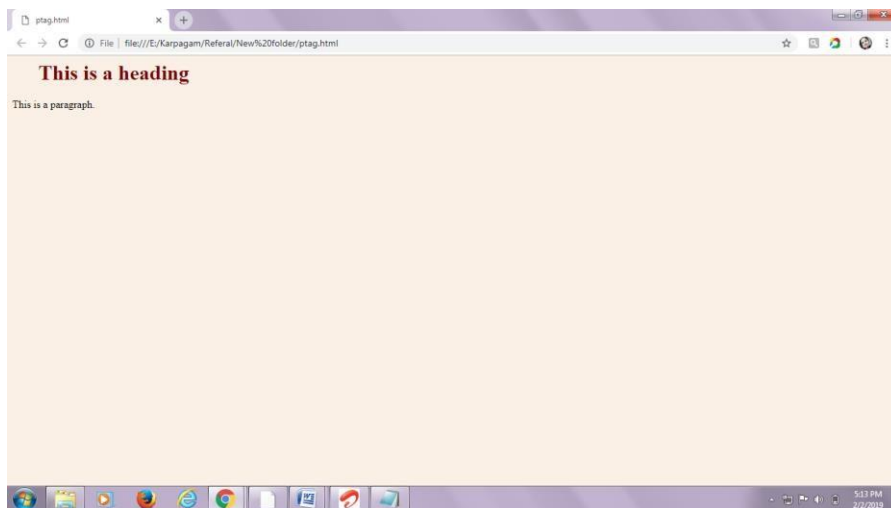
}

h1 {

    color: maroon;
```

```
margin-left: 40px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

### Output



## 2.10 Conflicting Styles

When more than one type of style sheet is used in a HTML program then it is said to conflicting styles.

Here in this example program you can see a HTML program with the combination of both external and internal style sheets.

In this example program style element is used along with the external css file.

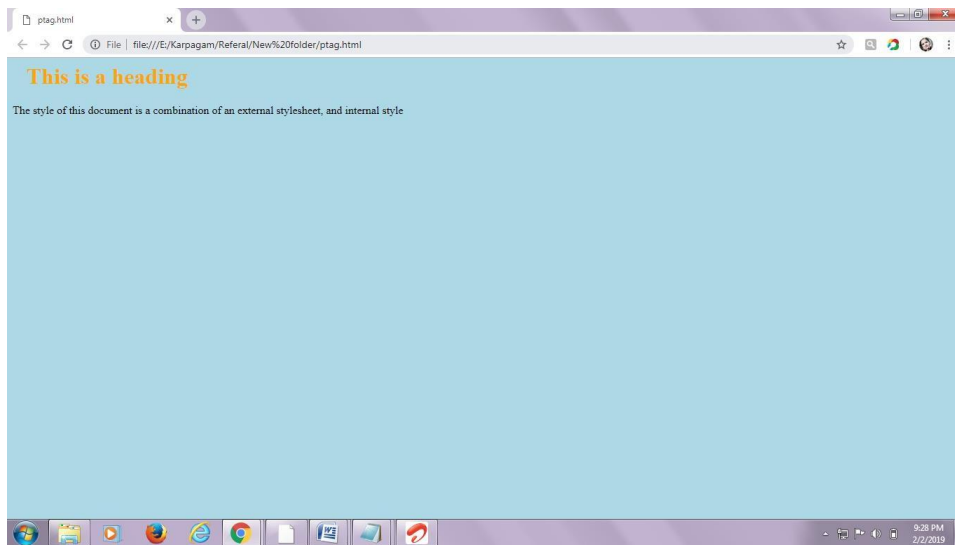
```
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
```

```
h1 {  
    color: orange;  
}  
  
</style>  
  
</head>  
  
<body>  
  
<h1>This is a heading</h1>  
  
<p>The style of this document is a combination of an external stylesheet, and internal  
style</p>  
  
</body>  
  
</html>
```

**mystyle.css**

```
body {  
    background-color: lightblue;  
}  
  
h1 {  
    color: navy;  
    margin-left: 20px;  
}
```

**Output**



## 2.11 Linking External Style Sheets

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the `<link>` element.

The `<link>` element goes inside the `<head>` section.

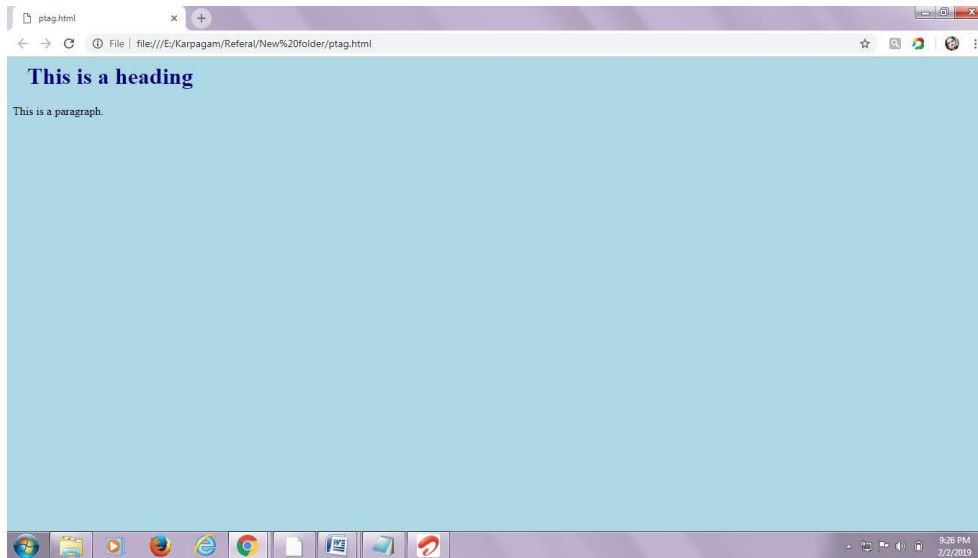
### Example Program

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

```
body {
    background-color: lightblue;
}
h1 {
```

```
color: navy;  
margin-left: 20px;  
}
```

**Output****2.12 Position Elements**

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

**Syntax**

*position: static/absolute/fixed/relative/sticky/initial/inherit;*

**Property Values**

Value	Description
static	Default value. Elements render in order, as they appear in the document flow
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position

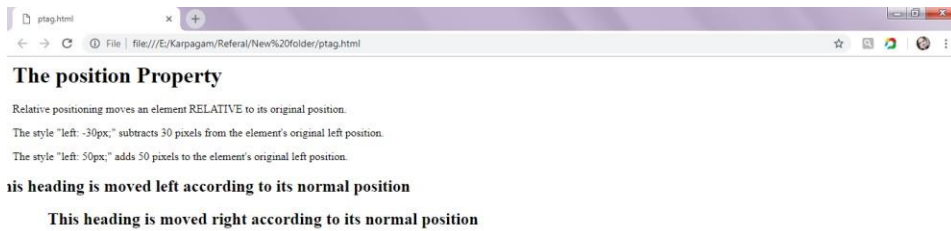


sticky	The element is positioned based on the user's scroll position  A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

**Example Program**

```
<html>
<head>
<style>
h2.pos_left {
position: relative;
left: -30px;
}
h2.pos_right {
position: relative;
left: 50px;
}
</style>
</head>
<body>
<h1>The position Property</h1>
<p>Relative positioning moves an element RELATIVE to its original position.</p>
<p>The style "left: -30px;" subtracts 30 pixels from the element's original left position.</p>
<p>The style "left: 50px;" adds 50 pixels to the element's original left position.</p>
<h2 class="pos_left">This heading is moved left according to its normal position</h2>
<h2 class="pos_right">This heading is moved right according to its normal position</h2>
</body>
</html>
```

## Output



## 2.13 Backgrounds

The background property is a shorthand property for:

- background-color
- background-image
- background-position
- background-size
- background-repeat
- background-origin
- background-clip
- background-attachment

It does not matter if one of the values above are missing, e.g. background:#ff0000 url(smiley.gif); is allowed.

### Syntax

background: *bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment* initial|inherit;

If one of the properties in the shorthand declaration is the bg-size property, you must use a / (slash) to separate it from the bg-position property, e.g. background:url(smiley.gif) 10px 20px/50px 50px; will result in a background image, positioned 10 pixels from the left, 20 pixels from the top, and the size of the image will be 50 pixels wide and 50 pixels high.

If using multiple background-image sources but also want a background-color, the background-color parameter needs to be last in the list.

**Property Values**

## Property Values

<b>Value</b>	<b>Description</b>
<u><i>background-color</i></u>	Specifies the background color to be used
<u><i>background-image</i></u>	Specifies ONE or MORE background images to be used
<u><i>background-position</i></u>	Specifies the position of the background images
<u><i>background-size</i></u>	Specifies the size of the background images
<u><i>background-repeat</i></u>	Specifies how to repeat the background images
<u><i>background-origin</i></u>	Specifies the positioning area of the background images
<u><i>background-clip</i></u>	Specifies the painting area of the background images
<u><i>background-attachment</i></u>	Specifies whether the background images are fixed or scrolls with the rest of the page
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

The background-image property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

The background image for a page can be set like this:

**Example**

```
<html>
<head>
<style>
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
```

```
}  
</style>  
</head>  
<body>  
<h1>Hello World!</h1>  
<p>Here, a background image is repeated only horizontally!</p>  
</body>  
</html>
```

**Output****Hello World!**

Here, a background image is repeated only horizontally!

**2.14 Element Dimension**

The height and width properties are used to set the height and width of an element.

The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in length values, like px, cm, etc., or in percent (%) of the containing block.

**Setting max-width**

The max-width property is used to set the maximum width of an element.

The max-width can be specified in length values, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

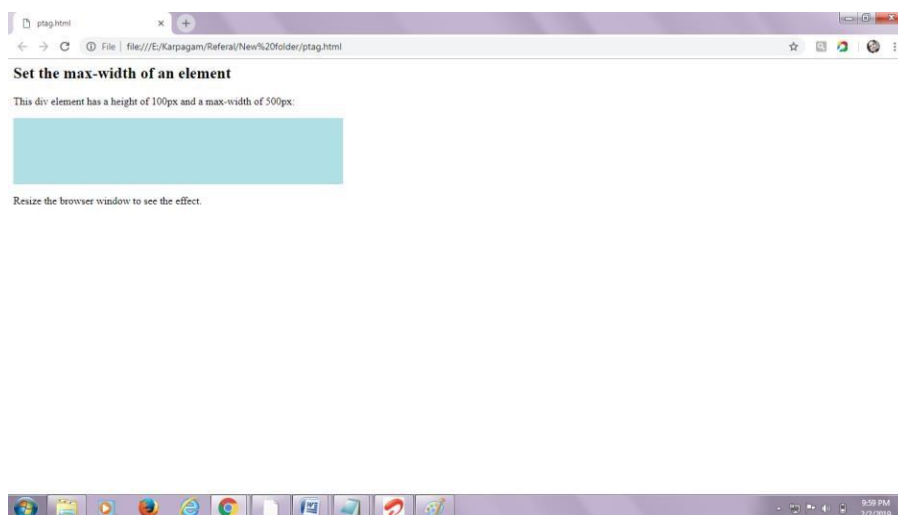
The problem with the <div> above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows.

### Example Program

```
<html>
<head>
<style>
div {
    max-width: 500px;
    height: 100px;
    background-color: powderblue;
}
</style>
</head>
<body>
<h2>Set the max-width of an element</h2>
<p>This div element has a height of 100px and a max-width of 500px:</p>
<div></div>
<p>Resize the browser window to see the effect.</p>
</body>
</html>
```

### Output



## 2.15 Text Flow

The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. It can be clipped, display an ellipsis (...), or display a custom string.

Both of the following properties are required for text-overflow:

- white-space: nowrap;
- overflow: hidden;

### Syntax

text-overflow: clip|ellipsis|*string*|initial|inherit;

### Property Values

Value	Description
clip	Default value. The text is clipped and not accessible
ellipsis	Render an ellipsis ("...") to represent the clipped text
<i>string</i>	Render the given string to represent the clipped text
initial	Sets this property to its default value. <a href="#">Read about <i>initial</i></a>
inherit	Inherits this property from its parent element. <a href="#">Read about <i>inherit</i></a>

### Example Program

```
<html>
<head>
<style>
div.a {
  white-space: nowrap;
  width: 100px;
  overflow: hidden;
  text-overflow: ellipsis;
  border: 1px solid #000000;
}
```

```
div.a:hover {  
    overflow: visible;  
}  
  
</style>  
</head>  
<body>  
<h1>The text-overflow Property</h1>  
<p>Hover over the div below, to see the entire text.</p>  
<div class="a">This is some long text that will not fit in the box.</div>  
</body>  
</html>
```

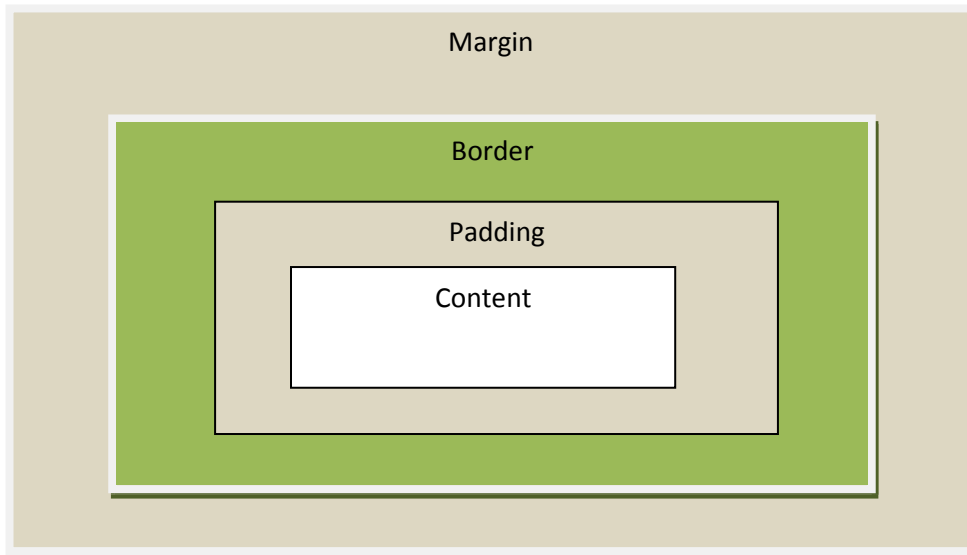
### Output



## 2.16 Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

**Explanation of the different parts:**

**Content** - The content of the box, where text and images appear

**Padding** - Clears an area around the content. The padding is transparent

**Border** - A border that goes around the padding and content

**Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

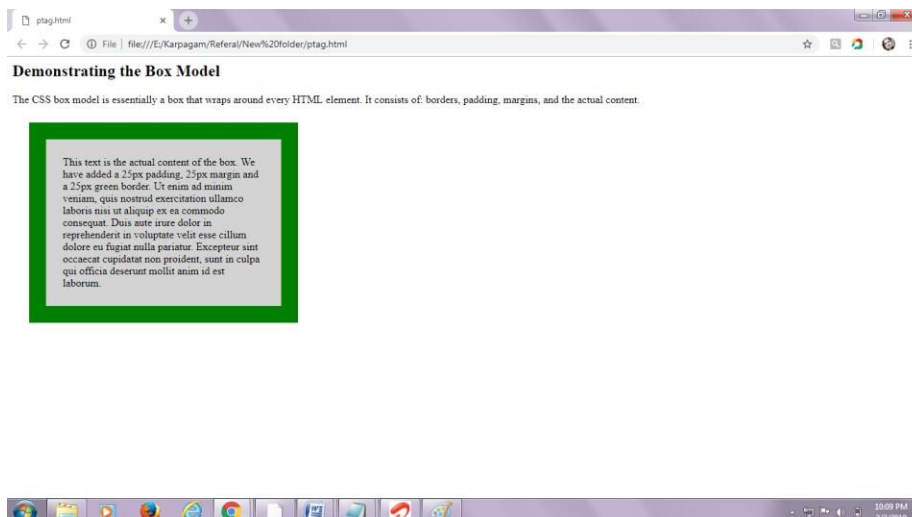
**Example Program**

```
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
```



```
border: 25px solid green;
padding: 25px;
margin: 25px;
}
</style>
</head>
<body>
<h2>Demonstrating the Box Model</h2>
<p>The CSS box model is essentially a box that wraps around every HTML element. It
consists of: borders, padding, margins, and the actual content.</p>
<div>This text is the actual content of the box. We have added a 25px padding, 25px margin
and a 25px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate
velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div>
</body>
</html>
```

## Output



**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed University Established Under Section 3 of UGC Act 1956)  
Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

**Department of Commerce (CA)**

**SUBJECT: INTERNET AND WEB DESIGNING**

**SEMESTER : VI**

**SUBJECT CODE: 17CCU601A**

**CLASS: III B.COM CA**

**UNIT:II**

<b>S. N O</b>	<b>QUESTIONS</b>	<b>OPTION 1</b>	<b>OPTION 2</b>	<b>OPTION 3</b>	<b>OPTION 4</b>	<b>ANSWER</b>
1	_____HTML attribute is used to define inline styles.	Class	Styles	Style	Font	Style
2	The_____CSS property controls the text size.	text-style	font-size	text-size	font-style	font-size
3	The_____CSS Property is used to give border around an image.	border-color	border-decoration	border-style	border-line	border-style
4	The_____property is used to display border around a cell without any content.	empty-cell	blank-cell	noncontent-cell	void-cell	empty-cell
5	If we want to wrap a block of text around an image, which css property will we use?	wrap	push	float	align	float
6	_____value is used to show an arrow as a cursor.	Pointer	Default	Arrow	Arr	Default
7	The_____CSS property is used to not allow floating point to DIV.	margin	clear	float	padding	clear

8	Suppose we want to arrange five nos of divs so that div4 is placed above div1. Now, which property of css will we use to control the order of stack?	d-index	s-index	x-index	z-index	z-index
9	DHTML stands for_____.	Domain Hyper Text Markup Language	Dual Hyper Text Markup Language		Dynamic Hyper Text Markup Language	Dynamic Hyper Text Markup Language
10	The_____CSS selector is used to define style for an unique element.	id	text	name	class	id
11	The <map> tag defines a_____.	image map	map	map image	images	map
12	Which one of the following is difficult to read?	Background color, text color	text color	improper statements	all the above.	Background color, text color
13	_____are used to select different kinds of user input.	HTML forms	HTML table	HTML lists	None	HTML forms
14	In HTML, Comments lines are denoted by_____.	<!-- & -->	<!-- & - >	!-- & ->	! & !	<!-- & -->
15	How can you make an e-mail link?	<u>&lt;a href=mailto:XXX@YYY&gt;</u>	<a href = "XXX@YYY">	<mail>XXX@YYY</mail>	<mail href ="XXX@YYY">	<u>&lt;a href=mailto:XXX@YYY&gt;</u>
16	Images are embedded within a web document page with the use of_____.	input	Img tag	text file	http	Img tag
17	Does the image tag has an end tag?	random	yes	no	optional	no
18	The html element defines the_____of the document.	optional	Whole	Part	Body	Whole
21	What is the correct HTML for making a text area?	input	<input type = "text area">	<input type = "text box">	<text area>	<input type = "text area">

22	How many different layers are in HTML?	6	2	3	4	2
23	The event that captures one or more event and routes to object is_____.	event	Capture event	Route event	Release event	Capture event
24	For every web page we create, it must have_____.	<body>	<html>	<head>	<title>	<html>
25	Which of these functions will encloses the entire HTML document?	<!-->	<html>...</html>	<body>...</body>	<hr>	<html>...</html>
26	When the mouse is over the image, the function refers to_____.	high	change-high	change-low	change	change-high
27	When the mouse leaves the image area, the function refers to_____.	high	change-high	change-low	change	change-low
28	Html Attributes are always specified in the _____ tag.	Start	End	Paragraph	Comment line	Start
29	Attribute values are_____.	Case sensitive	Not case sensitive	Additional information	optional	Case sensitive
30	The_____tag defines how to divide the window into frames.	Frameset	Table	Frame	Form	Frameset
31	The_____tag is used to get response form the user at runtime.	Frameset	Table	Frame	Form	Form
32	To avoid empty cells we specify _____.	(&nbsp;)	Hyphen	Null character	0	(&nbsp;)
33	A form is defined with the_____.	<html>	<form>	<input>	<text area>	<form>
34	The image tag ends with the _____	<img>	</img>	</image>	no close tag	no close tag
35	What is the correct html for adding a background color?	<body color="yellow">	<Background yellow</background>	<body bgcolor="yellow">	<image>	<body bgcolor="yellow">
36	_____are used to select different kinds of user input.	HTML forms	HTML table	HTML lists	None	HTML forms
37	_____is not the type of input in html forms.	text area	button	list	images	images

38	_____is tag is used to give details about the webpage.	meta	image	title	frame	meta
39	The frames tag should have atleast _____ divisions.	1	2	3	4	2
40	The frame tag is used to divide the webpage into _____	rows and columns	header and footer	tables	forms	rows and columns
41	The page which is to be printed in the frame is given in _____parameter.	src	img	href	ref	src
42	What is the correct tag for inserting a line break?	 	<lb>	<break>	<none>	 
43	_____element will follow the head tag.	Body	Title	hr	br	title
44	Table values are displayed in a _____format.	Tabular	Frames	rows	columns	Tabular
45	For a <p> tag the end tag is required or not?	Required	Not Required	Optional	must	Optional
46	Choose the correct HTML tag to make a text italic?	<ii>	<i>	<Italic>	<it>	<i>
47	Choose the correct HTML tag to make a text underline?	<ul>	<u>	<und>	<underli ne>	<u>
48	The base element specifies an absolute _____.	URL address	IP address	TCP address	href	URL address
49	How can you make a list that lists the items with bullets?	<List>	<ul>	<ol>	<DI>	<ul>
50	Passing a Boolean value _____the sound playing.	Start	begins	ends	pause	begins
51	Which of these attributes will specify “location of an objects implementation?	code base	class id	data	code type	class id
52	Which of the following options are not object tag attributes?	code base	class id	code base	code type	code base
53	which of the following is not a direct animation controls?	code base	Structured graphics	Path	Sprite	code base
54	Each IRC conversation is carried on a _____.	Channel	Wires	Cables	Circuits	Channel

55	The Description of web search engines is _____.	Search the web	Receives messages	Gathering information	All the above	Search the web
56	Menu – Based information is a_____.	Gopher	Archie	Usenet	Mail	Gopher
57	Multi – linked information is a_____.	The web	Mailing lists	Talk facility	Using of e-mail	The web
58	_____ is a hardware device.	Modem	Cable	Wires	None	Modem
59	_____is the process of converting from a digital to Analog format.	Modulation	Demodulation	Modem	Data	Modulation
60	_____is the process of converting from Analog to digital format.	Modulation	Demodulation	Modem	Data	Demodulation

**UNIT-III****SYLLABUS**

**Introduction to Java Script:** Operators – Arithmetic Operators – Precedence of Operators – Relational Operators – Control Structures – Assignment Operators – Increment and Decrement Operators – For Loops – Switch – Do While – Break – Continue – Arrays – Functions.

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

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

**Java Scrip Syntax**

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

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

The `<script>` tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

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

The script tag takes two important attributes –

- **Language** – This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** – This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

So your JavaScript segment will look like –

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

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

This function can be used to write text, HTML, or both. Take a look at the following code.

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

This code will produce the following result –

Hello World!

### **Whitespace and Line Breaks**

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

### **Semicolons are Optional**

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language="javascript" type="text/javascript">  
<!--  
  var l=10
```



```
var2=20  
//-- >  
</script>
```

### Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

### Comments in JavaScript

JavaScript supports both C-style and C++-style comments, Thus –

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

### JavaScript Datatypes

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types –

- **Numbers**, eg. 123, 120.50 etc.
- **Strings** of text e.g. "This text string" etc.
- **Boolean** e.g. true or false.

JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as **object**. We will cover objects in detail in a separate chapter.

JavaScript does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values. JavaScript represents numbers using the 64-bit floating-point format defined by the IEEE 754 standard.

### JavaScript Variables

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
<!--
    var money, name;
    var name;
//-- >
</script>
```

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named **money** and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
<!--
    var name="Ali";
    var money;
    money=2000.50;
//-- >
</script>
```

### Operators

Let us take a simple expression **4 + 5 is equal to 9**. Here 4 and 5 are called **operands** and '+' is called the **operator**. JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators

- Assignment Operators
- Increment/Decrement Operators

**Arithmetic Operators and Increment/Decrement Operators**

JavaScript supports the following arithmetic operators –

Assume variable A holds 10 and variable B holds 20, then –

Sr.No.	Operator & Description
1	<b>+</b> ( <b>Addition</b> ) Adds two operands <b>Ex:</b> A + B will give 30
2	<b>-</b> ( <b>Subtraction</b> ) Subtracts the second operand from the first <b>Ex:</b> A - B will give -10
3	<b>*</b> ( <b>Multiplication</b> ) Multiply both operands <b>Ex:</b> A * B will give 200
4	<b>/</b> ( <b>Division</b> ) Divide the numerator by the denominator <b>Ex:</b> B / A will give 2
5	<b>%</b> ( <b>Modulus</b> ) Outputs the remainder of an integer division <b>Ex:</b> B % A will give 0
6	<b>++</b> ( <b>Increment</b> ) Increases an integer value by one <b>Ex:</b> A++ will give 11
7	<b>--</b> ( <b>Decrement</b> ) Decreases an integer value by one <b>Ex:</b> A-- will give 9

**Example**

The following code shows how to use arithmetic operators in JavaScript.

```
<html>
<body>
  <script type="text/javascript">
    <!--
    var a =33;
    var b =10;
```

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: III

BATCH-2017-20

```
var c="Test";
var linebreak="<br/>";
var result;
document.write("a+b=");
result =a+b;
document.write(result);
document.write(linebreak);
document.write("a/b = ");
result=a/b;
document.write(result);
document.write(linebreak);
document.write("a%b =");
result =a%b;
document.write(result);
document.write(linebreak);
document.write("a+b+c =");
result=a+b+c;
document.write(result);
document.write(linebreak);
a=++a;
document.write("++a = ");
result=++a;
document.write(result);
document.write(linebreak);
b=--b;
document.write(result);
document.write(linebreak);
//-- >
</script>
</body>
</html>
```

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA  
COURSE CODE: 17CCU601A

COURSE NAME: INTERNET AND WEB DESIGNING  
UNIT: III  
BATCH-2017-20

### Output

```
a+b=4  
a-b=23  
a/b=3.3  
a%b=3  
a+b+c=43Test  
++a=35  
--b=8
```

### Precedence of Operators

Precedence of operators gives the order of evaluating an expression with n number of operators and the order in which the evaluation of an expression. The following table define the precedence of the operators in JavaScript,

Value	Operator	Description	Example
20	()	Expression grouping	(3 + 4)
19	.	Member	person.name
19	[]	Member	person["name"]
19	()	Function call	myFunction()
19	new	Create	new Date()
17	++	Postfix Increment	i++
17	--	Postfix Decrement	i--
16	++	Prefix Increment	++i
16	--	Prefix Decrement	--i
16	!	Logical not	!(x==y)
16	typeof	Type	typeof x
15	**	Exponentiation (ES7)	10 ** 2
14	*	Multiplication	10 * 5
14	/	Division	10 / 5
14	%	Division Remainder	10 % 5
13	+	Addition	10 + 5
13	-	Subtraction	10 - 5
12	<<	Shift left	x << 2

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: III

BATCH-2017-20

12	>>	Shift right	x >> 2
12	>>>	Shift right (unsigned)	x >>> 2
11	<	Less than	x < y
11	<=	Less than or equal	x <= y
11	>	Greater than	x > y
11	>=	Greater than or equal	x >= y
11	in	Property in Object	"PI" in Math
11	instanceof	Instance of Object	instanceof Array
10	==	Equal	x == y
10	===	Strict equal	x === y
10	!=	Unequal	x != y
10	!==	Strict unequal	x !== y
9	&	Bitwise AND	x & y
8	^	Bitwise XOR	x ^ y
7		Bitwise OR	x   y
6	&&	Logical AND	x && y
5		Logical OR	x    y
4	? :	Condition	? "Yes" : "No"
3	+=	Assignment	x += y
3	+=	Assignment	x += y
3	-=	Assignment	x -= y
3	*=	Assignment	x *= y
3	%=	Assignment	x %= y
3	<<=	Assignment	x <<= y
3	>>=	Assignment	x >>= y
3	>>>=	Assignment	x >>>= y
3	&=	Assignment	x &= y
3	^=	Assignment	x ^= y
3	=	Assignment	x  = y
2	yield	Pause Function	yield x
1	,	Comma	5 , 6

### Relational Operators

JavaScript supports the following relational operators –

Assume variable A holds 10 and variable B holds 20, then –

Sr.No.	Operator & Description
1	<b>&amp;&amp; (Logical AND)</b> If both the operands are non-zero, then the condition becomes true. <b>Ex:</b> (A && B) is true.
2	<b>   (Logical OR)</b> If any of the two operands are non-zero, then the condition becomes true. <b>Ex:</b> (A    B) is true.
3	<b>! (Logical NOT)</b> Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. <b>Ex:</b> ! (A && B) is false.

### Example

Try the following code to learn how to implement Logical Operators in JavaScript.

```
<html>
<body>
  <script type="text/javascript">
    <!--
    var a=true;
    var b=false;
    var linebreak="<br/>";
    document.write("(a && b) =>");
    result=(a && b);
    document.write(result);
    document.write(linebreak);
    document.write("(a || b) =>");
    result=(a || b);
    document.write(result);
    document.write(linebreak);
    document.write("! (a && b) =>");
    result=! (a && b);
    document.write(result);
    document.write(linebreak);
    //-- >
  </script>
</body>
</html>
```

**Output**

(a && b)=>false

(a || b)=>true

!(a && b)=>true

**Control Structures****Assignment Operators**

JavaScript supports the following assignment operators –

Sr.No.	Operator & Description
1	<b>= (Simple Assignment)</b> Assigns values from the right side operand to the left side operand <b>Ex:</b> C = A + B will assign the value of A + B into C
2	<b>+= (Add and Assignment)</b> It adds the right operand to the left operand and assigns the result to the left operand. <b>Ex:</b> C += A is equivalent to C = C + A
3	<b>-= (Subtract and Assignment)</b> It subtracts the right operand from the left operand and assigns the result to the left operand. <b>Ex:</b> C -= A is equivalent to C = C - A
4	<b>*= (Multiply and Assignment)</b> It multiplies the right operand with the left operand and assigns the result to the left operand. <b>Ex:</b> C *= A is equivalent to C = C * A
5	<b>/= (Divide and Assignment)</b> It divides the left operand with the right operand and assigns the result to the left operand. <b>Ex:</b> C /= A is equivalent to C = C / A
6	<b>%= (Modules and Assignment)</b> It takes modulus using two operands and assigns the result to the left operand. <b>Ex:</b> C %= A is equivalent to C = C % A

**Example**

Try the following code to implement assignment operator in JavaScript.

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



```
<script type="text/javascript">
<!--
var a=3;
var b=10;
var linebreak="<br/>";
document.write("Value of a=> (a=b)=>");
result = (a=b);
document.write(result);
document.write(linebreak);
document.write("Value of a=> (a+=b)=>");
result = (a+=b);
document.write(result);
document.write(linebreak);
document.write("Value of a=> (a -= b)=>");
result = (a -= b);
document.write(result);
document.write(linebreak);
document.write("Value of a=> (a *= b)=>");
result = (a *= b);
document.write(result);
document.write(linebreak);
document.write("Value of a=> (a /= b)=>");
result = (a /= b);
document.write(result);
document.write(linebreak);
document.write("Value of a=> (a %= b)=>");
result = (a %= b);
document.write(result);
document.write(linebreak);
//-- >
</script>
```

</body>

</html>

### **Output**

Value of a => (a = b) => 10  
Value of a => (a += b) => 20  
Value of a => (a -= b) => 10  
Value of a => (a \*= b) => 100  
Value of a => (a /= b) => 10  
Value of a => (a %= b) => 0

### **For Loop**

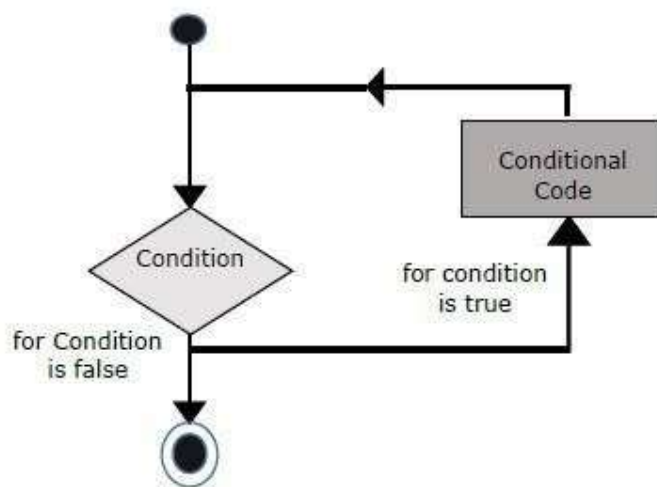
The 'for' loop is the most compact form of looping. It includes the following three important parts –

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

### Flow Chart

The flow chart of a **for** loop in JavaScript would be as follows –



### Syntax

The syntax of **for** loop in JavaScript is as follows –

```
for (initialization; test condition; iteration statement)
{
    Statement(s) to be executed if test condition is true
}
```

### Parameter Values

Parameter	Description
initialization	Optional. Executed before the loop (the code block) starts. Normally this statement is used to initialize a counter variable. To initiate multiple values, separate each value with a comma. <b>Note:</b> This parameter can be omitted. However, do not omit the

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: III

BATCH-2017-20

	semicolon ";"
test condition	<p>Optional. Defines the condition for running the loop (the code block). Normally this statement is used to evaluate the condition of the counter variable. If it returns true, the loop will start over again, if it returns false, the loop will end.</p> <p><b>Note:</b> This parameter can be omitted. However, do not omit the semicolon ";". Also, if you omit this parameter, you must provide a break inside the loop. Otherwise the loop will never end, which will crash your browser</p>
iteration statement	<p>Optional. Executed each time after the loop (the code block) has been executed. Normally this statement is used to increment or decrement the counter variable.</p> <p><b>Note:</b> This parameter can be omitted (e.g. to increase/decrease values inside the loop)</p>

### Example program

```
<html>
<body>
  <script type = "text/javascript">
    var count;
    document.write("Starting Loop" + "<br />");
    for(count = 0; count < 10; count++)
    {
      document.write("Current Count : " + count );
      document.write("<br />");
    }
    document.write("Loop stopped!");
  </script>
</body>
</html>
```

### Output

Starting Loop  
Current Count : 0  
Current Count : 1  
Current Count : 2  
Current Count : 3  
Current Count : 4  
Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

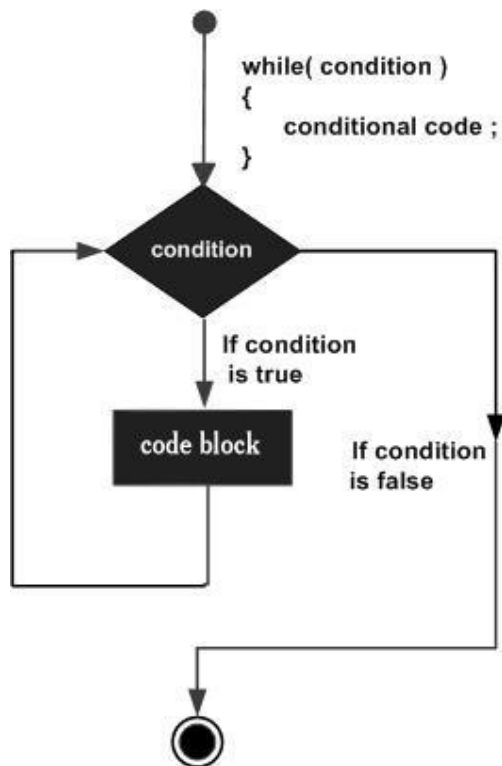
Loop stopped!

### Switch Statement

The most basic loop in JavaScript is the **while** loop which would be discussed in this chapter. The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false**, the loop terminates.

### Flow Chart

The flow chart of **while loop** looks as follows –



### Syntax

The syntax of **while loop** in JavaScript is as follows –

```
while (expression)
```

```
{
```

```
    Statement(s) to be executed if expression is true
```

```
}
```

**Example Program**

```
<html>
  <body>
    <script type = "text/javascript">
      var count = 0;
      document.write("Starting Loop ");

      while (count < 10) {
        document.write("Current Count : " + count + "<br />");
        count++;
      }
      document.write("Loop stopped!");
    </script>
  </body>
</html>
```

**Output**

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

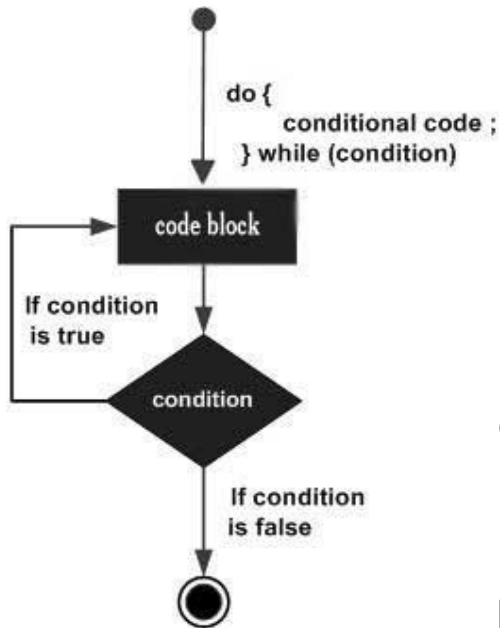
Loop stopped!

**Do While**

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

**Flow Chart**

The flow chart of a **do-while** loop would be as follows –

**Syntax**

The syntax for **do-while** loop in JavaScript is as follows

```
do {  
    Statement(s) to be executed;  
} while (expression);
```

**Example Program:**

```
<html>  
<body>  
    <script type = "text/javascript">  
        var count = 0;  
        document.write("Starting Loop" + "<br />");  
        do {  
            document.write("Current Count : " + count + "<br />");  
            count++;  
        }  
    </script>  
</body>  
</html>
```

```
        while (count < 5);  
        document.write ("Loop stopped!");  
    </script>  
</body>  
</html>
```

**Output:**

Starting Loop  
Current Count : 0  
Current Count : 1  
Current Count : 2  
Current Count : 3  
Current Count : 4  
Loop Stopped!

**Break and Continue statement**

Break and Continue statement are also known as unconditional branching statements.

The break statement "jumps out" of a loop.

The continue statement "jumps over" one iteration in the loop.

**The Break Statement**

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch() statement.

- The break statement can also be used to jump out of a loop.
- The break statement breaks the loop and continues executing the code after the loop.

**Example Program**

```
<html>  
<body>  
<p>A loop with a break.</p>  
<p id="demo"></p>  
<script>  
var text = "";  
var i;  
for (i = 0; i < 10; i++) {
```

```
    if (i === 3) { break; }  
    text += "The number is " + i + "<br>";  
}  
</script>  
</body>  
</html>
```

### **Output**

A loop with a break.

The number is 0  
The number is 1  
The number is 2

### **The Continue Statement**

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

### **Example Program:**

This example skips the value of 3:

```
<html>  
<body>  
<p>A loop which will skip the step where i = 3.</p>  
<p id="demo"></p>  
<script>  
var text = "";  
var i;  
for (i = 0; i < 10; i++) {  
    if (i === 3) { continue; }  
    text += "The number is " + i + "<br>";  
}  
</script>  
</body>  
</html>
```

### **Output:**

A loop which will skip the step where i = 3.



The number is 0

The number is 1

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 JavaScript arrays are used to store multiple values in a single variable.

### **Arrays**

JavaScript arrays are used to store multiple values in a single variable.

#### **Creating an Array**

Using an array literal is the easiest way to create a JavaScript Array.

#### **Syntax:**

```
var array_name = [item1, item2, ...];
```

An array is a special variable, which can hold more than one value at a time.

#### **Example:**

```
<html>
```

```
<body>
```

```
<h2>JavaScript Arrays</h2>
```

```
<p id="demo"></p>
```

```
<script>
```

```
var cars = ["Saab", "Volvo", "BMW"];
```

```
document.getElementById("demo").innerHTML = cars;
```

```
</script>
```

```
</body>
```

```
</html>
```

## Output

## JavaScript Arrays

Saab, Volvo, BMW

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Saab";
```

```
var car2 = "Volvo";
```

```
var car3 = "BMW";
```

An array can hold many values under a single name, and you can access the values by referring to an index number.

You access an array element by referring to the index number.

This statement accesses the value of the first element in cars:

```
cars[1] = "Saab";
```

```
cars[2] = "Volvo";
```

```
cars[3 ]= "BMW";
```

### Array Properties

---

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

Sr.No.	Property & Description
1	<u>constructor</u> Returns a reference to the array function that created the object.
2	<b>index</b> The property represents the zero-based index of the match in the string
3	<b>input</b> This property is only present in arrays created by regular expression matches.
4	<u>length</u> Reflects the number of elements in an array.
5	<u>prototype</u> The prototype property allows you to add properties and methods to an object.

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA  
COURSE CODE: 17CCU601A

COURSE NAME: INTERNET AND WEB DESIGNING  
UNIT: III  
BATCH-2017-20

In the following sections, we will have a few examples to illustrate the usage of Array properties.

### Array Methods

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

Sr.No.	Method & Description
1	<u>concat</u>  Returns a new array comprised of this array joined with other arrays and/or values.
2	<u>every</u>  Returns true if every element in this array satisfies the provided testing function.
3	<u>filter</u>  Creates a new array with all of the elements of this array for which the provided filtering function returns true.
4	<u>forEach</u>  Calls a function for each element in the array.
5	<u>indexOf</u>  Returns the first least index of an element within the array equal to the specified value, or -1 if none is found.
6	<u>join</u>  Joins all elements of an array into a string.
7	<u>lastIndexOf</u>  Returns the last greatest index of an element within the array equal to the specified value, or -1 if none is found.
8	<u>map</u>  Creates a new array with the results of calling a provided function on every element in this array.
9	<u>pop</u>  Removes the last element from an array and returns that element.
10	<u>push</u>  Adds one or more elements to the end of an array and returns the new length of the array.
11	<u>reduce</u>  Apply a function simultaneously against two values of the

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA

COURSE NAME: INTERNET AND WEB DESIGNING

COURSE CODE: 17CCU601A

UNIT: III

BATCH-2017-20

	array from left to right as to reduce it to a single value.
12	<u>reduceRight</u>  Apply a function simultaneously against two values of the array from right to left as to reduce it to a single value.
13	<u>reverse</u>  Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
14	<u>shift</u>  Removes the first element from an array and returns that element.
15	<u>slice</u>  Extracts a section of an array and returns a new array.
16	<u>some</u>  Returns true if at least one element in this array satisfies the provided testing function.
17	<u>toSource</u>  Represents the source code of an object
18	<u>sort</u>  Sorts the elements of an array
19	<u>splice</u>  Adds and/or removes elements from an array.
20	<u>toString</u>  Returns a string representing the array and its elements.
21	<u>unshift</u>  Adds one or more elements to the front of an array and returns the new length of the array.

### Functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

### JavaScript Function Syntax

```
function name(parameter1, parameter2, parameter3) {
```

// code to be executed

}

- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)
- The code to be executed, by the function, is placed inside curly brackets: {}
- Function parameters are listed inside the parentheses () in the function definition.
- Function arguments are the values received by the function when it is invoked.
- Inside the function, the arguments (the parameters) behave as local variables.
- A Function is much the same as a Procedure or a Subroutine, in other programming languages.

### **Function Invocation**

The code inside the function will execute when "something" invokes (calls) the function, as

1. When an event occurs (when a user clicks a button)
2. When it is invoked (called) from JavaScript code
3. Automatically (self invoked)

### Function Return

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a **return value**. The return value is "returned" back to the "caller":

```
<html>
<body>
<h2>JavaScript Functions</h2>
<p>This example calls a function which performs a calculation and returns the result:</p>
<p id="demo"></p>
<script>
var x = myFunction(4, 3);
document.getElementById("demo").innerHTML = x;
function myFunction(a, b) {
  return a * b;
}
```

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

### **Output**

### **JavaScript Functions**

This example calls a function which performs a calculation and returns the result:

12

KAHE

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

**Department of Commerce (CA)**

**SUBJECT: INTERNET AND WEB DESIGNING**

**SEMESTER : VI**

**SUBJECT CODE: 17CCU601A**

**CLASS : III**

**UNIT:III**

**B.COM CA**

<b>S.NO</b>	<b>QUESTIONS</b>	<b>OPTION 1</b>	<b>OPTION 2</b>	<b>OPTION 3</b>	<b>OPTION 4</b>	<b>ANSWER</b>
1	JavaScript is a_____language.	Applicati on	Scripting	Programm ing	compilin g	Scripting
2	JavaScript is a_____side scripting language.	server	client	browser	ISP	browser
3	JavaScript is designed to_____.	style HTML page	add interactivit y to HTML page	perform server side scripting	execute query related to DB on server	add interactivit y to HTML page
4	JavaScript can be written_____.	directly into HTML page	directly into JS file	directly on the server script	included separatel y to HTML	directly into HTML page
5	JavaScript code is placed inside_____HTML element.	<scriptin g>	<javascript >	<script>	<js>	<script>
6	Function “myFunction” is called as_____.	myFuncti on()	call myFunctio	call function	myFuncti on	myFunctio n()

			n()	myFunction()		
7	How to write an IF statement in JavaScript?	if(i==5)	if i=5	if i==5 then	if i=5 then	if(i==5)
8	How does a FOR loop start?	for i = 1 to 5	for (i = 0; i <= 5)	for (i <= 5; i++)	for (i = 0; i <= 5; i++)	for (i = 0; i <= 5; i++)
9	How do you find the number with the highest value of x and y?	Math. Ceil(x, y)	top(x, y)	Math.max(x, y)	ceil(x, y)	Math.max(x, y)
10	Which of the following is not an assignment operator?	= =	+=	-=	*=	= =
11	Executable single line of script in JavaScript is known as _____.	Statement	Breakpoint	Line	Code	Statement
12	JavaScript statements are executed by_____.	Server	Compiler	JVM	Browser	JVM
13	Java Statement terminated by_____.	Slash	Comma	Semicolon	Full Stop	Semicolon
14	_____statement is used to declare variable in JavaScript.	Assignment	Declaration	Executable	Conditional	Declaration
15	Variable can hold_____value at a time.	Double	Triple	Single	Multiple	Single
16	Integer Variable is declared using_____syntax in JavaScript.	var num	Integer num;	int num;	integer num;	var num
17	The_____method is used to return the value of the last element inserts in an array	last()	get()	pop()	push()	pop()
18	The_____statement is used to print a line using JavaScript.	window. write()	document. write()	navigator. write()	window. write()	document. write()
19	The_____function is used to print a string in italic.	fixed()	fontcolor() )	fontsize()	italic()	italic()
20	The_____function of array object is used to reduce two array values into a single array value.	reduce()	reduceRight()	pop()	push()	reduce()
21	The_____method is used to insert an element into an array.	last()	get()	pop()	push()	push()
22	The equal comparison operator in JavaScript is_____.	<>	!=	"_"	^=	==



23	To add number to sum, you write (Note: JavaScript is case_sensitive).	number += sum;	number = sum + number;	sum=num ber	sum += number;	sum += number;
24	Which of the following expression results in a value 1?	2 % 1	15 % 4	9%9	37 % 6	37 % 6
25	Which of the Boolean expressions below is incorrect?	(true) && (3 ==> 4)	!(x > 0) && (x > 0)	!(x > 1) && (x > 0)	(x !== 0)    (x = 0)	(true) && (3 ==> 4)
26	Which of the following is the correct expression that evaluates to true if the number x is between 1 and 100 or the number is negative?	1 < x < 100 && x < 0	((x < 100) && (x > 1))    (x < 0)	1 < x < 100 && x < 10	(1 > x > 100)    (x < 0)	((x < 100) && (x > 1))    (x < 0)
27	The "less than or equal to" comparison operator in JavaScript is_____.	<	<=	>	<<	<=
28	Suppose x=10 and y=10 what is x after evaluating the expression (y > 10) & (x++ > 10).	9	10	11	12	11
29	What is the representation of the third element in an array called a?	a[2]	a(2)	a{2}	a(3)	a[2]
30	the length of a string by calling the_____method	strlen()	len()	length()	none	length( )
31	Unary operators require_____Operand	One	Two	Three	Four	One
32	Binary Operator requires_____Operand	One	Two	Three	Four	Two
33	In case of strings '+' is used to _____	It seperates the first operand and the second operand	It reduces the data size	It increase the data size	It combines the first operand and the second operand	It combines the first operand and the second operand
34	The "<<" is known as	insertion operator	extraction operator	lesser	greater	insertion operator
35	Test is performed at the_____of the for loop.	top	middle	end	program terminates	top
36	The statements of javascript has multiple statements that	script tag	title tag	hr tag	font tag	script tag

	should be enclosed within					
37	The parameters of script tag are _____ and _____.	language and type	language and range	type and char	language and char	language and type
38	The value of ++I is	i=i+1	i=1+i	i=1+1	i=1-i	i=1+i
39	The value of I-- is	i=i-1	1=1-i	i=1-1	i=i-i	i=i-1
40	Which is not a increment decrement operator?	i++	i—	--i	1+i	1+i
41	Which loop executes the statement and then checks on the condition?	while	for	do while	for next	do while
42	In number of time the statement executed by do while is _____	equal to while	1+ while	1-while	2+while	1+ while
43	_____ is not a unconditional branching statement	break	continue	goto	switch	switch
44	Find the odd man out?	Start	<th>	<td>	<hr>	start
45	The amount of space between the cells in the table refers to _____.	cell spacing	cell padding	table	cells	cell spacing
46	Which of the following options are not object tag attributes?	cell spacing	Quote	Single quote	none	cell spacing
47	which of the following is not a direct animation controls?	cell spacing	quote	Single quote	none	cell spacing
48	_____ is an example of method call	mul();	mul;	mul	mul()	mul();
49	_____ is type of variables can be given as arrays.	char	numbers	string	all types of data	all types of data
50	Which type of data type is not declared as var?	String	array	numbers	character s	String
51	Data and time can be declared as _____ data type	String	array	var	date and time	date and time
52	Functions are known as _____.	sub program	main program	arrays	datas	sub program
53	Functions should have _____.	declaratio n	call of function	call of values	call of data	Declaratio n
54	Array are stored in _____ memory.	random	same	in sequence	as per user data	same

55	The_____method is used to return the value of the first element inserts in an array.	last()	get()	pop()	push()	last()
56	The_____method is used to get input values.	last()	get()	pop()	push()	get()
57	The alternate language of javascript is _____ JavaScript was developed by_____	jscript Sun Microsystem	j2ee Microsoft	jstring IBM	jswing Netscape Navigator	jscript Sun Microsystem
58						
59	JavaScript is_____language	scripting	markup	extended	standard	scripting
60	JavaScript is used to design a_____.	Web page	Image	Frame	Flex	Web page

**UNIT-IV- Active Server Page (ASP)**

**SYLLABUS**

Introduction – How ASP work – five objects of ASP – Client Side Scripting Vs Server Side Scripting – Server Side  
ActiveX component – Session Tracking and Cookies – file system objects

**Introduction:**

Active Server Pages were introduced by Microsoft in 1996 as a downloadable feature of Internet Information Server(IIS). An Active Server Page allows code written in the JavaScript or VBScript languages to be embedded within the HTML tags of a Web page and executed on the Web server. There are great advantages to this, not the least of which is security. Since your code is executed on the Web server, only HTML tags are sent to the browser. The result is that the ASP code is invisible to the end user.

The server-side script concept is that it allows things like database connections to be made from the Web server rather than from the client. Therefore, any special configurations that might need to be set up, like ODBC data sources, only have to exist on the server. Of course, before you can create an Active Server Page (ASP), you'll need to look at the software requirements.

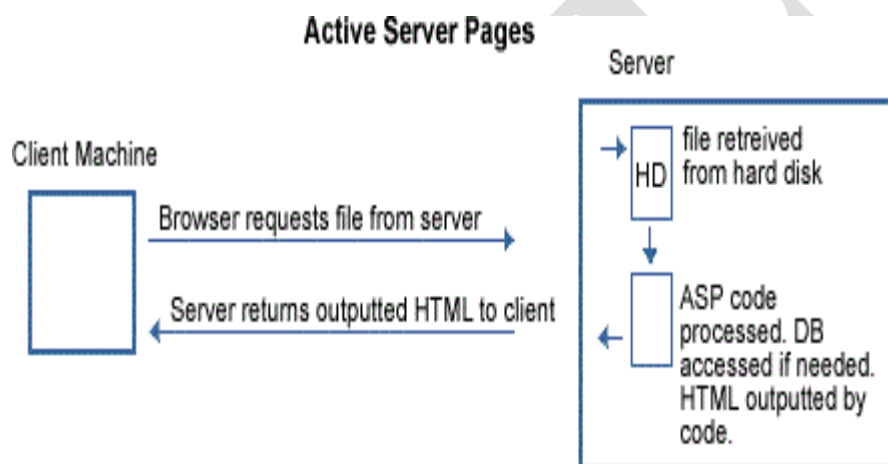
There are products to allow ASPs to be used on just about any Web server out there. This fact makes using ASPs that much more attractive because you aren't limited in the choice of hardware, operating system, or Web server to host your Web pages. As you can see, there are even ASP-compatibility products for the iSeries.

Active Server Page support to non-Microsoft Web servers. It has a more complete list of ASP compatibility products and the operating systems and Web servers they run on.

IIS or Personal Web Server, no additional software is required to support Active Server Pages. To allow a user to access an ASP, the ability to do so must be enabled on the IIS server. This is done by selecting "Scripts" or "Execution (Including Scripts)" from the "Home Directory" tab of the Properties window for your Web site.

### **What is ASP?**

- ASP stands for Active Server Pages
- ASP is a program that runs inside IIS
- IIS stands for Internet Information Services
- IIS comes as a free component with Windows 2000
- IIS is also a part of the Windows NT 4.0 Option Pack
- The Option Pack can be downloaded from Microsoft
- PWS is a smaller - but fully functional - version of IIS
- PWS can be found on your Windows 95/98 CD



### **ASP Compatibility**

- ASP is a Microsoft Technology
- To run IIS you must have Windows NT 4.0 or later
- To run PWS you must have Windows 95 or later
- ChiliASP is a technology that runs ASP without Windows OS
- InstantASP is another technology that runs ASP without Windows

### **What is an ASP File?**

- An ASP file is just the same as an HTML file
- An ASP file can contain text, HTML, XML, and scripts
- Scripts in an ASP file are executed on the server
- An ASP file has the file extension ".asp"

**Advantages of Using Asp:**

- Minimizes network traffic by limiting the need for the browser and server to talk to each other.
- Makes for quicker loading time since HTML pages are only downloaded.
- Allows running programs in languages that are not supported by the browser.
- Can provide the client with data that does not reside on the client's machine.
- Provides improved security measures since the script cannot be viewed by the browser.

**How Does ASP Differ from HTML?**

- When a browser requests an HTML file, the server returns the file
- When a browser requests an ASP file, IIS passes the request to the ASP engine. The ASP engine reads the ASP file, line by line, and executes the scripts in the file. Finally, the ASP file is returned to the browser as plain HTML.

ASP	HTML
ASP is a server-side language. This means that the code that is written gets sent to the server, and it returns some code depending on what it was asked to do.	HTML is a client-side language. Basically it has to do with the user interface, with which the user interacts. This interface, is most often, the browser on the user's machine.
ASP can use any scripting language, so as to embed programming and a server side directives into an HTML web page.	HTML allows web browsers to interpret display content written between tags. It allows images and objects to be embedded in the webpage.
ASP is used to design user-interactive	HTML is basically used to

or dynamic web pages.

create static web pages.

ASP is case sensitive.

HTML is not case sensitive.

ASP or ASP.NET pages can connect to the database so as to derive its content.

HTML cannot connect to a database.

When the browser requests an ASP file, it passes the request to the ASP engine. The ASP engine reads the file, line by line, and then executes the script line by line. Finally, the ASP file returns to the browser as plain HTML.

When a browser requests an HTML file, the server returns the file.

**A sample ASP coding:**

```
<HTML>
<HEAD>
<TITLE>Sample ASP Clock</TITLE>
</HEAD>
<BODY BGCOLOR="BLACK">
<FONT COLOR="GREEN">
<SCRIPT LANGUAGE="VBScript" RUNAT="SERVER">
RESPONSE.WRITE NOW()
</SCRIPT>
</BODY>
</HTML>
```

**Declaring an ASP script within a webpage**

An ASP script begins with `<%` and ends with `%>`. It can be placed anywhere within a webpage.

**Syntax:**

`<% script contents go here %>`

**Example:**

```
<html>
<head>
<title>ASP script</title>
</head>
<body>
<% Response.Write "ASP is cool!"; %>
</body>
</html>
```

**Output:**

ASP is cool

**Printing text on a webpage**

Printing text on a webpage with ASP is simple. To do so use the Response.Write() statement.

**Syntax:**

Response.Write("textToPrint").

**Example:**

```
<% Response.Write("ASP stands for Active Server Pages") %>
```

**Output:**

ASP stands for Active Server Pages

Alternatively, you could just use an equal sign instead of the Response.Write() statement like this:

```
<%= "ASP stands for Active Server Pages" %>
```

**Output:**

ASP stands for Active Server Pages

**How ASP work:**

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

For example, a client types in a URL into your browser. The browser requests the ASP page from the web server. The server proceeds the file with “.asp” extension to ASP Engine in which Objects



or ActiveX Components can be used to extend the web server with application-specific functionality. In addition, ASP will use ADO to connect to a database (SQL, Access, Oracle, etc.) to pull out the relevant data, such as the current weather in a specific area.

Thus, a different page is generated according to the area specified and time that the page is accessed. Then, the server generates HTML tags before sending it back to the client. Therefore, when you view the source of an ASP file, you will not see any difference from a standard HTML file.

A page created using ASP typically contains a mixture of HTML, scripts, and other components written in any programming language. When a client requests an ASP file, the scripts in the file are processed on the server.

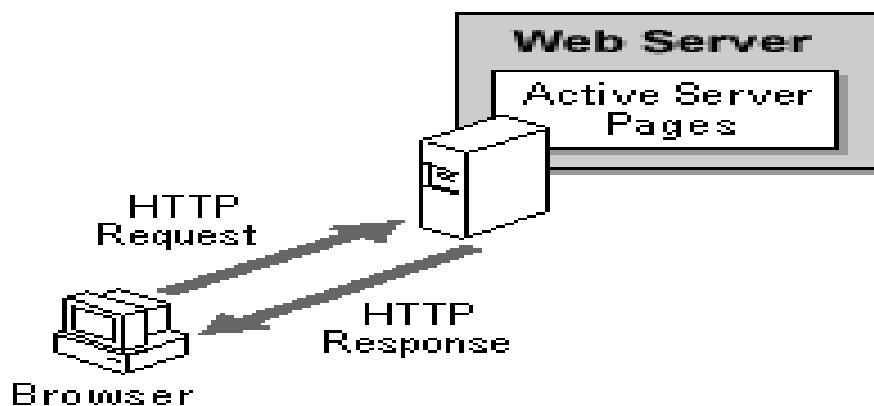
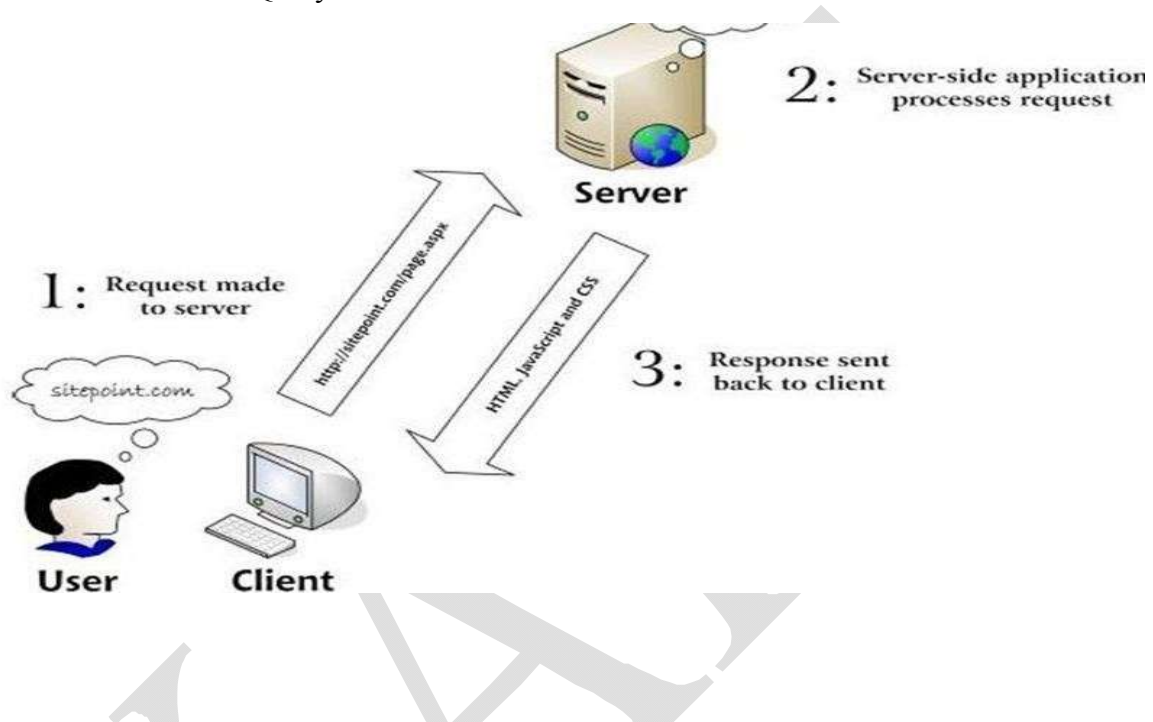
The scripts can reference components running on either the local server or any other accessible server, and can perform actions such as accessing a database, sending e-mail, or processing information in another fashion. The result is then returned by the server to the client as a standard HTML file and displayed in the usual way.

For example, when requested, the following ASP file will return the current time and browser type to the requesting client:

```
<HTML>
<HEAD>
<TITLE>Sample Web Page</TITLE>
</HEAD>
<BODY>
The time right now is <%= now %>
Your browser type is
<%= Request.ServerVariables("http_user_agent")%>
</BODY>
</HTML>
```

### Two Request methods:

- GET – Gets (retrieves) information from the server – Retrieve HTML document or image
- POST – Posts (sends) data to the server – Send info from HTML form »Client -entered data » Info to search Internet » Query for a database » Authentication info.



**Five objects of ASP:**



ASP includes five build-in objects:

- Request – to get information from the user that is passed along with an HTTP request
- Response – to output data to the requesting client
- Server – to control the Internet Information Server
- Session – to store variables associated with a given user session
- Application – to store information that remains active for the lifetime of an application, such as a page counter.

**Request Object**

It allows data to be read that was sent by the client browser: Form, Querystring, and HTTP Cookie. It also provides information on the server, the client browser, and retrieve HTTP Cookie stored on the visitor's machine. Can retrieve data from a form using both methods HTTP:

It allows access to both the header and body of the HTTP request.

**Syntax:**

Request.QueryString ("parameter")

- Form: used today with method POST to provide access to the pair - fieldname = value.

Syntax: Request.Form ("fieldname")

- Query String: used in the past with method GET to pass parameters.
- Server Variables: displays the content of the server environment variables, including REQUEST\_METHOD which allows the testing of a method (GET or POST).

Syntax: Request.ServerVariables ("environment-varname")

- Cookies: allows cookie processing. Syntax: Request.Cookies ( )
- ClientCertificate: used in secure transactions. Syntax: Request.ClientCertificate ( )

**Example:**

Request.Form reads data sent by POST.

Request.QueryString reads data sent by GET.

<%

Response.Write "Welcome" "!"

%>



The Request object — This is used to receive data from the client's browser and computer, including cookies, files, and HTML forms. It provides a way for the server to store and retrieve the received data.

### Response Object:

It allows control over the HTTP response to the client. Properties need to be inserted after `<%@ LANGUAGE="VBSCRIPT"%>` directive, but before the first HTML tag.

- Buffer: controls if the response is sent as the page and script are created, or if are kept in a buffer and sent when processing is completed.

**Syntax:** Response.Buffer = True (False is the default)

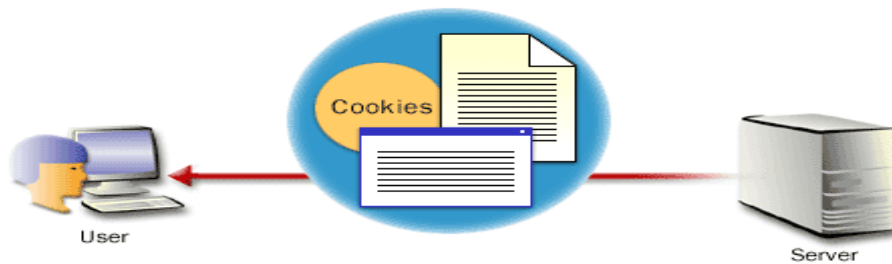
- CacheControl: allows caching by a Proxy server, but the Proxy server needs to be setup to cache pages.

Syntax: Response.CacheControl = "setting" , where setting is Public or Private(default)

- ContentType: allows setting the page content type. The default id text/html.

### Example:

```
<% Response.ContentType = "text/HTML" %>
<% Response.ContentType = "image/GIF" %>
<% Response.ContentType = "image/JPEG" %>
<% Response.ContentType = "text/plain" %>
<% Response.ContentType = "image/JPEG" %>
```



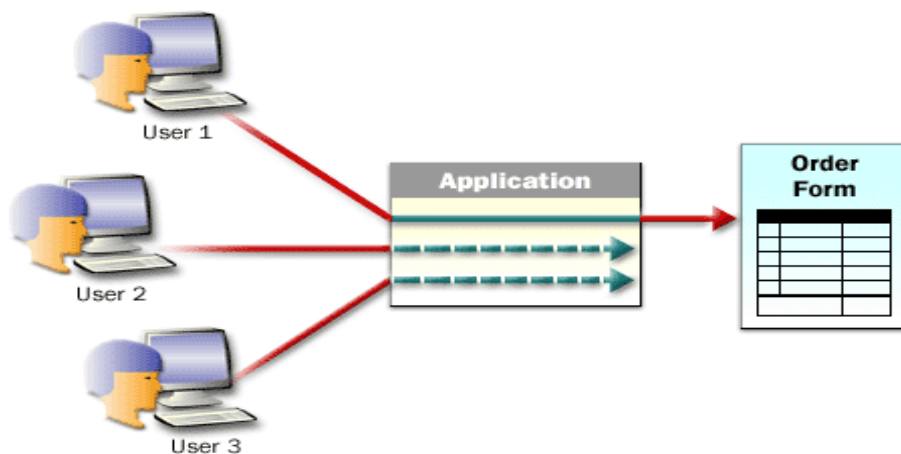
The Response object —This is used to send data or HTML code from directories or memory locations on the server back to the client browser and computer, including cookies and Web pages containing the results of ASP scripts.

### Application Object

The running Web server is an application. Using the Application object, we can control features related to starting and stopping the application, as well as store information that should be accessed by the application as a whole. It stores global variables.

#### Example:

```
<%  
Application("Ali") = "My ASP Application"  
Response.Write "Welcome to " & Server.Encode(Application("Ali")) & "!"  
%>
```



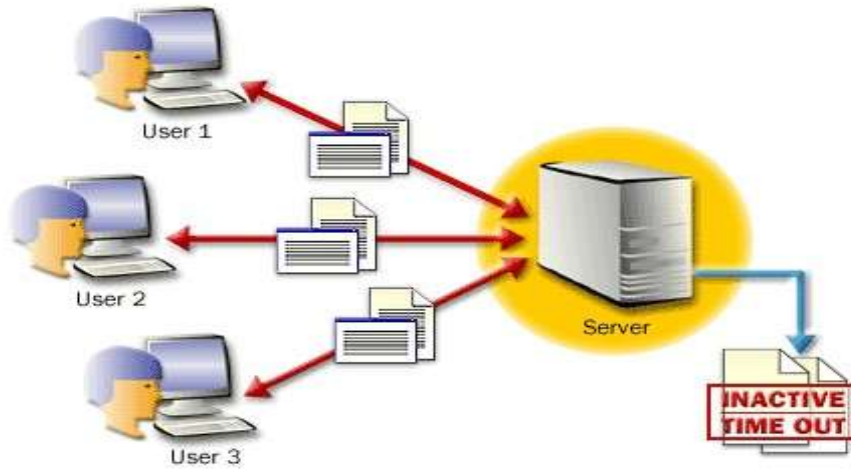
The Application object — This is used to allow more than one user at a time to run a Web application, which can be a separate program linked to the Web pages or the Web pages and ASP scripts themselves. Without this, one user would be setting and resetting counters and calculations that were in use by another user ... an important consideration in an online store Web site.

### Server Object:

The server object, this is used to contain and send information about the web server itself, perform cleanup of user information no longer needed, and create and terminate input/output connection to user files and other applications like database records.

### Example:

```
Dim oAdoCon, oAdoRec, oAdoStm, oCdoCon, oCdoMsg, oSciDic, oSciFsm, oMswAdr
Set oAdoCon = Server.CreateObject("ADODB.Connection")
Set oAdoRec = Server.CreateObject("ADODB.Recordset")
Set oAdoStm = Server.CreateObject("ADODB.Stream")
Set oCdoCon = Server.CreateObject("CDO.Configuration")
Set oCdoMsg = Server.CreateObject("CDO.Message")
```



The Server object —This is used to contain and send information about the Web server itself, perform cleanup of user information no longer needed, and create and terminate input/output connections to user files and other applications (like database records) that cannot or should not be shared among multiple users.

### Session object:

This object is used to store information with a scope to that user session. The information stored is maintained even when the user moves through various pages in the web application.

The session object has two properties.

- SessionID - Created by the web application and sent as a cookie to the client.
- Timeout - To set a Session timeout period.

### Example:

```
<%
```

```
If Len(Request.QueryString("name")) > 0 Then
```

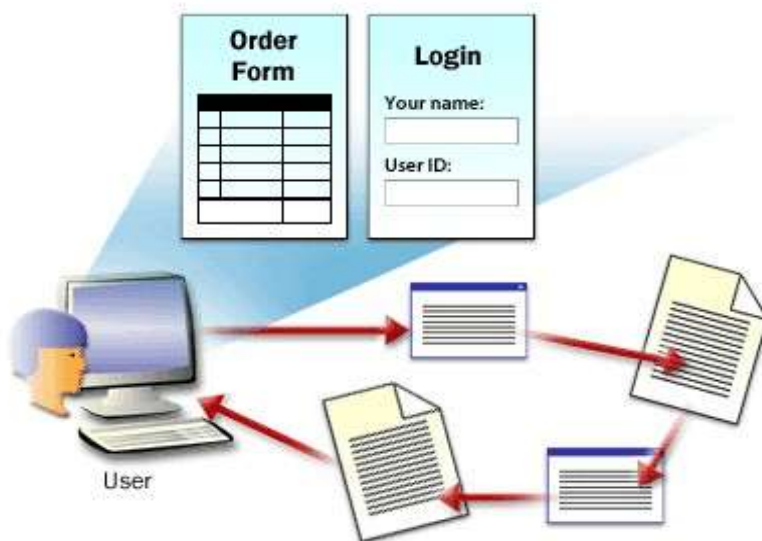
```
Session("name") = Request.QueryString("name")
```

```
End If
```

```
Response.Write "Welcome " & Server.HtmlEncode(Session("name")) & "!"
```

```
%>
```





The Session object —This is used to store data and create a "work area" for each user who visits and navigates around the Web site. It provides the continuity ("state") that is lost in static HTML pages when a user moves from one HTML page to another and can initiate special routines or display special pages when a user first arrives at the site and exits from the site.

### **Client Side Scripting Vs Server Side Scripting:**

#### **Client-side Environment:**

The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.

The scripting language needs to be enabled on the client computer. Sometimes if a user is conscious of security risks they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when the script is attempting to run.

The scripts can be written in two forms, at the server end (back end) or at the client end (server end). The main difference between server-side scripting and client-side scripting is that the server side scripting involves server for its processing. On the other hand, client-side scripting requires browsers to run the scripts on the client machine but does not interact with the server while processing the client-side scripts.

A script is generally a series of program or instruction, which has to be executed on other program or application. As we know that the web works in a client-server environment. The client-

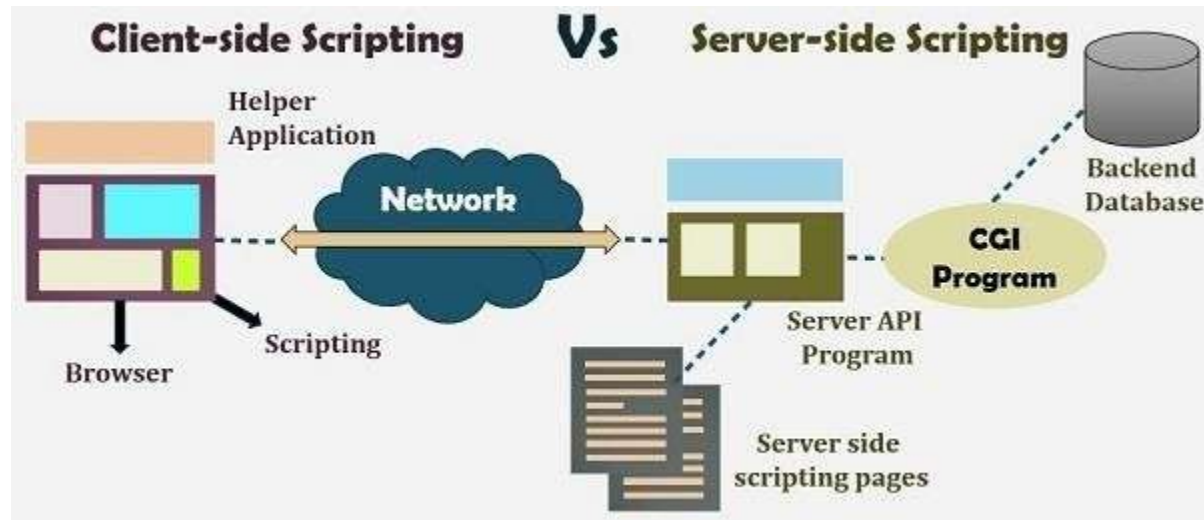
side script executes the code to the client side which is visible to the users while a server-side script is executed in the server end which users cannot see.

### **Server-side Environment**

The server-side environment that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server.

This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores. Server-side scripting is a technique of programming for producing the code which can run software on the server side, in simple words any scripting or programming that can run on the web server is known as server-side scripting. The operations like customization of a website, dynamic change in the website content, response generation to the user's queries, accessing the database, and so on are performed at the server end.

The server-side scripting constructs a communication link between a server and a client (user). Earlier the server side scripting is implemented by the CGI (Common Gateway Interface) scripts. The CGI was devised to execute the scripts from programming languages such as C++ or Perl on the websites.



Basic

Works in the back end which could not be visible at the client end.

Works at the front end and script are visible among the users.

Languages involved

PHP, ASP.net, Ruby on Rails, ColdFusion, Python, etcetera.

HTML, CSS, JavaScript, etc.

**Server Side ActiveX component:**

An ActiveX control is a component program object that can be re-used by many application programs within a computer or among computers in a network. The technology for creating ActiveX controls is part of Microsoft's overall ActiveX set of technologies, chief of which is the Component Object Model (COM).

**Types of Server side ActiveX Components:**

**Ad Rotator**

The Ad Rotator streamlines the process of setting up a delivery system for your banner ads. In a separate file, you store information regarding the banner. The component then delivers a randomly selected banner every time the page is loaded.

The Browser component lets you determine what browser a user is using and what features are supported by that browser Collaboration Data Objects (CDO).

Tied in with the IIS SMTP server, CDO lets you send and receive email. With CDO, for example, you can process a form without relying upon a Perl script and CGI.

**Content Linking**

This is a handy object for creating a linear or sequential pathway through your site or a subsection of the site. You maintain a simple text file that lists the files in the proper sequence. Simple next and previous links then can be added to the page, and a table of contents can be easily generated.

**Content Rotator**

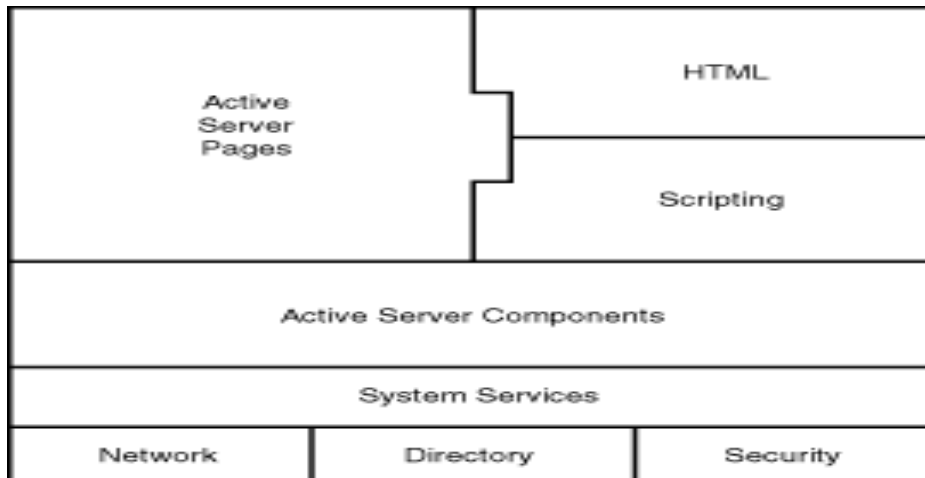
If you have a need for rotating content, this will be a favorite component. It is easy to use and allows you to add dynamic content to any page without using a database. In a separate text file, you store chunks of HTML code that you want alternately dropped into a space on the page. The Content Rotator will display one of the chunks each time the page is reloaded.

**Database Access**

Using this component, you can hook into a database to write contents to the browser screen and to create or update existing database files.

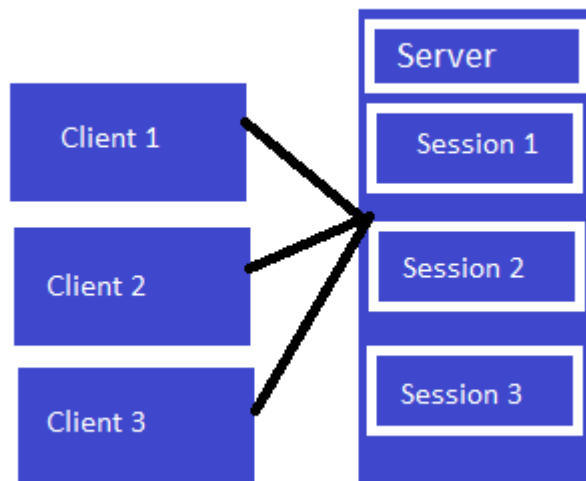
**KARPAGAM ACADEMY OF HIGHER EDUCATION, COIMBATORE****Class: III BCOM (CA)****Course Name: INTERNET AND WEB DESIGN****Course Code: 17CCU601A****Unit -IV****Semester: VI****Year: 2017-2020Batch**

MSWC.BrowserType	ActiveX component for gathering information about the client's browser (e.g., type, version).
MSWC.AdRotator	ActiveX component for rotating advertisements on a Web page.
MSWC.NextLink	ActiveX component for linking Web pages together.
MSWC.ContentRotator	ActiveX component for rotating HTML content on a Web page.
MSWC.PageCounter	ActiveX component for storing the number of times a Web page has been requested.
MSWC.Counters	ActiveX components that provide general-purpose persistent counters.
MSWC.MyInfo	ActiveX component that provides information about a Web site (e.g., owner name, owner address).
Scripting.FileSystemObject	ActiveX component that provides an object library for accessing files on the server or on the server's network.
ActiveX Data Objects (ADO) Data Access Components	ActiveX components that provide an object library for accessing databases.



### **Session Tracking and Cookies:**

Session object stores information about, or change settings for a user session.



### **The Session object**

The Session object stores information about, or change settings for a user session.

The Variables stored in a 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.

**Session Start:**

A session starts when:

- A new user requests an ASP file, and the Global.asa 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

**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, use the Timeout property.

**Example:**

```
<%
```

```
Session.Timeout=5
```

```
%>
```

Store and Retrieve Session Variables

```
<%
```

```
Session("username")="Donald Duck"
```

```
Session("age")=50
```

```
%>
```

**Remove Session Variables**

The Contents collection contains all session variables.

It is possible to remove a session variable with the Remove method.

```
<%
```

```
Session.Contents.RemoveAll()
```

```
%>
```

### Cookies:

The Cookies collection is used to set or get cookie values. If the cookie does not exist, it will be created, and take the value that is specified. The Response. Cookies command must appear before the <html> tag.

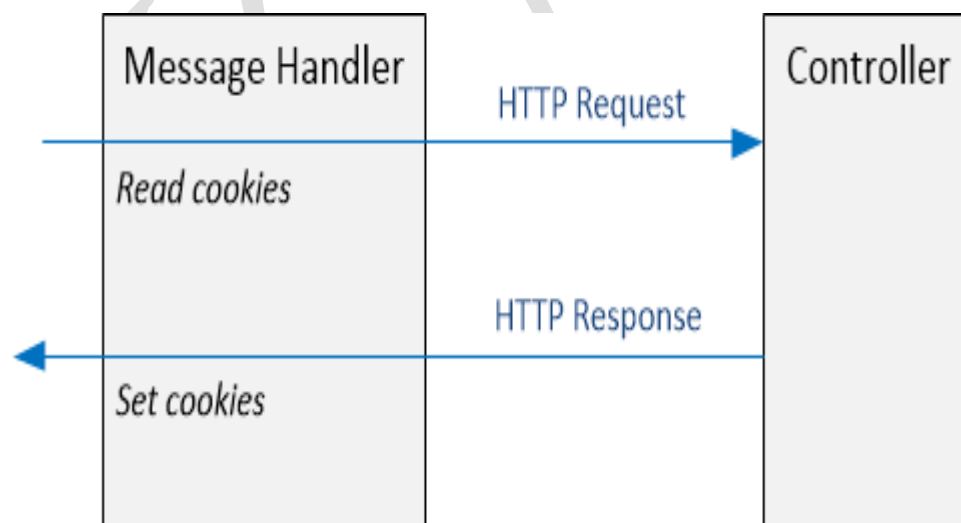
### Example:



### Syntax:

```
Response.Cookies(name)[(key)]. attribute] =value  
variablename=Request.Cookies(name)[(key)]. attribute]
```

### cookies message read and set:





**KARPAGAM ACADEMY OF HIGHER EDUCATION, COIMBATORE****Class: III BCOM (CA)****Course Name: INTERNET AND WEB DESIGN****Course Code: 17CCU601A****Unit -IV****Semester: VI****Year: 2017-2020Batch**

Parameter	Description
Name	Required. The name of the cookie valueRequired for the Response.Cookies command. The value of the cookie.

Attribute	Optional. Specifies information about the cookie. Can be one of the following parameters:
1. Domain	Write only. The cookie is sent only to requests to this domain
2. Expires	Write only. The date when the cookie expires. If no date is specified, the cookie will expire when the session ends
3. HasKeys	Read only. Specifies whether the cookie has keys (This is the only attribute that can be used with the Request.Cookies command)
4. Path	Write only. If set, the cookie is sent only to requests to this path. If not set, the application path is used
Secure	Write only. Indicates if the cookie is secure
key	Optional. Specifies the key to where the value is assigned

- A name-value pair containing the actual data
- An expiry date after which it is no longer valid
- The domain and path of the server it should be sent to

**Syntax :**

<%

Response.Cookies("firstname")="Alex"

%>

**Example :**

!DOCTYPE html>

<html>

<body>

<%

dim x,y

for each x in Request.Cookies

    response.write("<p>")

    if Request.Cookies(x).HasKeys then

        for each y in Request.Cookies(x)

            response.write(x & ":" & y & "=" & Request.Cookies(x)(y))

            response.write("<br>")

        next

    else

        Response.Write(x & "=" & Request.Cookies(x) & "<br>")

    end if

    response.write "</p>"

next

%>

</body>

</html>



### Creating a Cookie

Cookies are created using response. Cookies are sent in the headers of the response so you should always set cookies before sending the body of the response.

### Example Cookie Creation

```
<%  
Response.Cookies("username") = "earthskater"  
Response.Cookies("username"). Expires = Date + 365  
%>
```

In the example above a cookie named user was created with a value of “earthskater”. Then the expiry date was set using the expires property. The cookie was set to expire 1 year from the day’s date. When the cookie expires it is deleted from the user's Web computer.

### Retrieving Cookies

To retrieve cookies we use request.cookies.

### Example Cookie Retrieval

```
<%  
username = Request.Cookies("username")
```

```
response.write("Username: " & username)
%>
```

This code retrieves the value of the cookie named “user name and prints it in the browser. Note that cookies can be retrieved anywhere in the code, unlike creating them which has to be done before anything is written to the body of the response.

### **Cookies with Multiple Values**

Cookies can also contain multiple values, using something called **keys**. Take a look at this example.

```
<%
Response.Cookies("user")("realname") = "John Doe"
Response.Cookies("user")("username") = "johnny55"
Response.Cookies("user")("age") = "55"
%>
```

This is fairly straightforward to understand. Retrieving the cookies is very simple too.

```
<%
realname = Response.Cookies("user")("realname")
username = Response.Cookies("user")("username")
age = Response.Cookies("user")("age")
%>
```

### **Difference between cookies and sessions:**

1. Cookies can store only "string" data type
2. They are stored at Client side
3. Cookie is non-secure since stored in text format at client side
4. Only in few situations we can use cookies because of no security

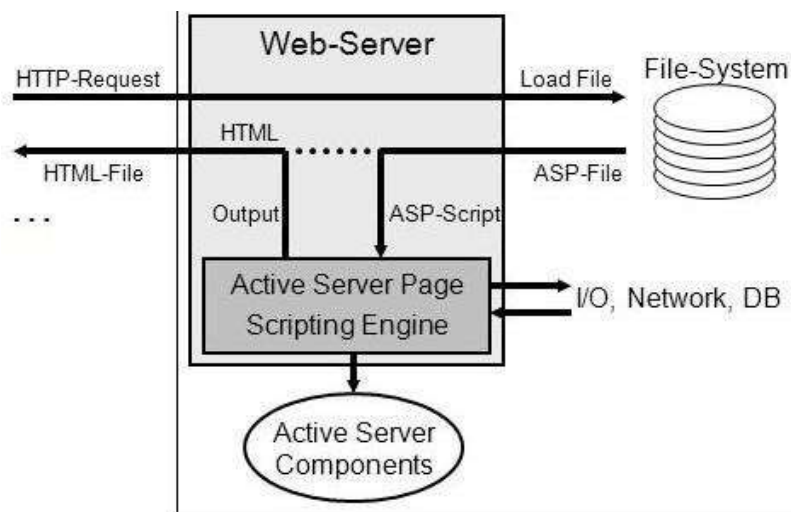
1. Session can store any data type
2. These are stored at Server side
3. Session are secure because it is stored in binary format
4. For all condition /situations we can use sessions

**Cookies Collection:**

Property	Description	Access
Domain	Tells the browser only to send the cookie to the specified domain.	Write only
Expires	Tells the browser to delete the cookie at the specified date. If no date is specified, the cookie expires when the browser is closed.	Write only
HasKeys	Determines whether the cookie has any <i>keys</i> .	Read only
Path	Tells the browser only to send the cookie to the specified path.	Write only
Secure	If set to true, then the cookie will only be sent to the server when using a secure connection. This defaults to false.	Write only

**File system objects:**

The File System Object is used to access the file system on a server. The File System Object 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.



**Syntax**

FileSystemObject.CreateTextFile(filename[,overwrite[,unicode]])

**Scripting.FileSystemObject**

- The CreateObject function returns the FileSystemObject (fs)
- The CreateTextFile method creates the file as a TextStream object (a).
- The WriteLine method writes a line of text to the created text file.
- The Close method flushes the buffer and closes the file.

**Example:**

```
<%  
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  
%>
```

Property	Description
AtEndOfLine	Returns true if the file pointer is positioned immediately before the end-of-line marker in a TextStream file, and false if not.
AtEndOfStream	Returns true if the file pointer is at the end of a TextStream file, and false if not.
Column	Returns the column number of the current character position in an input stream.

**KARPAGAM ACADEMY OF HIGHER EDUCATION, COIMBATORE****Class: III BCOM (CA)****Course Name: INTERNET AND WEB DESIGN****Course Code: 17CCU601A****Unit -IV****Semester: VI****Year: 2017-2020Batch**

Property	Description
Line	Returns the current line number in a TextStream file.
Method	Description
Close	Closes an open TextStream file.
Read	Reads a specified number of characters from a TextStream file and returns the result.
ReadAll	Reads an entire TextStream file and returns the result.
ReadLine	Reads one line from a TextStream file and returns the result.
Skip	Skips a specified number of characters when reading a TextStream file.
SkipLine	Skips the next line when reading a TextStream file.
Write	Writes a specified text to a TextStream file.
WriteBlankLines	Writes a specified number of new-line characters to a TextStream file.
WriteLine	Writes a specified text and a new-line character to a TextStream file.

**The File System Object properties and methods:**

Property	Description
Drives	Returns a collection of all Drive objects on the computer

**KARPAGAM ACADEMY OF HIGHER EDUCATION, COIMBATORE****Class: III BCOM (CA)****Course Name: INTERNET AND WEB DESIGN****Course Code: 17CCU601A****Unit -IV****Semester: VI****Year: 2017-2020Batch****Methods**

<b>Method</b>	<b>Description</b>
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
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

**Example:**

```
<%  
dim fs,f  
set fs=Server.CreateObject("Scripting.FileSystemObject")  
set f=fs.CreateTextFile("c:\test.txt",true)  
f.WriteLine("Hello World!")  
f.Close  
set f=nothing  
set fs=nothing  
%>
```

**POSSIBLE QUESTIONS –UNIT-IV**

**PART A (1 Mark)**

**(Online Examinations)**

**PART B (2 Marks)**

1. What is ASP?
2. What is a Scripting Language?
3. List the five objects in ASP.
4. Write the syntax for Response Object.
5. What are the Methods in Session Object?
6. Compare and contrast the Post & Get Method.
7. Differentiate the Client-side Script and Server-side Script
8. Define a cookie.
9. How to create session tracking in ASP?
10. What are the types of Activex server components?
11. What is Ad Rotator component?
12. Write the syntax for file system object.

**Part-C (6 Marks)**

1. Explain the Active server page with an example.
2. Explain the working principle of ASP.
3. Describe the various types of object in ASP.
4. Discuss the benefits of Server-side scripts.
5. How does the cookie create and retrieve with an example?
6. List the Activex Server Components and explain with an example.
7. Elucidate the session start and end with an example?
8. Expound the different methods and properties of the file system object.

**COURSE NAME: INTERNET AND WEB DESIGN**  
**COURSE CODE :17CCU601A**

**UNIT: IV (ACTIVE SERVER PAGE)**

**SEMESTER: VI**

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	The ASP stands for_____.	Active Server Pages	Activex Server Pages	Active Source Pages	Active Script Pager	Active Server Pages
2	The ASP is a program that runs inside an_____.	ISI	SII	IIS	ISS	IIS
3	The ASP property is used to identify an_____.	Cookies	The Application object	The Server object	ASP Name	Cookies
4	The ASP come with a _____standard component.	AdRotator	Advertise	Rotator Ads	Advertisement	AdRotator
5	The _____command is used to retrieve the information in ASP.	Response.Cookies	Request.SetCookies	Request.SetCookies	Request.Cookies	Request.Cookies
6	The _____represent a hierarchical organization chart.	Object model	Website	Dynamic object model	Web page	Object model
7	A buffer in ASP can be cleared by _____ method.	Clear	Clean	Buffer	Buffer clear	Clear
8	A common object model is a standard communication mechanism between the_____.	Components	Protocols	Scripts	Cookies	Components
9	The text-based files are combination of _____ and_____.	Html tags and ASP Scripts	Html tags and Dhtml	XML and DHTML	XML and ASP	Html tags and ASP Scripts
10	The application logic processed by the server and the component is used_____.	Variables	Constants	Operation	Keywords	Variables
11	The base directory of ASP application is referred to as a _____.	Virtual root	Physical root	Root node	Directory	Virtual root
12	The ASP programming is the processing of _____ script.	Sever-side	Client-side	Server –client-side	Object	Sever-side
13	An ASP script the delimiter is_____.	<% %>	<! !>	<@ @>	<\$ \$>	<% %>
14	A _____is used to track and manage individual user session in an asp application.	Session object	Application object	Request object	Response object	Session object
15	The Session object has _____events.	2	3	4	1	2
16	The fundamental Active Server Page consists of _____objects.	6	5	4	2	6
17	A _____is used to manage all the information in ASP application.	Application object	Session object	Server Object	Response object	Application object
18	A control transaction processing using the MTS is_____.	Context Object	Server object	Request object	Response object	Context Object

19	The_____property is used to name a web control.	Control name	Designation	Id	Name	Id
20	The_____is responsible for sending output from the server to the requesting client.	Request object	Response object	Server object	Client object	Response object
21	The most common response object method is_____.	Write	Retrieve	Request	Scripts	Write
22	The object enrich the value on the client's browser is _____	Cookies	Variables	Scripts	Functions	Cookies
23	In ASP object have properties, methods, events and _____.	Collections	Containers	Counters	Classes	Collections
24	A_____user action will not generate a server-side event.	Mouse move	Text change	Button click	Button	Mouse move
25	The_____is used to retrieve information from a client browser.	Request object	Response object	Object context object	Session object	Request object
26	A_____file extension is used for an ASP.	asn	asx	asp	aspx	asp
27	The page is refreshed in AdRotator component display a_____.	New banner image	Old banner image	Constant	Current banner image	New banner image
28	The_____command is used to create cookies.	Response.Cookies	Request.SetCookies	Request.Cookies	Response.SetCookies	Response.Cookies
29	The_____destroy all the stored object in ASP.	Session Object	Cookies	Script	Activex	Session Object
30	The_____is a method of server object.	MapPath	AddHeader	BinaryWrite	Clear	MapPath
31	An ASP can access a database through the_____object.	Active data	Data Active	Database	Data	Active data
32	The_____script executed by the browser.	Server-side	Client-side script	Web server	Server page	Client-side
33	The_____script executed by the web server.	Web server	Web page	client-side	server-side	server-side
34	The_____is a Asp Scripting Object.	Server	Text Stream	Client	Application	Text Stream
35	The_____method generate a form collection.	Put	Get	Post	Set	Post
36	The_____command is used to retrieve cookies value.	Response.Cookies	Response.SetCookies	Request.Cookies	Request.setCookies	Request.Cookies
37	The_____object is used to access the file on a server.	File system	Application	Session	Cookies	File system
38	The_____method append a name to an existing path in a file system.	Copyfile	Createfolder	Driveexists	Buildpath	Buildpath
39	The_____method return a string that contains the drive name of the specified path.	Getdrivename	Getname	Getextensionname	Getfile	Getdrivename

40	A_____is a conversation between the server and the client.	File system	Session tracking	Web server	Web browser	Session tracking
41	The Session Tracking information is stored in_____.	Application object	Server	Scripts	Cookies	Cookies
42	The EOF stands for_____.	End of file	End of format	Entering of file	Exit of file	End of file
43	The IIS stands for_____.	Internet interchange services	Information internet services	Internet information services	Interchange information services	Internet information services
44	The_____property is a unique identifier that is generated by the server.	Session	Cookies	Activex component	Objects	Session
45	The major object use with ADO is_____.	Connection	Component	Session	Object	Record set
46	The_____component is the collection of object that enable ASP developer to connect a database.	Record set	ADO	DAO	Data	ADO
47	An error collection belongs to_____object.	Help file	String	Source	Connection	Connection
48	The_____method send information by appending it to the URL.	Post	Set	Put	Get	Get
49	A_____method is used to send HTTP output to the browser.	Write	Redirect	Response	Request	Write
50	A_____component display and rotate different advertisement on the web pages.	Ad Rotator	Advertisement	Activex	DAO	Ad Rotator
51	The_____property is used to set the permission for a connection.	Mode	Connection string	Connection Time out	Connection method	Mode
52	A_____text file that manages a frequency of the displayed banner.	Properties	Rotation schedule	Redirection	Rotator	Redirection
53	The_____component is used to create a record in ASP.	ADO	DAO	RDO	DOA	ADO
54	In Ad Rotator Component required files are_____.	Rotation Schedule	Redirection	Rotation Schedule and Redirection	Rotator	Rotation Schedule and Redirection
55	In AdRotator Component the page is refreshed it displays a_____.	New banner image	Old banner image	Constant	Banner image	New banner image
56	A_____method is used to controls the content.	Response. Write	Response. Redirect	Response.ContentType	Response. Cookies	Response.ContentType
57	A_____method allows accessing the content stored on the web server.	Text stream	File stream	Object stream	Image stream	Text stream
58	A_____tag is used to submit information back to the server.	From	Form	Text	Label	Form
59	The GET method has a_____data limit.	3kb	4kb	6kb	2kb	2kb
60	A_____contain all the variables established for a session without using the tag.	Static collection	Content collection	Dynamic collection	Static collection and Content collection	Content collection

**UNIT-V- XML (EXTENSIBLE MARKUP LANGUAGE)**

**SYLLABUS**

Introduction to XML: Introduction – The Syntax of XML – XML Document Structure – Document Type Definitions – Namespaces – XML Schemas – Displaying Raw XML Documents – Displaying XML Documents with CSS – XML Processors.

**Introduction to XML:**

**Introduction**

**XML—EXTENSIBLE MARKUP LANGUAGE**

The eXtensible Markup Language is the language that another version of HTML is based on. Like HTML, XML is also based off of SGML. It is less strict than SGML and stricter than plain HTML. XML provides the extensibility to create various different languages.

XML is a language for writing markup languages. For example, if you are working on genealogy, you might create tags using XML to define the father, mother, daughter, and son in your XML like this: <father> <mother> <daughter> <son>. There are also several standardized languages already created with XML: MathML for defining mathematics, SMIL for working with multimedia, XHTML, and many others.

**XHTML—EXTENDED HYPERTEXT MARKUP LANGUAGE**

XHTML 1.0 is HTML 4.0 redefined to meet the XML standard.

XHTML has been replaced in modern web design with HTML5 and the changes that have come since. You are unlikely to find any newer sites using XHTML, but if you are working on a much older site, you may still encounter XHTML out there in the wild.

There aren't a lot of major differences between HTML and XHTML, but here is what you will notice:

XHTML is written in lower case. While HTML tags can be written in UPPER case, MiXeD case, or lower case, to be correct, XHTML tags must be all lower case. (Note - many web professionals write HTML in all lowercase, even though it is not technical required).

- All XHTML elements must have an end tag. Elements with only one tag, such as `<br />` need a closing slash (/) at the end of the tag:

`<br />`

`<br />`

- All attributes must be quoted in XHTML. Some people remove the quotes around attributes to save space, but they are required for correct XHTML.
- XHTML requires that tags are nested correctly. If you open a bold (`<b>`) element and then an italics (`<i>`) element, you must close the italics element (`</i>`) before you close the bold (`</b>`). (Note that both of these elements have been deprecated because they are visual elements. HTML now uses `<strong>` and `<em>` in place of these two)
- HTML Attributes must have a name and a value. Attributes that are stand-alone in HTML must be declared with values as well, for example, the HR attribute would be written `noshade="noshade"`.

### **What is XML?**

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

### **Example**

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

`<note>`

`<to>Tove</to>`

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

### **XML Tags are Case Sensitive**

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.

### **Opening and closing tags must be written with the same case:**

<Message>This is incorrect</message>

<message>This is correct</message>

"Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.

### **XML Elements Must be Properly Nested**

In HTML, you might see improperly nested elements:

<b><i>This text is bold and italic</b></i>

In XML, all elements must be properly nested within each other:

<b><i>This text is bold and italic</i></b>

In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

### **XML Attribute Values Must be Quoted**



XML elements can have attributes in name/value pairs just like in HTML. In XML, the attribute values must always be quoted.

**INCORRECT:**

```
<note date=12/11/2007>
```

```
<to>Tove</to>
```

```
<from>Jani</from>
```

```
</note>
```

**CORRECT:**

```
<note date="12/11/2007">
```

```
<to>Tove</to>
```

```
<from>Jani</from>
```

```
</note>
```

**XML Elements**

An XML document contains XML Elements.

**What is an XML Element?**

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

```
<price>29.99</price>
```

 An element can contain:

- text
- attributes
- other elements
- or a mix of the above

**Comments in XML**

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

Two dashes in the middle of a comment are not allowed. Not allowed:

<!-- This is a -- comment ->

Strange, but allowed:

<!-- This is a - - comment -->

**White-space is preserved in XML**

XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):

XML:	Hello	Tove

**The Difference between XML and HTML:**

No.	HTML	XML
1)	HTML is used <b>to display data</b> and focuses on how data looks.	XML is a software and hardware independent tool used <b>to transport and store data</b> . It focuses on what data is.
2)		
3)	HTML is <b>not case sensitive</b> .	XML is <b>case sensitive</b> .

5)	HTML has its own predefined tags.	You can define tags according to your need.
		XML makes it mandatory to use a closing tag.
7)	HTML is <b>static</b> because it is used to display data.	XML is <b>dynamic</b> because it is used to transport data.
		L preserve whitespaces.

### **XML Does Not Use Predefined Tags**

The XML language has no predefined tags.

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

HTML works with predefined tags like <p>, <h1>, <table>, etc.

Most XML applications will work as expected even if new data is added (or removed).

Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <data>).

Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.

The way XML is constructed; older version of the application can still work:

<note>

<date>2015-09-01</date>

<hour>08:30</hour>

<to>Raja</to>

<from>Ram</from>

<body>Don't forget me this weekend! </body>

</note>

### **Design goals for XML**

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

### **A Family of Standards and Recommendations:**

XML is not a single standard. The core XML 1.0 syntactic definition is enhanced by the following standards that are either complete or under development:

#### **Recommendations**

- XML 1.0: February 1998 (Revised October 2000)
- XML Namespaces: January 1999
- XSLT: November 1999 (part of XSL)
- XPath: November 1999 (used by XSLT and XPointer)
- XHTML (HTML redefined as XML): January 2000
- XHTML Basic: December 2000 (Base subset)
- Canonical XML: March 2001

## KARPAGAM ACADEMY OF HIGHER EDUCATION, COIMBATORE

**Class: III BCOM (CA)**

**Course Name: INTERNET AND WEB DESIGN**

**Course Code: 17CCU601A**

**Unit -V**

**Semester: VI**

**Year: 2017-2020 Batch**

- XML Schemas Parts 1 and 2: May 2001
- XLink and XBase: June 2001
- XSL: October 2001
- XML Information Set: October 2001
- XML-Signature: February 2002

### **Proposed Recommendations**

- Exclusive XML Canonicalization: May 2002

### **Candidate Recommendations**

- XML Fragment Interchange: February 2001
- XPointer: September 2001
- XInclude: February 2002
- XML Signature Decryption: March 2002
- XML Encryption: March 2002

### **Working Drafts**

- XML Events: October 2001
- XML Protocol: December 2001
- XForms 1.0: January 2002
- XML Key Management: March 2002
- XKMS: March 2002
- XPath 2.0: April 2002
- XQuery: April 2002
- XML 1.1: April 2002

### **Important XML Standards**

This tutorial will also dig deep into the following important XML standards:

- XML AJAX
- XML DOM
- XML XPath
- XML XSLT

- XML XQuery
- XML DTD
- XML Schema
- XML Services

**Following is a partial list of related XML-related areas being worked on at W3C:**

- Associating Style Sheets with XML documents Version 1.0 Recommendation -- W3C [June 29, 1999]

Standardized syntax for using an xml- stylesheet processing instruction to associate an XML document with an XSL or CSS stylesheet.

- Authoring Tool Accessibility Guidelines Version 1.0

Recommendation -- W3C [Feb. 3, 2000] Guidelines for web authoring tool developers to assist in the design of authoring tools that produce accessible Web content and in the creation of an accessible authoring interface.

- CSS1 Recommendation (Revised) -- W3C [Jan. 11, 1999] CSS1 is a simple style sheet mechanism that allows authors and readers to attach style (e.g. fonts, colors and spacing) to HTML documents.

- CSS2 Recommendation -- W3C [May 12, 1998] Level 2 of the Cascading Style Sheet mechanism

- DOM Level 1 Recommendation -- W3C [Oct. 1, 1998] Document Object Model Level 1

- Namespaces in XML Recommendation -- W3C [Jan. 14, 1999] Provide a simple method for qualifying element and attribute names used in XML documents. This is accomplished by associating a particular tag set by associating a prefix with a URI reference.

- PICS Recommendation -- W3C [May 27, 1998] Standard format for making digitally-signed, machine-readable assertions about a particular information resource.

- PICS Rules Recommendation -- W3C [Dec. 29, 1997] ·RDF Model and Syntax Recommendation -- W3C [Feb. 22, 1999] A foundation for processing metadata; it provides

interoperability between applications that exchange machine-understandable information on the Web.

- XHTML 1.0 Recommendation -- W3C [Jan. 26, 2000] A reformulation of HTML 4 as an XML 1.0 application and three DTDs corresponding to the ones defined by HTML 4.

- XML Path Language (XPath) Version 1.0 Recommendation -- W3C [Nov. 16, 1999] Provide a common syntax and semantics for querying and addressing the contents of XML documents that could be used by XSLT (XSL Transformation Language), XLink, and XPointer. XML v. 1.0 DTD Revised XML Recommendation DTD -- W3C [Sept. 10, 1998] A revised version of the XML Recommendation's DTD.

· XML XPointer Requirements W3C [Feb. 24, 1999] · XSL Transformations (XSLT) Specification Version 1.0 Recommendation -- W3C [Nov. 16, 1999] A language used to "transform" (or reconstruct the structure of) the data structures contained within XML documents.

· XInclude Inclusion Proposal (proposed for XML version 2.0) Fragments of content from external resources are included in documents content along with the content residing at the actual URL being accessed (sort of like a server-side include).

### **XML Syntax :**

- XML documents must contain one root element that is the parent of all other elements.
- All XML Elements must have their corresponding Closing Tag.

### **Xml syntax:**

```
<?xml  
  version = "version_number"  
  encoding = "encoding_declaration"  
  standalone = "standalone_status"  
?>
```

**Example:**

```
<?xml version = "1.0"?>
<contact-info>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</contact-info>
```

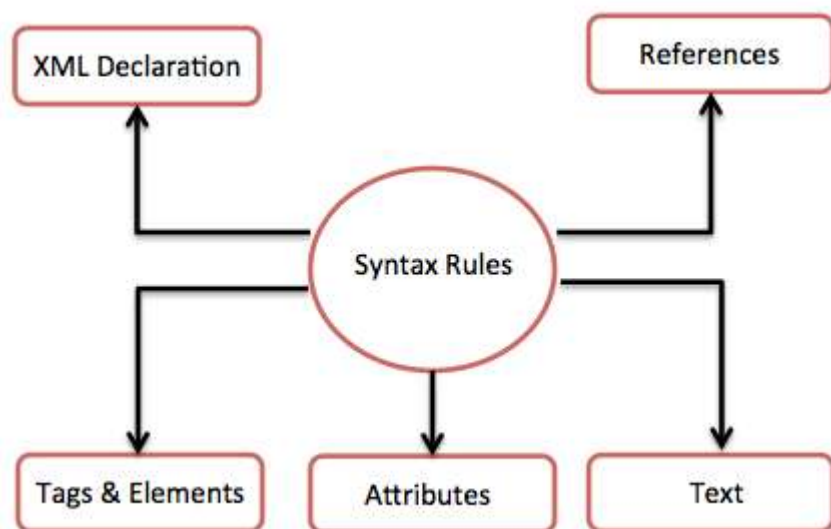
**XML Syntax rules:**

- All XML elements must have a closing tag.
- It is illegal to omit the closing tag when you are creating XML syntax.
- XML tags are case sensitive.
- All XML elements must be properly nested.
- All XML documents must have a root element.
- Attribute values must always be quoted.
- 

**XML Declaration**

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





### **XML Document Structure:**

A well-formed XML document complies with the syntactic rules of XML markup. These rules are strict but basically quite simple.

- documents must be self-describing (they start with an XML declaration)
- a document must contain one or more elements
- documents must contain a single root element
- start and end tags must be used to identify elements (must have closing tags)
- empty elements must be marked as such (with a self-closing />)
- attribute values must be quoted (single or double quotes are fine)
- Element names and attributes names are case sensitive.
- element tags must be correctly nested (must not overlap)
- Element names (attribute names) must start with a letter

XML Document consists of many parts. XML Document has 2 main parts:

1. XML Prolog

2. XML Body.

### **XML document structure**

This section introduces the terms used to describe the parts of an XML document, starting at the top level and working down to the smallest parts.

XML documents are made up of two parts, called the prolog and the body

### **XML prolog**

The prolog contains optional information such as the XML version the document conforms to, information about the character encoding used to encode the contents of the document, and a reference to either a document type definition (DTD) or XML Schema document which describes the grammar and vocabulary of the document. The XML Schema document is the more modern way to describe XML grammar and vocabulary. XML Schemas and DTDs are usually stored in external documents and the prolog can reference both XML Schemas and DTDs.

This simple example illustrates the prolog:

```
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE maillist SYSTEM "maillist.dtd">
```

### **XML body**

The body contains a single top-level element called the root element, which contains all other elements and other markup information.

This simple example illustrates the prolog and body, where addresses is the root element and encloses the entire body, or content, of the XML document:

```
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE maillist SYSTEM "maillist.dtd">  
<addresses>  
  <address>...</address>  
  <address>...</address>  
  <address>...</address>  
</addresses>
```

**Example:**

```
<?xml version="1.0"?>
<contact-info>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</contact-info>
```



The diagram illustrates the structure of the XML code. A bracket on the right side groups the first line, `<?xml version="1.0"?>`, under the label "Document Prolog". Another bracket groups the subsequent lines, `<contact-info>` through `</contact-info>`, under the label "Document Elements".

**Describing Structures in XML**

XML was describing a tree structure of data. And tree structures have one root, child elements, branches, attributes, values. Following are simple XML structures.

`<root>`

`<element>`

`<subelement>...</subelement>`

`</element>`

`</root>`

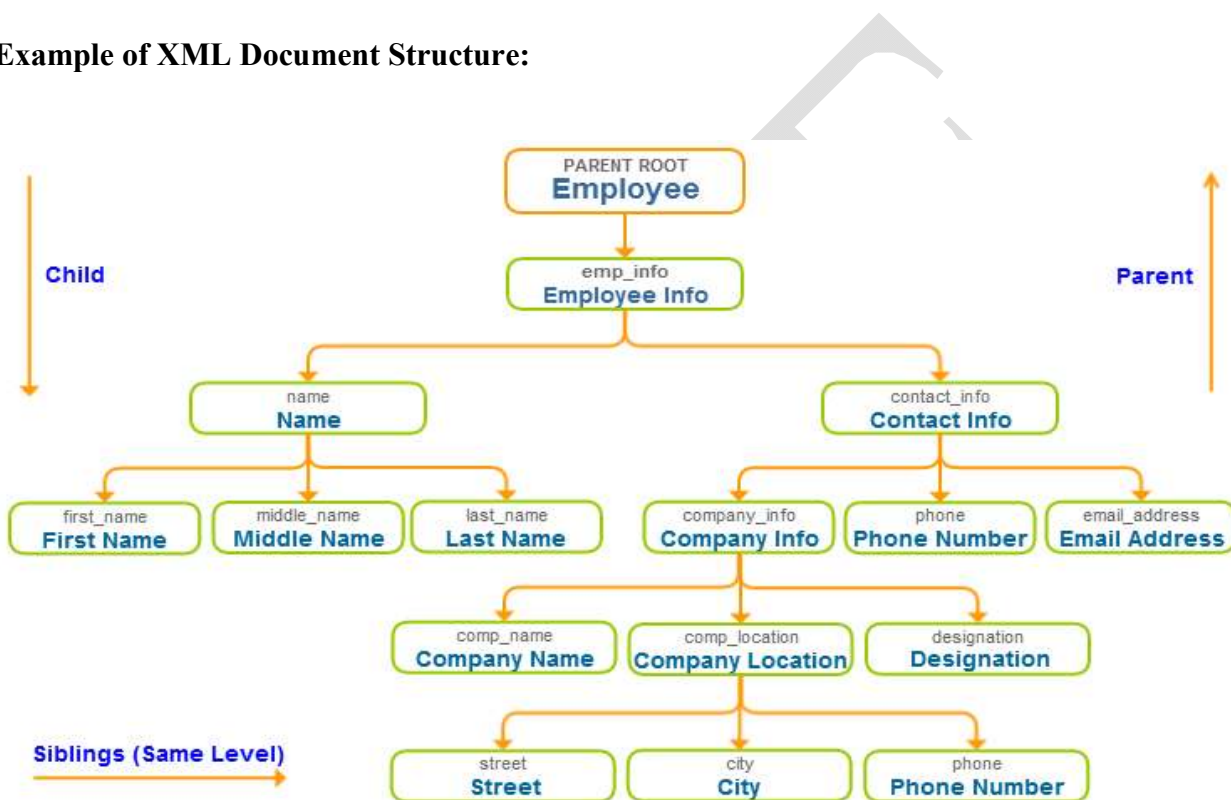
**XML Document Example**

Therefore a complete well formed XML document may look like:

```
<?xml version="1.0"?>
<students>
  <student>
    <name>ram</name>
    <rollno>1</rollno>
    <program>Mca</program>
  </student>
  <student>
    <name>shyam</name>
```

```
<rollno>2</rollno>
<program>Mca</program>
</student>
</students>
```

**Example of XML Document Structure:**



**Example Xml structure:**

Raw XML Content	Type of Node	Value	Number of Children
<?xml version="1.0" ?>	Processing Instruction		
<Order>	Element		3

**KARPAGAM ACADEMY OF HIGHER EDUCATION, COIMBATORE****Class: III BCOM (CA)****Course Name: INTERNET AND WEB DESIGN****Course Code: 17CCU601A****Unit -V****Semester: VI****Year: 2017-2020 Batch**

Raw XML Content	Type of Node	Value	Number of Children
<!-- This is an order >	Comment	This is an order	
<Name>	Element		1
Scott Bradley	Text	Scott Bradley	
</Name>			
<OrderItems>	Element		1
<Item id="1">	Element with Attribute		1
Lord of the Rings	Text	Lord of the Rings	
</Item>			
</OrderItems>			
</Order>			

**Document Type Definitions:**

DTD stands for **Document Type Definition**. It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes.

**DTD - Syntax**

An XML DTD can be either specified inside the document, or it can be kept in a separate document and then the document can be linked to the DTD document to use it.

**Syntax**

Basic syntax of a DTD is as follows –

```
<!DOCTYPE element DTD identifier
```

```
[  
  declaration1  
  declaration2  
  .....  
>]
```

### **Purpose of DTD**

Its main purpose is to define the structure of an XML document. It contains a list of legal elements and defines the structure with the help of them.

### **Checking Validation**

Before proceeding with XML DTD, you must check the validation. An XML document is called "well-formed" if it contains the correct syntax.

A well-formed and valid XML document is one which has been validated against DTD.

### **Example:**

#### **employee.xml**

```
<?xml version="1.0"?>  
<!DOCTYPE employee SYSTEM "employee.dtd">  
<employee>  
  <firstname>vimal</firstname>  
  <lastname>jaiswal</lastname>  
  <email>vimal@javatpoint.com</email>  
</employee>
```

In the above example, the DOCTYPE declaration refers to an external DTD file. The content of the file is shown in below paragraph.

#### **description of dtd**

**<!DOCTYPE employee :** It defines that the root element of the document is employee.

**<!ELEMENT employee:** It defines that the employee element contains 3 elements "firstname, lastname and email".

**<!ELEMENT firstname:** It defines that the firstname element is #PCDATA typed. (parse-able data type).

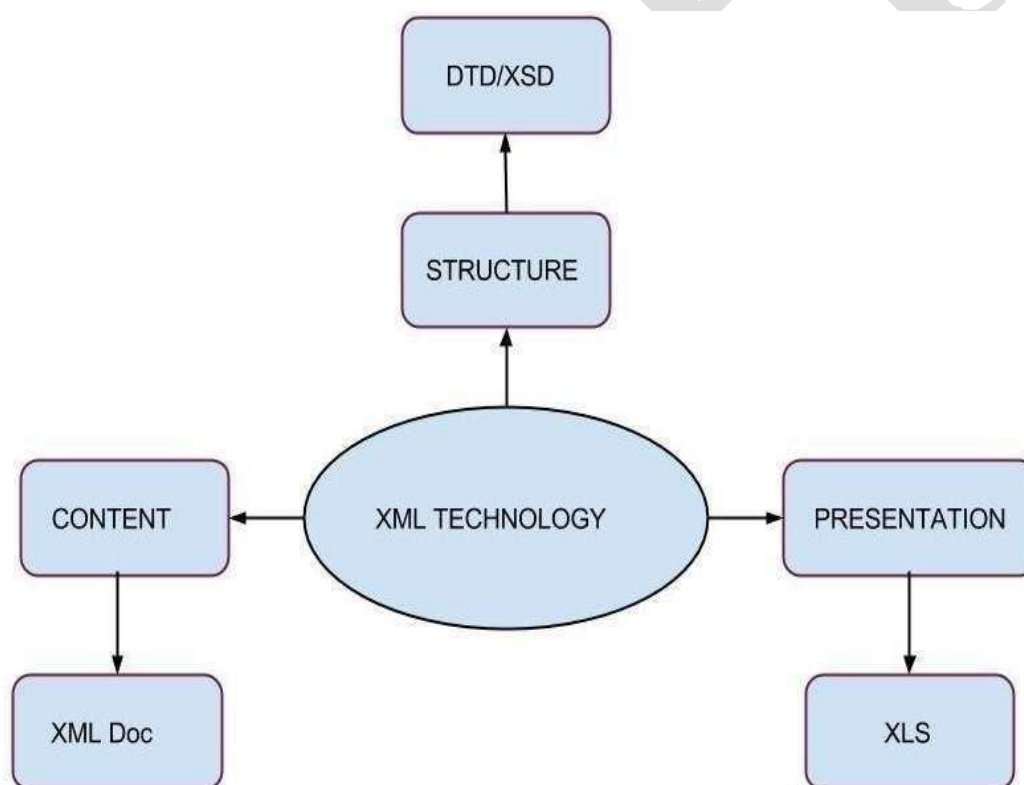
**<!ELEMENT lastname:** It defines that the lastname element is #PCDATA typed. (parse-able data type).

**<!ELEMENT email:** It defines that the email element is #PCDATA typed. (parse-able data type).

### XML DTD with entity declaration

A doctype declaration can also define special strings that can be used in the XML file. An entity has three parts:

1. An ampersand (&)
2. An entityname
3. A semicolon (;)



types

DTD can be classified on its declaration basis in the XML document, such as –

- Internal DTD
- External DTD

When a DTD is declared within the file it is called **Internal DTD** and if it is declared in a separate file it is called **External DTD**.

Features

Following are some important points that a DTD describes –

- The elements that can appear in an XML document.
- The order in which they can appear.
- Optional and mandatory elements.
- Element attributes and whether they are optional or mandatory.
- Whether attributes can have default values.

Advantages of using DTD

- **Documentation** – You can define your own format for the XML files. Looking at this document a user/developer can understand the structure of the data.
- **Validation** – It gives a way to check the validity of XML files by checking whether the elements appear in the right order, mandatory elements and attributes are in place, the elements and attributes have not been inserted in an incorrect way, and so on.

Disadvantages of using DTD

- It does not support the namespaces. Namespace is a mechanism by which element and attribute names can be assigned to groups. However, in a DTD namespaces have to be defined within the DTD, which violates the purpose of using namespaces.
- It supports only the text string data type.
- It is not object oriented. Hence, the concept of inheritance cannot be applied on the DTDs.
- Limited possibilities to express the cardinality for element.



## Namespaces

- A **Namespace** is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to a group. The Namespace is identified by URI(Uniform Resource Identifiers).

- NamespaceDeclaration

- A Namespace is declared using reserved attributes. Such an attribute name must either be **xmlns** or begin with **xmlns:** shown as below –

### Syntax

- The Namespace starts with the keyword **xmlns**.
- The word **name** is the Namespace prefix.
- The **URL** is the Namespace identifier.

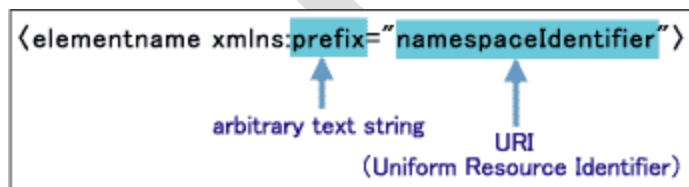
```
<element xmlns:name = "URL">
```

### Example

```
<?xml version = "1.0" encoding = "UTF-8"?>
<cont:contact xmlns:cont = "www.tutorialspoint.com/profile">
  <cont:name>Tanmay Patil</cont:name>
  <cont:company>TutorialsPoint</cont:company>
  <cont:phone>(011) 123-4567</cont:phone>
</cont:contact>
```

## Namespace Declaration

Write a namespace declaration according to the following description method, describing the element start tag:



## XML Naming Rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore

## KARPAGAM ACADEMY OF HIGHER EDUCATION, COIMBATORE

Class: III BCOM (CA)

Course Name: INTERNET AND WEB DESIGN

Course Code: 17CCU601A

Unit -V

Semester: VI

Year: 2017-2020 Batch

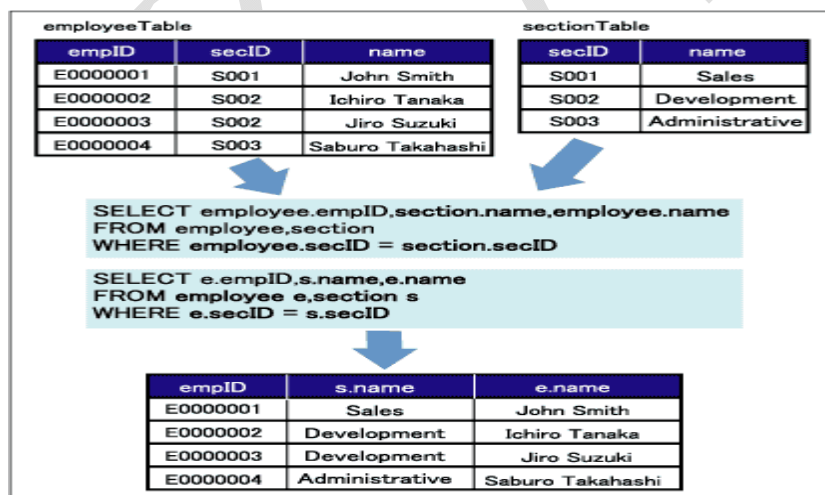
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

### Naming Styles

There are no naming styles defined for XML elements. But here are some commonly used:

Style	Example	Description
Upper case	<FIRSTNAME>	All letters upper case
Pascal case	<FirstName>	Uppercase first letter in each word

### Example namespacing:



### XML Namespaces -Attribute

When using prefixes in XML, a **namespace** for the prefix must be defined.

The namespace can be defined by an **xmlns** attribute in the start tag of an element.

The namespace declaration has the following syntax.

<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">

<h:tr>

<h:td>Apples</h:td>

<h:td>Bananas</h:td>

</h:tr>

</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">

<f:name>African Coffee Table</f:name>

<f:width>80</f:width>

<f:length>120</f:length>

</f:table>

</root>

### XML Schemas

#### What is XML SCHEMA?

XMLschema is a language which is used for expressing constraint about XML documents.

#### Schema syntax:

The <schema> element is the root element of every XML Schema:

<?xml version="1.0"?>

<xs:schema>

...  
...  
</xs:schema>

### XML Schema Example

```
<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <xs:element name="Author">  
    <xs:complexType>  
      <xs:sequence>  
        <xs:element name="FirstName" type="xs:string"/>  
        <xs:element name="LastName" type="xs:string"/>  
      </xs:sequence>  
    </xs:complexType>  
  </xs:element>  
</xs:schema>
```

### Description of XML Schema:

**<xs:element name="employee">** : It defines the element name employee.

**<xs:complexType>** : It defines that the element 'employee' is complex type.

**<xs:sequence>** : It defines that the complex type is a sequence of elements.

**<xs:element name="firstname" type="xs:string"/>** : It defines that the element 'firstname' is of string/text type.

**<xs:element name="lastname" type="xs:string"/>** : It defines that the element 'lastname' is of string/text type.

**<xs:element name="email" type="xs:string"/>** : It defines that the element 'email' is of string/text type.

### **Checking Validation**

An XML document is called well-formed if it contains the correct syntax. A well-formed and valid XML document is one which has been validated against Schema.

There are two types of data types in XML schema.

1. Simple Type
2. Complex type

### **Simple Type**

The simple Type allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.

### **Complex Type**

The complex Type allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.

### **XML Schemas Support Data Types**

One of the greatest strength of XML Schemas is the support for data types.

- It is easier to describe allowable document content
- It is easier to validate the correctness of data
- It is easier to define data facets (restrictions on data)
- It is easier to define data patterns (data formats)
- It is easier to convert data between different data types

### **Schema components:**

The main components of a schema are:

- **Element declarations**, which define properties of elements. These include the elementname and target namespace. An important property is the type of the element, which constrains what attributes and children the element can have. In XSD the type of the element may be conditional on the values of its attributes. An element may belong to a substitution group; if element E is in the substitution group of element H, then wherever the schema permits H to appear, E may appear in its place. Elements may have integrity constraints: uniqueness constraints determining that particular values must be unique within the subtree rooted at an element, and referential constraints determining that values must match the identifier of some other element. Element declarations may be global or local, allowing the same name to be used for unrelated elements in different parts of an instance document.
- **Attribute declarations**, which define properties of attributes. Again the properties include the attribute name and target namespace. The attribute type constrains the values that the attribute may take. An attribute declaration may also include a default value or a fixed value (which is then the only value the attribute may take.)
- **Simple and complex types**. These are described in the following section.
- **Model group** and **attribute group** definitions. These are essentially macros: named groups of elements and attributes that can be reused in many different type definitions.
- An **attribute use** represents the relationship of a complex type and an attributedeclaration, and indicates whether the attribute is mandatory or optional when it is used in that type.
- An **element particle** similarly represents the relationship of a complex type and an element declaration, and indicates the minimum and maximum number of times the element may appear in the content. As well as element particles, content models can include **model group** particles, which act like non-terminals in a grammar: they define the choice and repetition units within thesequence of permitted elements. In addition, **wildcard** particles are allowed, which permit a set of different elements (perhaps any element provided it is in a certain namespace).

Other more specialized components include annotations, assertions, notations, and the**schema component** which contains information about the schema as a whole.

### **Displaying Raw XML Documents:**

Raw XML files can be viewed in all major browsers.

XML is now mature enough that recent versions of the more popular web browsers support it natively. At the time of writing, the most recent versions of these browsers include:

- Microsoft Internet Explorer 6 (<http://www.microsoft.com/windows/ie/>)
- Mozilla 1.7 and Mozilla Firefox 0.9 (<http://www.mozilla.org>)
- Netscape 7.1 (<http://channels.netscape.com/ns/browsers/download.jsp>)
- Opera 7.51 (<http://www.opera.com>)
- Apple's Safari 1.2 (<http://www.apple.com/safari/>)

### **Raw XML Files**

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

```
- <note>
```

```
    <to>Tove</to>
```

```
    <from>Jani</from>
```

```
    <heading>Reminder</heading>
```

```
    <body>Don't forget me this weekend!</body>
```

```
</note>
```

### **. Displaying XML file using CSS:**

CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document.

#### **• Basic steps in defining a CSS style sheet for XML :**

For defining the style rules for the XML document, the following things should be done :-

1. Define the style rules for the text elements such as font-size, color, font-weight, etc.
2. Define each element either as a block, inline or list element, using the display property of CSS.
3. Identify the titles and bold them.

- **Linking XML with CSS :**

In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:

```
<?xml-stylesheet type="text/css" href="name_of_css_file.css"?>
```

**Example:**

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="Rule.css"?>
<books>
  <heading>Welcome To GeeksforGeeks </heading>
  <book>
    <title>Title -: Web Programming</title>
    <author>Author -: Chrisbates</author>
    <publisher>Publisher -: Wiley</publisher>
    <edition>Edition -: 3</edition>
    <price> Price -: 300</price>
  </book>
  <book>
    <title>Title -: Internet world-wide-web</title>
    <author>Author -: Ditel</author>
    <publisher>Publisher -: Pearson</publisher>
    <edition>Edition -: 3</edition>
    <price>Price -: 400</price>
  </book>
  <book>
    <title>Title -: Computer Networks</title>
    <author>Author -: Foruouzan</author>
    <publisher>Publisher -: Mc Graw Hill</publisher>
    <edition>Edition -: 5</edition>
```



```
<price>Price -: 700</price>
</book>
<book>
  <title>Title -: DBMS Concepts</title>
  <author>Author -: Navath</author>
  <publisher>Publisher -: Oxford</publisher>
  <edition>Edition -: 5</edition>
  <price>Price -: 600</price>
</book>
<book>
  <title>Title -: Linux Programming</title>
  <author>Author -: Subhitab Das</author>
  <publisher>Publisher -: Oxford</publisher>
  <edition>Edition -: 8</edition>
  <price>Price -: 300</price>
</book>
</books>
```

#### **Advantages of displaying XML using CSS:**

- CSS is used in XML or HTML to decorate the pages.
- CSS is used for interactive interface, so it is understandable by user.
- CSS enable multiple pages to share formatting, and reduce complexity and repetition in the structural content. So page loader is faster.

#### **Disadvantages of displaying XML using CSS:**

- Using CSS, no transformation can be applied to the XML documents.
- CSS uses different dimensions with different browsers. So the programmer has to run the code in different browser and test its compatibility to post it live.
- CSS have different level of versions, so it is confusing for the browser and user.

**Xml processor:**

When a software program reads an XML document and takes actions accordingly, this is called processing the XML. Any program that can read and process XML documents is known as an XML processor. An XML processor reads the XML file and turns it into in-memory structures that the rest of the program can access.

The most fundamental XML processor reads an XML document and converts it into an internal representation for other programs or subroutines to use. This is called a parser, and it is an important component of every XML processing program.

**Types**

XML processors are classified as validating or non-validating types, depending on whether or not they check XML documents for validity. A processor that discovers a validity error must be able to report it, but may continue with normal processing.

A few validating parsers are – xml4c (IBM, in C++), xml4j (IBM, in Java), MSXML (Microsoft, in Java), TclXML (TCL), xmlproc (Python), XML::Parser (Perl), Java Project X (Sun, in Java).

A few non-validating parsers are – OpenXML (Java), Lark (Java), xp (Java), AElfred (Java), expat (C), XParse (JavaScript), xmllib (Python).

**Components of XML processor**

**Parser:-** Every XML processor has a parser. An XML parser converts an XML document into an XML DOM object - which can then be manipulated with a JavaScript. The parser's job is to translate XML markup and data into a stream of bite-sized nuggets, called tokens, to be used for processing.

A token may be an element start tag, a string of character content, the beginning delimiter of a processing instruction, or some other piece of markup that indicates a change between regions of the document.

**Stage 1:** In this stage, the application parses and validates the source document; recognizes and searches for relevant information based on its location or its tagging in the source document; extracts the relevant information when it is located; and, optionally, maps and binds the retrieved information to business objects.

**Stage 2:** Business logic handling. This is the stage in which the actual processing of the input information takes place. It might result in the generation of output information.

**Stage 3:** XML output processing. In this stage, the application constructs a model of the document to be generated with the Document Object Model (DOM). It then either applies XSLT style sheets or directly serializes to XML.

**POSSIBLE QUESTIONS – UNIT- V**

**PART A (1 Mark)**

**(Online Examinations)**

**PART B (2 Marks)**

1. What is XML?
2. Differentiate the XML & HTML
3. What are the Recommendations in XML? Mention its Name.
4. Write the XML Syntax and structure with an example
5. Define XML element
6. What is Document Type Definitions?
7. What is the purpose of Document Type Definitions?
8. What are all the XML naming rules?
9. What is XML Namespaces?
10. Define XML processor.

**Part-C (6 Marks)**

1. List the XML standard and explain with an example
2. Elucidate the XML syntax rules with an example
3. Describe the XML document structure with an example
4. Explain the namespace with an example.
5. Discuss the Document Type Definitions with an example
6. Write the advantages and disadvantages of displaying XML using CSS
7. Explain the Design goals for XML.

**COURSE NAME: INTERNET AND WEB DESIGN**

**COURSE CODE :17CCU601A**

**UNIT: V (XML)**

**SEMESTER: VI**

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWERS
1	The XML stands for_____.	Extended Markup language	Extensible Markup Language	Extended Meaningful Language	X-Mark Language	Extensible Markup Language
2	The XML files end with the extension_____.	XSL	Txt	Bmp	XML	XML
3	The XML tabs are_____.	Case sensitive	Case insensitive	Easy	Difficult	Case sensitive
4	An_____block can be to embed a CSS style sheet within an XML document.	<Bk:Book>	<Element Type>	<HTML:STYLE>	<XML:STYLE>	<HTML:STYLE>
5	The XML document must contain_____root element.	Infinite number	Exactly one	Optionally one	Two	Exactly one
6	Which of the following is an error in XML?	Nesting of elements	Not declaring the version of XML	XML files not ending with.xml Extension	Creating more than one root element	Creating more than one root element
7	The XML is a_____language.	Programming	Formatting	Meta	Processing	Meta
8	The_____method of the Xml Document.	Write	Output	Display	Print	Write
9	The_____is required to parse the XML Document.	XML-Parser	DTD file	Schema	XSL	XML-Parser
10	What is an XML namespace?	Specific spaces within an XML document.	Specific XML vocabulary	Pertaining to a particular vocabulary	Specific XML parser	Specific XML vocabulary
11	A_____is used to manipulate the contents of the XML.	XSL	DTD	CSS	DOM	DOM
12	A Schema allows_____model.	Top-down approach	Bottom-up approach	Closed model	Procedure model	Bottom-up approach
13	In XPath the XML document is conceptually viewed as a_____.	Text file	Method	Event	Tree	Tree
14	The XML document may contain___character.	ASCII	Unicode	BCD	EBCDIC	Unicode

15	An XPath is a____based language.	Object	Event	Structure	String	String
16	An XLink element that specifies the linking information is called_____	Hyperlinks	Linking elements	Inline links	X-links	Linking elements
17	An XLink attribute____is to specify the resource should be retrieved.	Show	View	Retrieve	Actuate	Actuate
18	The XLink attribute is assigned a value locator for_____resources.	Local	Extended	Simple	Remote	Remote
19	The XML Element is enclosed within_____.	< >	()	{ }	[ ]	< >
20	The_____contains a starting tag and an ending tag.	Element	Attribute	Entity	Character	Element
21	A_____is used to reference fragments of a document.	Xlink	Xpath	Xpointer	Xbase	Xpointer
22	The cascading style sheet can be_____method.	External, Inline	Internal, External	Internal, External, Inline	External only	Internal, External, Inline
23	A ----- is used to check XML for syntax errors.	XML Validator	XML parser	XML Browser	XML Webpage	XML Validator
24	The_____is a term used about text data in the XML parser.	CDATA	PCDATA	EDATA	CPDATA	PCDATA
25	A comment in XML lies within_____.	< >	" "	/* */	<!-- -->	<!-- -->
26	An XML Document can be viewed in-----	IE 3.0	IE 2.0	IE 6.0	IE X.0	IE 6.0
27	A naming collision can be resolved by using_____.	Markup text	CDATA Section	Built -in entities	Namespaces	Namespaces
28	The_____Keyword is used to create namespace prefixes.	XML	Msxml	Xmlns	Xns	Xmlns
29	The XSL object uses to hold the content of the body is ----- --.	List-block	List-item-body	List-item	List-item-label	List-item-body
30	The XSL object which formats the data is-----.	Table	Table-body	Title	Table-content	Table
31	The ----- is a security technology in XML.	Public Key Cryptography	Encryption	Hashing Technology	XML Signature	XML Signature
32	An_____is a basic building block of an XML document.	Element	Attribute	Tag	Text	Element
33	In XML_____method is used to store data.	Verify	Transport	Design	Process	Transport

34	In XML_____is a syntax errors.	Schema	Validator	Browser	Parser	Validator
35	An entity in XML is a_____.	Class	Logical information	Simple information	The flexible information storage unit	The flexible information storage unit
36	In XML schema has single precision of----- floating point.	16 bit	32 bit	8 bit	4 bit	8 bit
37	An attribute value must be quoted with_____.	Double quotes	Single quotes	Double quotes and Single quotes	Name of attributes	Double quotes
38	An XML Schemas consist of_____.	Properties and methods	Elements and attributes	Structure and data	Tables and relationships	Elements and attributes
39	An XML component that defines the structure of a document is known as a----- --.	HTML Stylesheet	DOCTYPE	DTD	#PCDATA	DTD
40	An XML data instance conforms to the DTD, the document is said to be ----- --.	Type-invalid	Type-valid	Not-type-valid	HTML document	Type-valid
41	The XML Schemas can be eliminate_____.	Intersection table	Global elements	Normalized table	Parameters	Global elements
42	A character is used to declare_____.	Elements	Parameter entity	Attribute defaults	Enumeration	Parameter entity
43	The conditional sections of DTDs are often used with_____.	Elements	Entities	Attributes	Parameters	Entities
44	The materialize XML documents is to use_____.	DTD	XSLT	HTML	SOAP	XSLT
45	In XML schema Boolean type holds-----.	True, False	1,0	True ,false and 1,0	True,False and any number except 0	True ,false and 1,0
46	The DTD is defined outside the XML document it is called_____.	Public file	External subset	Internal subset	System file	External subset
47	In Document type definition_____keyword used in XML.	External	Doctype	System	Subset	System
48	An XML is a_____state method.	Static	Dynamic	Static and Dynamic	Partial static and Dynamic	Dynamic
49	The_____is a subset of SGML.	XML	HTML	XSL	SGML	XML

50	In XML empty tags are close with_____slash.	Backslash	Forward slash	Backslash and Forward slash	Black slash	Forward slash
51	In_____method is used to the XML document into the memory.	Get( )	Post( )	Load( )	Parse( )	Load( )
52	The XSL stands for_____.	Extensible standard Language	Extensible Stylesheet Language	Extended Simple Language	Extended Standard Language	Extensible Stylesheet Language
53	An XML document begins at a_____node.	Root	Context	Location path	Expanded name	Context
54	The element frequency is specified by using the_____.	Element type indicator	Occurrence indicator	Generic identifier	Attributes elements	Occurrence indicator
55	The_____tag is used in XML.	Predefined	Custom	Defined	Predefined and custom	Custom
56	A schema document use_____syntax.	EBNF	XML	Java	DTD	XML
57	A-----incorporate the element of CSS.	XSL	XML	DSL	Xlink	XSL
58	A_____method is to replace the structure.	CSS	XSL	DOM	Schemas	Schemas
59	A_____is the root element of every XML Schema documents.	<Xml>	<Root>	<Schema>	<Xsl>	<Schema>
60	In XML Schema_____defines an element.	Element	ElementType	Element	Xml:Element	ElementType