WEB TECHNOLOGIES

UNIT 1

1. History of the Internet and World Wide Web

1.1 History of Internet

The U.S. Department of Defense (DOD) has decided to build a large scale computer in 1960.This network was developed for establishing the communication among the computers, accessing of remote computers and for sharing each other's program.

This type of network is having robustness. Robustness means in a network even if one of the computer in a network gets failed the network should continue its work. The DOD's Advanced Research Project Regency (ARPA) funded this project hence this project was named as ARPANET. The first node of the network was established in 1969.

Although this network was established for communication, it was only connected to the laboratories and to the ARPA funded projects. The universities and other important organizations were not connected with ARPANET. Hence many small networks were developed for their own needs.

In 1986, a national network NSFnet was created. The NSFnet connected five universities which were funded by the NSF (National Science Foundation) itself. Afterwards it was available to other universities and research laboratories. The network continued to grow very rapidly. By 1992, NSFnet connected around 1 million computers around the world. Thus NSFnet and ARPANET collectively were known as Internet.

1.2 Internet

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.

Using Internet many people can share resources and communicate with each other. To have internet service one must go to the service providers. This means computer must be connected to the **I**nternet **s**ervice **p**roviders (ISP) through cable modem, phone line modem.

`There are some privately owned internet service providers from which we can hire the internet services. The setup required for using Internet is as follows.

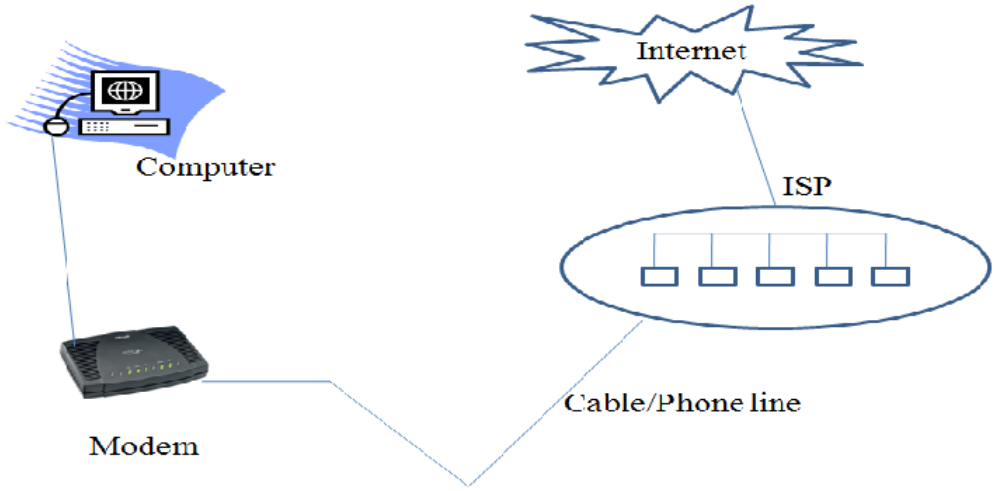


Fig: Setup required for Internet

Suppose if we want to use internet on computer then there must be a Network Interface Card (NIC) inside the computer that connects to the local area network (LAN).Then this LAN gets connected to ISP. If computer is not residing in the LAN then it must be connected to the local ISP's using cables/phone lines.

1.3 WWW

World Wide Web (WWW) is a collection of software and corresponding protocols used to access the resources over the network.

The World Wide Web is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia, and navigate between them via hyperlinks.

History of WWW

The concept of WWW was introduced by Sir **Tim Berners-Lee** the contractor at European organization for Nuclear Research (CERN), Switzerland in 1980.He built a personal database of people and software models and used hypertext so that each new page of information was linked to an existing page. In 1990, Berners-Lee introduced Hypertext transfer protocol(HTTP),Hypertext markup language(HTML) and web browser. This is the time when first website http://info.cern.ch/ was built.

During 1992-1995, along with HTTP a new protocol was named **Gopher** protocol which provides access to content through hypertext menus presented at a file system rather than through HTML files. In 1993 a new web browser with graphical user interface **Mosaic** got introduced.

In 1994, the **World Wide Web Consortium (W3C)** was founded by Berners-Lee at the **Massachusetts Institute of Technology (MIT)** with support from the **Defense Advanced Research Project Regency (DARPA).** This organization was built for creating standards and recommendations to improve quality of web. Berners-Lee made the web freely with no patents. The W3C decides that their standards must be based on royalty-free technology, so they can be easily adopted by anyone. At the end of 1994 a large number of websites got activated with popular web services.

During 1996-1998 trade marketing started using WWW and E-Commerce introduced. During 1999-2000 many entrepreneurs started selling their ideas using **dotcom** boom. From 2002 WWW has got an evolving nature due to various developments such as online-booking, efficient search engines and agent based technologies, Face book, and Social networking sites and so on.

1.4 Web Browser

It is a kind of software which is basically used to use resources on web. Over the networks, two computers communicate with each other called Client & Server.

Client→ when the request made by one computer then the computer is called Client.

Server→ when the request gets served by another computer then the computer is called Server.

The exchange of information takes place via client-server communication.

When user wants some web document then he makes the request for it using the web browser. The browsers are the programs that are running on the client machines. The request gets served by the server and the requested page is returned to the client. It is getting displayed to the client on the web browser.

→The commonly used web browsers are

Internet explorer, Mozilla fire fox, Netscape navigator, Google chrome

Web browsers supports protocols like Hypertext transfer protocol (HTTP).

`1.5 Web Servers

Web server is a special type of server to which the web browser submits the request of web page which is desired by client. The web servers which are used popularly are Apache &IIS server.

Web server operations→

We often browse internet for several reasons. It is needed to know how a web page demanded by us gets displayed on web browser. The explanation is as follows.

1. User types website address for demanding the desired webpage.

Ex: **http://www.vtubooks.com/home.aspx**

Then the home page of website appears on screen. The web address is divided into three parts. The first part is protocol. The http is a hypertext transfer protocol which tells the web browser that the user wishes to communicate with web server on port 80.port 80 is reserved for the communication between web server and web browser. The second part is server address. This tells the web browser which server it needs to contact in order to retrieve the information we are looking for. The web browser communicates with a Domain name server(DNS) to find out IP addresses for the website. All communications on the internet use IP addresses for communications. Using the numeric address for accessing the web server is avoided because it is easier to remember textual information rather than that of numeric one. Hence normally the web server's addresses are textual. The third part of this address denotes the resource user wants to see.

2. The web browser, having found the IP address it needs by communicating with the name server, then sends a request directly to the web server, using port 80, asking for the file **home.aspx**.

3. The web server sends the html for this page back to user's web browser, which reads the HTML tags and formats them for viewing on screen. If there are additional files needed in order to show the web page the web browser makes additional requests for each of these.

General characteristics→

1.There are two types of directories for the server. The roots of these directories are **document root** and **server root.**

2.The files that are stored directly in the document root directory are available to the clients using the top-level URL. Normally clients do not access the document root directly.

3.Normally server stores the documents that are readable to its client outside the document root.

4.The **virtual document trees** are the areas from which the server can serve the documents to its clients.

5.If the documents stored in the sub directories then client can refer to these web documents using the URL with a particular file path to that directory from the document root directory.

6.Some servers allow the access to the web documents that are in the document root of other machine. Such servers are called **proxy servers.**

7.Web servers support various protocols such as **HTTP, FTP, GOPHER, News and mail.**

8.All the web servers can interact with the database systems with the help of common gateway interface (CGI) or server side script.

Apache→

It is an excellent server because of its Reliability and Efficiency. It is popular because it is an open source software. Apache web server is best suitable for UNIX systems but it can also be used on windows box. The Apache web server can be configured as per the requirements using the file **httpd.conf**. This file is present in Apache software package.

IIS→

The **Internet information services** or **Internet information server** is a kind of web server provided by Microsoft. This server is most popular on windows platform.

The differences between Apache & IIS servers are as follows.

S.NO	Apache web server	IIS web server
1.	It is useful on both Unix & Windows platform	It is used on windows platform.
2.	It is an open source product.	It is a vendor specific product and can be used on windows products only.
3.	It can be controlled by editing the configuration file httpd.conf	Its behaviour is controlled by modifying the windows based management programs called IIS snap in. We can access IIS snap-in through the Control panel->Administrative tools

1.6 Uniform Resource Locator(URL)

URL is used to identify documents on internet. In most web browsers, the URL of a web page is displayed on top inside an address bar. There is variety of URL depending upon the type of resources.

URL Formats→

The general form of URL is **Scheme: Address**

EX: protocol://username@hostname/path/filename

The **scheme** specifies Communication protocol such as http, ftp, gopher, file, mailto, news.....

The most commonly used protocol for communication between web server and web browser is **HTTP**(hypertext transfer protocol), which is based on request response mechanism.

Using **http** the **address** part of URL can be written as-- //Domain_name/path_to_document

file is another scheme used in URL. It allows to reside the document in the client's machine from which the web browser is making out the demand.

Using **file** the **address** part of URL can be written as-- file://path_to_document

The default port number for http is 80.Any URL does not allow spaces in it. But there are special characters like &, %,..

URL Paths→

The path to the web document is similar to the path to the particular file present in the folder. In this path the directory names and files are separated by the separator characters. The character that is used as a separator is **slash**. Unix system uses forward slash where as windows system uses backward slash.

Ex: http://www.mywebsite.com/mydocs/index.html

The URL path that includes all the directories along the path to the file is called **complete path.**

Sometimes the base URL path is specified in configuration file of server. In such a case does not needed to specify complete path for accessing the particular file such a path is called **partial path.**

Ex: http://www.mywebsite.com

This indicates that file mydocs/index.html is specified in the configuration file.

2 HTML Common tags

2.1 Introduction

HTML stands for **Hypertext Markup Language**.HTML is a subset of SGML(Standard general markup language). It is used to display the document in the web browsers. HTML pages can be developed to be simple text or to be complex multimedia program containing sound, moving images and java applets. HTML is considered to be the global publishing format for Internet. It is not a programming language. HTML was developed by Tim Berners-Lee. HTML standards are created by a group of interested organizations called W3C (World Wide Web consortium). In HTML formatting is specified by using tags. A tag is a format name surrounded by angle brackets.

The HTML program should be written with in tags <html> and </html>. The tags are not case sensitive i.e., <head>, <HEAD> and <Head> is equivalent. <html> indicates the start of the program and </html> denotes ending of the program. Use of slash(/) in the angular bracket indicates end of that particular tag.

Structure of an HTML document:

The HTML document contains three main parts.

1. DOCTYPE declaration
2. <head> section
3. <body> section

Basic HTML Document structure

The basic document structure is as follows.

```
<!DOCTYPE.....>
<html>
    <head>
        <title>.....</title>
    </head>
    <body>
          .....
          .....
          .....
     </body>
</html>
```

The DOCTYPE specifies document type. Comments in HTML documents start with <! and end with >. Each comment can contain as many lines of text as you like. If comment is having more lines, then each line must start and end with -- and must not contain -- within its body.

<! -- this is a comment line - -
-- which can have more lines - ->

The head part acts as a header of a file and contains some information like setting of title of webpage.

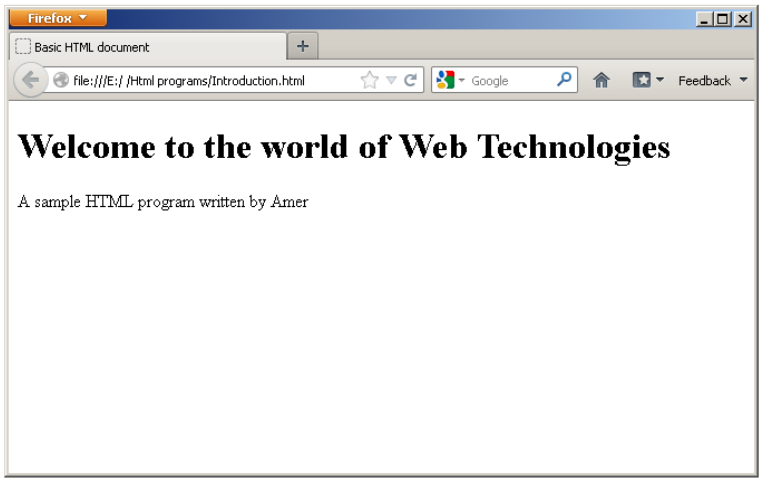
```
<head>
    <title> Basic HTML document </title>
</head>
```

The body part will help us to create a look and feel of the webpage. It contains the actual user information that is to be displayed on the screen.

```
<body>
    <h1> Welcome to the world of Web Technologies</h1>
    <p> A sample HTML program written by Amer </p>
```

The basic document is shown below.

```
</body>
<html>
<head>
<title> Basic HTML document </title>
</head>
<body>
<h1> Welcome to the world of Web Technologies</h1>
<p> A sample HTML program written by Amer </p>
</body>
</html>
```



Basic HTML tags

1. Body tag:

Body tag contain some attributes such as bgcolor, background etc. bgcolor is used for background color, which takes background color name or hexadecimal number and #FFFFFF and background attribute will take the path of the image which you can place as the background image in the browser.

<body bgcolor="#F2F3F4" background= "c:\vnits\imag1.gif">

2. Paragraph tag:

Most text is part of a paragraph of information. Each paragraph is aligned to the left, right or center of the page by using an attribute called as align.

<p align="left" | "right" | "center">

3. Heading tag:

HTML is having six levels of heading that are commonly used. The largest heading tag is <h1> . The different levels of heading tag besides <h1> are <h2>, <h3>, <h4>, <h5> and <h6>. These heading tags also contain attribute called as align.

<h1 align="left" | "right" | "center"> . . . . <h2>

4. hr tag:

This tag places a horizontal line across the system. These lines are used to break the page. his tag also contains attribute i.e., width which draws the horizontal line with the screen size of the browser. This tag does not require an end tag.

<hr width="50%">.

5. base font:

This specify format for the basic text but not the headings.

<basefont size="10">

6. font tag:

This sets font size, color and relative values for a particular text.

<font size="10" color="#f1f2f3">

7. bold tag:

This tag is used for implement bold effect on the text

 …….

8. Italic tag:

This implements italic effects on the text.

…….

9. strong tag:

This tag is used to always emphasized the text

…….

10. tt tag:

This tag is used to give typewriting effect on the text

<tt>……..</tt>

11. sub and sup tag:

These tags are used for subscript and superscript effects on the text.

_{…….}

^{……..}

12. Break tag:

This tag is used to the break the line and start from the next line.

13. &amp &lt &gt   &quot

These are character escape sequence which are required if you want to display characters that HTML uses as control sequences.

Example: < can be represented as &lt.

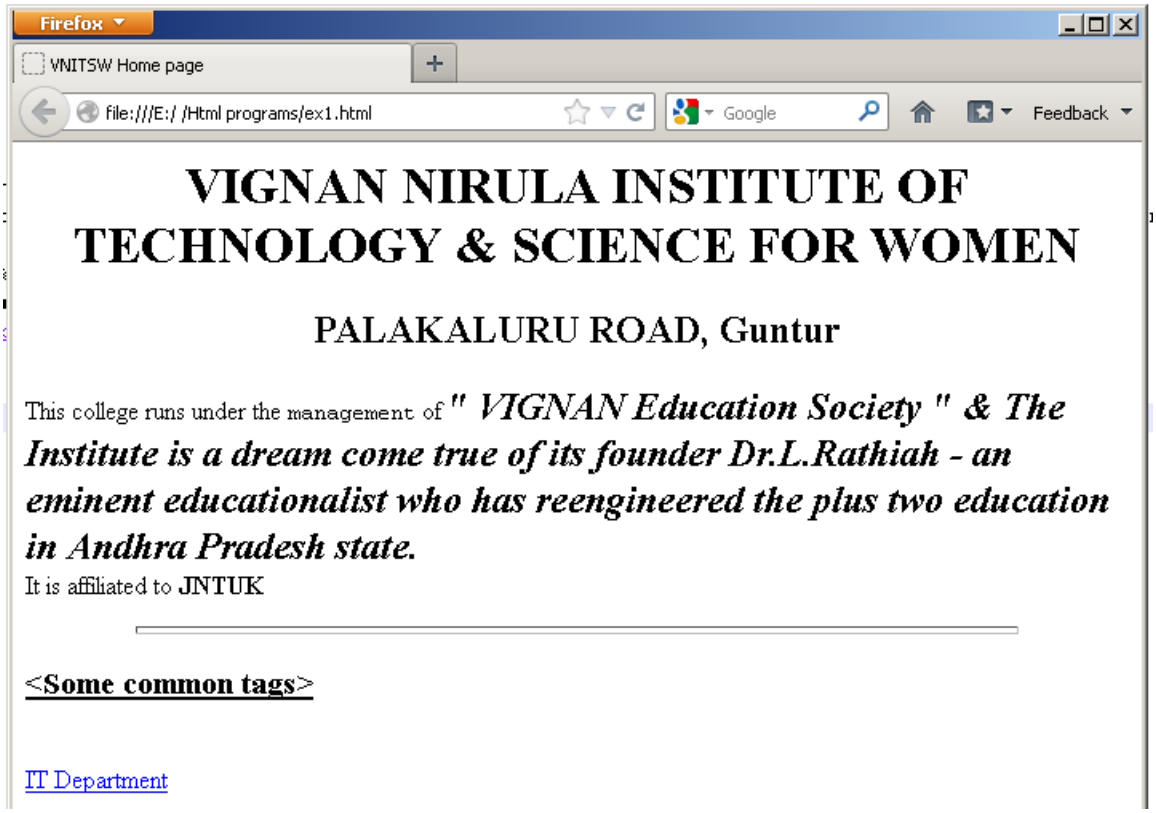
14. Anchor tag:

This tag is used to link two HTML pages, this is represented by <a> some text href is an attribute which is used for giving the path of a file which you want to link.

Example 1: HTML code to implement common tags.

ex1.html

```
<html>
<head> <! -- This page implements common html tags -->
<title> VNITSW Home page </title>
</head>
<body >
<h1 align="center">  VIGNAN NIRULA INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN</h1>
<h2 align="center"> PALAKALURU ROAD, Guntur</h2>
<basefont size=40>
<p> This college runs under the <tt>management</tt> of <font size=5> <b><i>&quot
VIGNAN Education Society &quot &amp The Institute is a dream come true of its founder Dr.L.Rathiah - an eminent educationalist who has reengineered the plus two education in Andhra Pradesh state. </i></b></font><br>
It is affiliated to <strong> JNTUK</strong>
<hr size=5 width=80%>
<h3> <u>&ltSome common tags&gt</u> </h3><br>
<a href="E:/Html programs/Introduction.html"> IT Department</a><br>
</body>
</html>
```



2.2 Lists

One of the most effective ways of structuring a web site is to use lists. Lists provides straight forward index in the web site. HTML supports three types of lists. They are

1. Unordered list
2. Ordered list
3. Definition list

1. Unordered list: Unordered lists are used to make a list of bulleted points. Each bulleted point or line should be placed between the following tags.

<ul>
** Text to be displayed
**
**
**

The unordered (bulleted) lists are made up of sets of list items. This tag is used to write list items

**<ul type="disc" | "square" | "circle" > ….. **

This tag is used for basic unordered list which uses a bullet in front of each tag; everything between the tags is encapsulated within <li> tags

Example:

```
<html>
<head>
<title>Creating Unordered Lists</title>
</head>
<body>
<ul type=”disc”>
<li>Apple</li>
<li>Mangoes</li>
<li>Orange</li>
</ul>
<ul type=”square”>
<li>Apple</li>
<li>Mangoes</li>
<li>Orange</li>
</ul>
<ul type=”circle”>
<li>Apple</li>
<li>Mangoes</li>
<li>Orange</li>
</ul>
</body>
</html>
```

Output:

- **Apple**
- **Mangoes**
- **Orange**
  - **Apple**
  - **Mangoes**
  - **Orange**
    - **Apple**
    - **Mangoes**
    - **Orange**

2. Ordered list: Using Ordered list, we can display the text carrying large roman numbers, small roman numbers, and letters etc. Ordered list usually initiated using the following tag,

**<ol type=”1” | ”a” | “I” start=”n”>….. **

Here type attribute is used to select numbers, letters, large roman numbers and small roman numbers.

Value for type attribute	Description	Examples
1	Arabic numerals	1,2,3,4,5
A	Capital letters	A,B,C,D,E
a	Small letters	a,b,c,d,e
I	Large roman numerals	I,II,III,IV,V
i	Small roman numerals	i,ii,iii,iv,v

Example:

```
<html>
<head>
    <title>Creating Unordered Lists</title>
</head>
<body>
    <ol type=”1”>
        <li>Apple</li>
        <li>Mangoes</li>
        <li>Orange</li>
    </ol>
    <ol type=”A”>
        <li>Apple</li>
        <li>Mangoes</li>
        <li>Orange</li>
    </ol>
    <ol type=”I”>
        <li>Apple</li>
        <li>Mangoes</li>
        <li>Orange</li>
    </ol>
    <ol type=”i” start=”4”>
        <li>Apple</li>
        <li>Mangoes</li>
        <li>Orange</li>
    </ol></body></html>
```

Output:

1. **Apple**
2. **Mangoes**
3. **Orange**

A. **Apple**
B. **Mangoes**
C. **Orange**

I. **Apple**
II. **Mangoes**
III. **Orange**

iv. **Apple**
v. **Mangoes**
vi. **Orange**

3. Definition Lists: The definition list is a special kind of list for providing terms followed by a short text definition or description for them. Definition lists are contained inside the **<dl>….. </dl>** element then contains alternating **<dt> and <dd>** elements**.<dt>** is a sub tag of the <dl> tag called as definition term, which is used for marking the items whose definition is provided in the next data definition.**<dd> i**s a sub tag of the <dd> tag, definition of the terms are enclosed within these tags. The definition may include any text or block.

Example: HTML code showing list tags.

```
<html>
<head>
<title> list page </title>
</head>
<body>
        <dl>
            <dt>Unordered List</dt>
                <dd>A list of bullet points</dd>
            <dt>Ordered List</dt>
                <dd>An ordered list of points, such as numbered set of steps</dd>
            <dt>Definition List</dt>
                <dd>A list of terms and definitions</dd>
        </dl>
</body>
</html>
```

Output:

Unordered List
    A list of bullet points
Ordered list
    An ordered list of points, such as numbered set of steps
Definition list
    A list of terms and definitions

Nesting Lists: List with in other lists is called as nested lists. For example

```
<html>
<head>
        <title>Creating Nested Lists</title>
</head>
<body>
        <ol type="I">
                <li>Item One</li>
                <li>Item Two</li>
                <li>Item Three</li>
                <li>
```

Output:

I. Item One
II. Item Two

III. Item three

i. Item one
ii. Item two
iii. Item three

```
                    <ol type="i">
                        <li>Item One</li>
                        <li>Item One</li>
                        <li>Item One</li>
                    </ol>
                </li>
            </ol>
        </body>
        </html>
```

2.3 Tables

The main use of table is that they are used to structure the pieces of information and to structure the whole web page. The HTML table allows arranging data -- text, preformatted text, images, links, forms, form fields, other tables, etc. -- into rows and columns of cells.

Below are some of the tags used in table.

<table align="center" | "left" | "right" border="n" width="n%" cellpadding="n" cellspacing="n">……</table>

Everything that we write between these two tags will be within a table.

The following are the attributes for element <table>…</table>

align: Using this attribute we can direct the placement of a given table in the web browser window. Alignment is performed horizontally by assigning the values LEFT/RIGHT/CENTER respectively.

background: Using this attribute we can place image or other graphics as background to the table. It usually carries a URL path corresponding to the given image or graphics.

bgcolor: Using this attribute we can set the background color to the table.

bordercolor: This attribute is used to set the border color.

border: This attribute is used to set width of the border of the table. If its value is '0', displays a table with no border.

cols: This attribute is used to specify number of columns in the table.

height: This attribute is used to specify height of the given table

width: This attribute is used to specify width of the given table

cellpadding: It creates space between the cell edges and content in the cell. It takes the amount of padding or space required in percentage value or pixels. For example cellpadding="5" or cellpadding="2%".

cellspacing: It creates the space between the cells. Like cellpadding, cellspacing takes a value either in percentage or in pixels. For examples, cellspacing=”5” or cellspacing=”2%”

The following are the attributes for element <tr>…</tr>

align: It directs the alignment of text in a given row of a table. Its value can be LEFT/CENTER/RIGHT.

bgcolor: It is used to set the background color of the blocks of a table corresponding to a given row.

bordercolor: It sets the border color for a given row of a table.

valign: It directs the vertical alignment of data corresponding to a given row of the table. It can have values as TOP/BOTTOM/MIDDLE.

The following are the attributes for element <th>…</th>

align: It directs alignment of data in the header column of the table. Its value can be LEFT/RIGHT/CENTER.

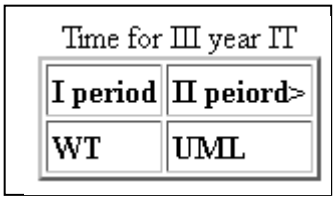
bgcolor: It is used to set the background color for the table headers.

bordercolor: It sets the border color for the corresponding table headers.

valign: It directs vertical alignment of data in the table header.

width: It sets the width of the table header.

Example : HTML code showing the use of table tag




```
<html>
<head>
<title> table page</title>
</head>
<body>
<table align="center" cellpadding="2" cellspacing="2" border="2">
<caption>Time for III year IT</caption>
        <tr>
                <th> I period </th>
                <th> II peiord> </th>
        </tr>
        <tr>
                <td> wt </td>
                <td> uml</td>
        </tr>
</table></body></html>
```

2.4 Images

The HTML **img** tag is used for embedding images into an HTML document. To use this tag, the images you specify in the **src** attribute needs to be available on a web server or on your system. The following are the attributes for **img** tag.

name: Assigns a name to an image. This is used when referencing the image with style sheets or scripts. However, you should use the id attribute instead.

alt: Specifies a alternate message when image is not displayed

src: Location of the image

align: For alignment (left, center, right, justify)

width: Specifies the width of the image.

height: Specifies the height of the image.

border: Size of the image border. For no border use 0 (zero).

hspace/vspace: Used to place gaps or whitespaces between the text and images.

Example:

```
<html>
<head>
        <title>Inserting images into Web Page</title>
</head>
<body>
<p><b> Vignan Nirula Institute of Technology & Science for WOmen.</b></p>
<img src="e:\Vignan.jpg" width="225" height="151" alt="VNITSW">
</body>
</html>
```

2.5 Frames

HTML frames allow authors to present documents in **multiple views**, which may be independent windows or sub windows. Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced. For example, within the same window, one frame might display a static banner, a second a navigation menu, and a third the main document that can be scrolled through or replaced by navigating in the second frame. Frames provide a pleasing interface which makes your web site easy to navigate. When we talk about frames actually we are referring to frameset, which is a special type of web page. The frameset contains a set of references to HTML files, each of which is displayed inside a separate frame. There are two tags related to frames i.e., frameset and frame.

<frameset cols=" % , %" | rows=" % , %">……..</frameset>

<frame name="name" src="filename" scrolling =" yes" | "no" frameborder ="0"|"1">

Tags in Frames:

*The **Frameset** Tag*

- The <frameset> tag defines how to divide the window into frames
- Each frameset defines a set of rows **or** columns
- The values of the rows/columns indicate the amount of screen area each row/column will occupy

*The **Frame** Tag*

- The <frame> tag defines what HTML document to put into each frame

Attributes of <frameset> element:

cols: This attribute specifies how many columns are in the frameset

rows: This attribute specifies how many rows are in the fameset

border: This attribute specifies the width of the border of each frame in pixels.

frameborder: This attribute specifies whether three dimensional border should be displayed between frames or not. If it's value is 1, indicates border should be displayed.

framespacing: This attribute specifies the amount of space between frames in a frameset.

Attributes of <frame> element:

src: This attribute indicates the file that should be used in the frame.

name: This attribute allows to give name to the frame.

frameborder: This attribute specifies whether borders of the frame should be shown or not. Its value 1 indicates yes, 0 indicates false.

marginwidth: This attribute specifies the width of the space between the left and right of the frame border's and the fames content.

marginheight: This attribute specifies the height of the space between the top and bottom of the frames borders and its contents.

scrolling: This attribute is used to place scroll bar in web pages. It takes YES/NO/AUTO.

- Yes, indicates the frame must always contain scroll bar whether or not they are required.
- No, indicates the frame must not contain scroll bar even it is required.
- Auto, indicates that the browser should indicate scroll bars when it is required.

Crating links between pages: One of the most popular uses of frames is to place navigation bars in one frame and then load the pages with the content into separate frame. This can be done using target attribute as follows:

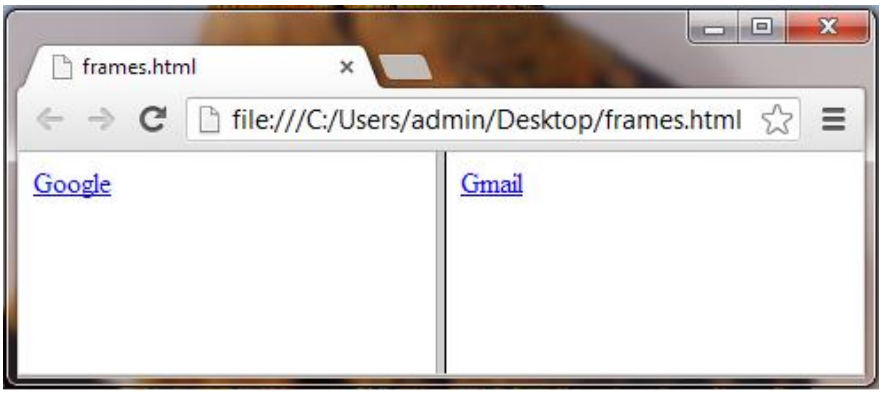
```
<html>
<frameset cols="200,*">
        <frame src="d:\1.html" name="leftframe">
        <frame src="d:\2.html" name="rightframe">
</frameset>
</html>
```

1.html

```
<html>
<body>
        <a href="http://www.google.com"  target="rightframe">Google</a>
</html>
```

2.html

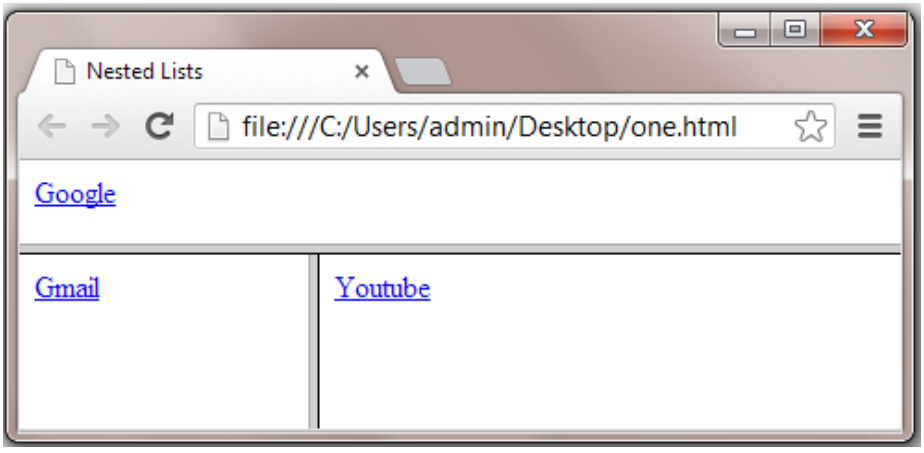
```
<html>
<body>
        <a href="http://www.gmail.com"  target="rightframe">Gmail</a>
</body>
</html>
```



Nested Frameset: A <frameset> element with in other <frameset> element is known as nested frameset. This can be shown as follows:

Example :

```
<html>
<head>
        <title> Nested Lists </title>
</head>
<frameset rows="25%,50%">
        <frame name="a" src="d:\1.html">
        <frameset cols="25%,50%">
                <frame name="b" src="d:\2.html">
                <frame name="abc" src="d:\3.html">
        </frameset>
</frameset>
</html>
```



2.6 Forms

Forms are the best way of adding interactivity of element in a web page. They are usually used to let the user to send information back to the server but can also be used to simplify navigation on complex web sites. The tags that use to implement forms are as follows.

<forms action="URL" method = "post" | "get">…….</form>

When get is used, the data is included as part of the URL. The post method encodes the data within the body of the message. Post can be used to send large amount of data, and it is more secure than get.

Attributes of form element:

action: The action attribute indicates which page or program on the server will receive the information from this form when a user presses the submit button.

method: This attribute is used to send form data to the server. This can be done in two ways:

- The **get** method, which sends data as part of the URL
- The **post** method, which hides data in the http headers

id: This attribute is used to provide a unique identifier for the <form> element.

name: This attribute is used to provide name for the form

Form Controls:

The following are the different types of form controls

1. **Text input controls**
2. **Buttons**
3. **Checkboxes**
4. **Radio buttons**
5. **Select boxes**

1. Text Input Controls: There are three types of text input controls, they are

a) Single line text input controls: Used for taking line of text as input, such as search boxes, names etc. They are created using <input> element whose type attribute value is "text" as shown below.

<input type="text" name="txt1" value="searchfor" size="30" maxlength="40">

The following are the attributes of <input> element.

type: This attribute indicates type of input control

name: This attribute is used to give name for the text control

value: This attribute is used to provide initial value for the text input control

size: This attribute is used to specify width of the text input control

maxlength: This attribute is used to specify maximum number characters can be entered into the text input control

b) Password input controls: This control is used to secured data as input. They are created using <input> element whose type attribute value is "password" as shown below.
<input type="password" name="pwd" value="" size="30" maxlength="40">

c) Multi line text input controls: This control is used to take more than one line as input from the user. They are created using <textarea> element as shown below.
<textarea name="txt2" rows="10" cols="10">
Here, name attribute provides name for the control, rows and cols indicates maximum number of lines and number characters for each line the control accepts from user

2. Buttons: Buttons are most commonly used to submit, clear, reset a form data and also used to trigger client side scripts. We can create buttons in three ways as shown below:

Using an <input> element with a type attribute whose value is submit, reset, or button
In this method, the type attribute can take the following values:

- **submit**, which creates a button which automatically submits a form
  <input type="submit" name="sub" value="Submit">
- **reset**, which creates a button that automatically resets form controls to their initial values
  <input type="reset" name="res" value="Reset">
- **button**, which creates a button that is used to trigger a client side script
  <input type="button" name="but" value="Submit">

Using an <input> element with a type attribute whose value is image
<input type="image" src="d:\submit.jpg" alt="Submit" name="sub" value="Submit">
Here attribute src specifies the source of the image file, alt attribute provides alternate text for the image, when the image is not loaded.

Using an <button> element
<button type="submit" name="button1" value="Submit">
<button type="reset" name="button2" value="Reset">
<button type="button" name="button3" value="Submit">

3. Checkboxes: Checkboxes provide simple yes or no response with one control and allows selecting several items from a list of possible options. A check box is created using <input> element whose type attribute value is "checkbox" as shown below.

<input type="checkbox" name="box1" value="html" checked="checked" size="10">HTML
<input type="checkbox" name="box1" value="dhtml">DHTML
<input type="checkbox" name="box1" value="xml">XML
<input type="checkbox" name="box1" value="js">JS

Attributes:
type: Indicates type of the control
name: gives name of the control
value: Provides name for the control
checked: Indicates when the page loads, the checkbox should be selected
size: Indicates the size of the checkbox in pixels

4. Radio buttons: Unlike checkboxes, radio buttons allows single selection. They are created using <input> element with type attribute whose value is "radio" as shown below.

```
<input type="radio" name="radio1" value="first" checked="checked">Male
<input type="radio" name="radio1" value="second">Female
```

type: Indicates type of the control
name: gives name of the control
value: Provides name for the control
checked: Indicates when the page loads, the radio button should be selected
size: Indicates the size of the radio button in pixels

5. Select Boxes: A drop down select box allows user to select one item from a drop down menu. Drop down select boxes takes less space than a group of radio buttons. They are created by using <select> element, in which each individual option within that list is contained within an <option> element as shown below.

```
<select name="satatelist" size="3" multiple="multiple">
    <option selected="selected">Select Your Sate</option>
    <option>Andhra Pradesh</option>
    <option>Madhya Pradesh</option>
    <option>Uttar Pradesh</option>
    <option>Himachal Pradesh</option>
</select>
```

Attributes of <select> element:
name: This attribute is used to provide name for the control
size: This attribute can be used to present a scrolling list box. If it's value is 3, the select box displays three options at a time.
multiple: This attribute allows to select multiple items from the select box.
Attributes of <option> element:
selected: This attribute specifies that the option should be initially selected.

2.7 Links and Navigation

The link acts as a pointer to some webpage or resource. Use of hyperlink in the webpage allows that page to link logically with other page.

A link is specified by using <a> element. Anything between<a> and tags become part of the link a user can click in a browser. The steps to specify web link in a page are as follows.

a) Beginning of web link can be specified by .

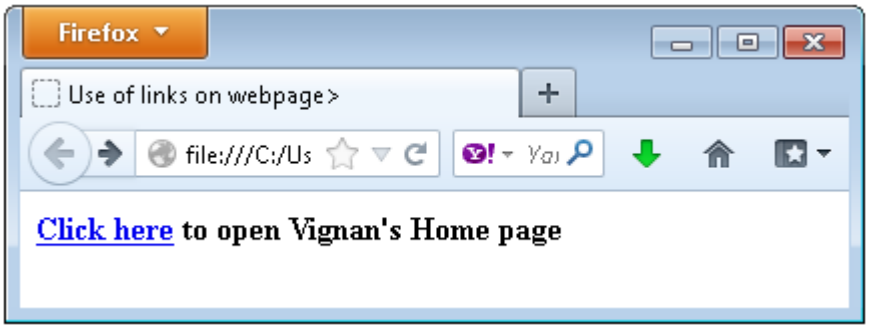
Inside the double quotes mention URL of desired links.

b) Write some text that should act as a hyperlink.

c) Ending of web link can be specified by

Example:

```
<html>
<head>
<title>Use of links on webpage>
</title>
</head>
<body>
<a href="http://www.vignanuniversity.org"><b>Click here</b></a><b> to open Vignan's Home page</b>
</body>
</html
```



If we click on the hyperlink then we will get the Vignan's Homepage.

Use of image as a Link

Similar to a text we can set an image itself as a link. Following HTML document explains this idea.

Example:

```
<html>
<head>
<title>Use of image as a hyperlink
</title>
</head>
<body>
<h1> Vignan Nirula Institute of technology & science for women </h1>
<a href="http://www.vignanuniversity.org">
<img src="d:\Vignan.jpg" alt="Vignan" />
</a>
<br> Vignan Niruls's Home page</br>
</body>
</html>
```

In this HTML document a hyperlink to an image is existed. When we click on the image the html page is referred by <a href> tag must be opened. By default this page gets opened in the browser's window. If we want to get that link opened in another window we can mention **_target** property.

Various targets can be

- **_self** loads the page into the current window.
- **_blank** loads the page into a new separate browser window.
- **_parent** loads the page into the frame that is superior to the frame the residing hyperlink.
- **_top cancels** all frames, and loads in full browser window.

One can adjust the position of the image using the property **align**.

****

The uses of hyperlinks for a web document are

- One can link logically related documents together using links in the webpage.
- Use of links enhances readability of the web document.
- User can click on the link and can learn more about a sub topic and then can return to the main topic. This navigation within the WebPages is possible due to the links.

3. Cascading style sheets (CSS)

3.1 Introduction

The cascading style sheet is markup language used in web document for presentation purpose. The purpose of CSS is to separate web content from web presentation.

Advantages:

1. By combining CSS with HTML document, considerable amount of flexibility into the content submission can be achieved.

2. Separating out the style from actual contents help in managing large scale complex sites. So, CSS facilitates publication of contents in multiple presentation formats.

3. If CSS is used effectively then global style sheet can be applied to a web document. This helps maintaining consistency in web document.

4. If a small change needs to be done in the style of web content, then CSS makes it more convenient.

3.2 Levels of CSS

A Cascading Style Sheet is a collection of CSS rules. A CSS rule made up of two parts:

i. The selector, which indicates to which element or elements the declaration applies to

ii. The declaration, which sets out how the elements should be styled. The declaration is also split into two parts, separated by a colon:

- A property, which is the property of the selected elements
- A value, which is a specification for the property

selector { property : value ; property : value ; …….}
----------Declaration----------

h1 { font – family : arial } → This CSS rule applies to all <h1> elements

There are three mechanisms by which we can apply styles to our HTML documents. They are

a) Internal style Sheet

b) Embedded Style Sheet

c) External Style Sheet.

a). Internal (Inline) Style Sheet: Internal styles are styles that are written directly in the HTML tag on the document. The normal rules of CSS apply inside the style attribute. Each CSS statement must be separated with a semicolon ";" and colons appear between the CSS property and its value. For example,

** A new background and font color with inline CSS **

Advantages of Inline Styles

1.Inline styles have the highest precedence. That means they are going to be applied no matter what. The only styles that have higher precedence than inline style.

2.Inline styles are easy and quick to add.

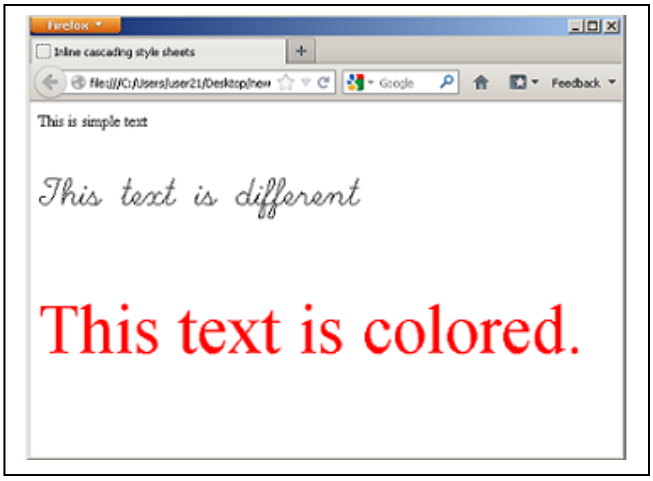
Disadvantages of Inline Styles

1.Inline styles must be applied to every element you want them on.

2. It's impossible to style pseudo-elements and -classes with inline styles.

Example:

```
<html>
<head>
<title>Inline cascading style sheets</title>
</head>
<body>
<p> This is simple text</p>
<p style="font-size:30pt;font-family:script">This text is different</p>
<p style="font-size:50pt;color:#ff0000">This text is colored.</p>
</body>
</html>
```



b) Embedded Style Sheet: Embedded Style Sheets refer to embed style sheet information into an HTML document using the style element. Embedding the style sheet information within **<style>...</style>** tags which appears only in the head section of your document.

Advantages:

The Embedded style sheet helps to decide the layout of the webpage.

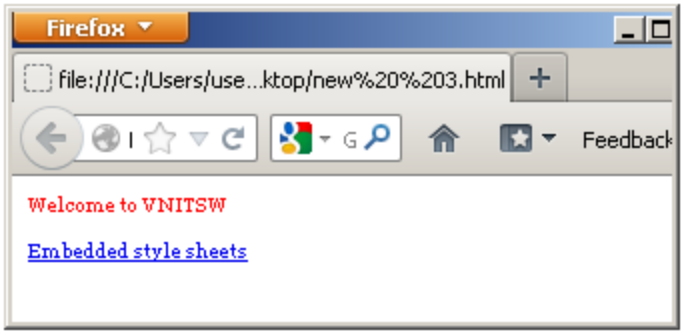
It is Useful when we want to apply the **unique style sheet** for the webpage.

Disadvantages:

When we want to apply the style to more than one documents at a time then this style sheet is of no use.

Example:

```
<html>
<head>
<style type="text/css">
  p {font-family: georgia; font-size: x-small;color:red}
</style>
</head>
<body>
<p>Welcome to VNITSW</p>
<p><a href=" C:\Users\user21\Desktop\vignan.jpg" target="_blank">Embedded style sheets</a></p>
</body>
</html>
```



c) External Style Sheet: Sometimes we need to apply particular style to more than one web documents, in such cases external style sheets can be used. The central idea of style sheet is defined in **.css** file. The style defined in **.css** file will be applied to all the web pages by using LINK tag.

External Style Sheet is specified using the LINK element within the HEAD element block to specify the URL location of the External Style Sheet. URL values may be relative or absolute.

Syntax:

<link rel="stylesheet" type="text/css" href="[Style sheet URL]">

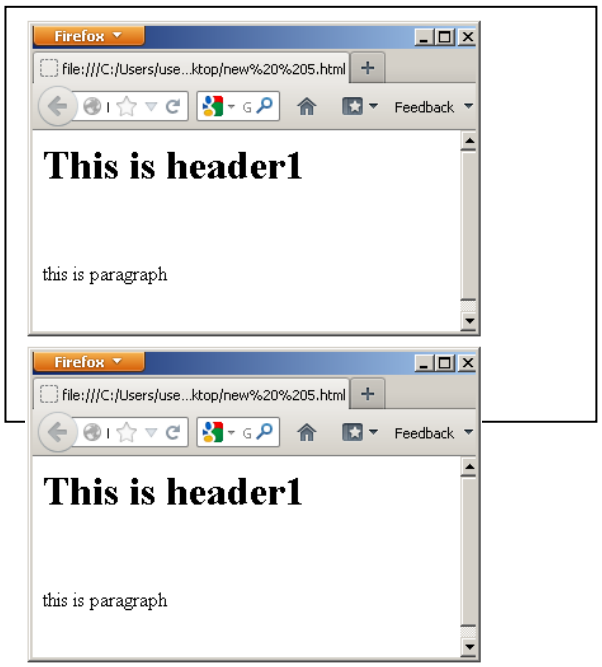
Example:

This is the style sheet file (**ex1.css**):

```
body {background-color:tan;}
h1 {color:maroon;font-size:20pt;}
hr {color:navy;}
p {font-size:11pt;margin-left:15px;}
```

Main html file:

```
<html>
<head>
<link rel=”stylesheet” type=”text/css” href=”C:\Users\user21\Desktop\ex1.css”/>
</head>
<body>
<h1>This is header1</h1><br>
<p>this is paragraph</p><br>
</body>
</html>
```



3.3 Style Specification formats

A Cascading Style Sheet is a collection of CSS rules. A CSS rule made up of two parts:

i. The selector, which indicates to which element or elements the declaration applies to

ii. The declaration, which sets out how the elements should be styled. The declaration is also split into two parts, separated by a colon:

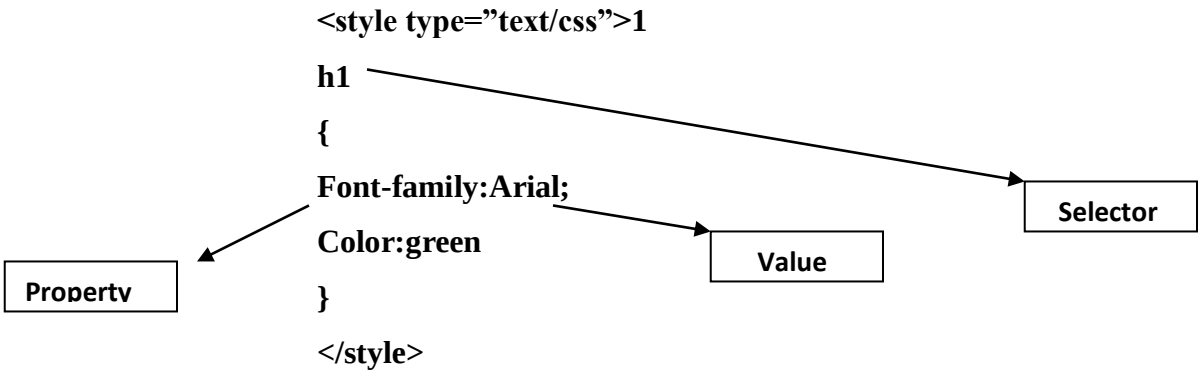
- A property, which is the property of the selected elements
- A value, which is a specification for the property

selector { property : value ; property : value ; …….}
----------Declaration----------

h1 { font – family : arial } → This CSS rule applies to all <h1> elements

The CSS is specified differently for each different level. For instance

- For Inline CSS the **style** appears inside the tag for defining the value.
  <p style="font-size:30pt;font-family:Script">
- For Embedded CSS the **style** specification appear as the content of a style element with in header of a document.




```
<style type="text/css">1
h1
{
Font-family:Arial;
Color:green
}
</style>
```

The **type** attribute tells the browser what it is reading is text which is effected by CSS.

In CSS we define each selector with help of property and value. The several categories of properties are as follows.

Font	**Text**	**Background**	**Border**	**Margin**	**Padding**
font	color	background	border-bottom-color	margin-left	padding-left
font-family	Direction	Background attachment	border-right-color	margin-right	padding-right
font-size	letter-spacing	background-color	border-top-color	margin-top	padding-top
font-size-adjust	text-align	background-image	border-left-color	margin-bottom	padding-bottom
font-stretch	text-decoration	background-position	border-bottom-style		
font-style	text-indent	background-repeat	border-top-style		
font-variant	text-shadow	**Dimensions**	border-right-style	**TABLE**	**LIST and MARKER**
font-weight	text-transform	line-height	border-left-style	border-collapse	list-style
	unicode-bidi	max-height	border-bottom-width	border-spacing	list-style-image
	white-space	max-width	border-top-width	caption-side	list-style-position
	word-spacing	min-height	border-left-width	empty-cells	list-style-type
		min-width	border-right-width	table-layout	marker-offset

Text Formatting

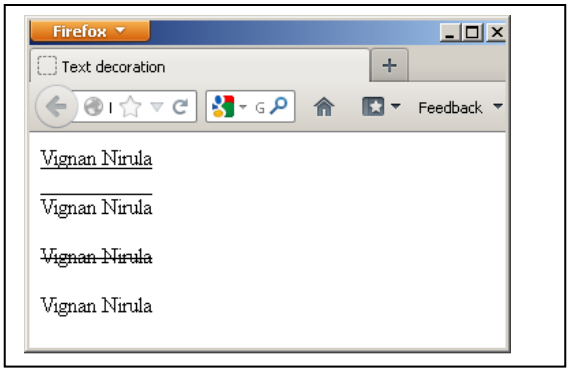
3.5.1 text-decoration

The text-decoration property allows you to specify the values shown in the table that follows.

Value	Purpose
underline	Adds a line under the content.
overline	Adds a line over the top of the content.
line-through	Like strikethrough text, with a line through the middle. In general, this should be used only to indicate text that is marked for deletion.
blink	Creates blinking text (which is generally frowned upon and considered annoying).

Example:

```
<html>
<head>
<title>Text decoration</title>
<style type="text/css">
p.underline {text-decoration:underline;}
p.overline {text-decoration:overline;}
p.linethrough {text-decoration:line-through;}
p.blink {text-decoration:blink;}
</style>
</head>
<body>
<p class="underline">Vignan Nirula</p>
<p class="overline">Vignan Nirula</p>
<p class="linethrough">Vignan Nirula</p>
<p class="blink">Vignan Nirula</p>
</body>
</html>
```



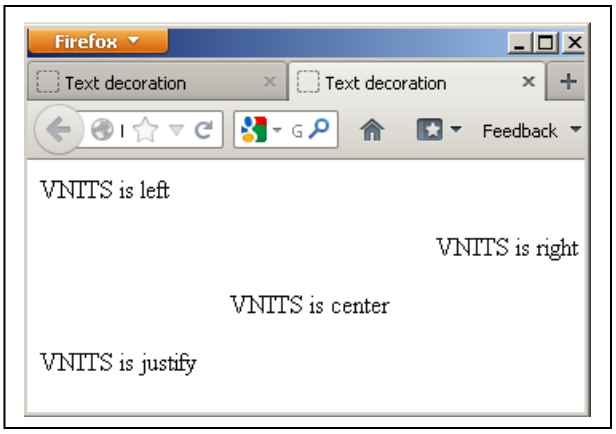
3.5.2 Text alignment

It aligns the text within its containing element or the browser window. This alignment can be done using following properties such as

Value	Purpose
left	Aligns the text with the left border of the containing element
right	Aligns the text with the right border of the containing element
center	Centers the content in the middle of the containing element
justify	Spreads the width across the whole width of the containing element

Example:

```
<html>
<head>
<title>Text alignment</title>
<style type="text/css">
p.leftalign {text-align:left;}
p.rightalign {text-align:right;}
p.center {text-align:center;}
p.justify {text-align:justify;}
</style>
</head>
<body>
<p class="leftalign">VNITS is left</p>
<p class="rightalign">VNITS is right</p>
<p class="center">VNITS is center</p>
<p class="justify">VNITS is justify</p>
</body>
</html>
```



The alignment of an image can be done using the property **float** and its value can be **left, right** and **none.** Using float property we can display the image at left or right wherever we wnat in the browser.

Selector Forms

There are various ways by which the selector can be defined.

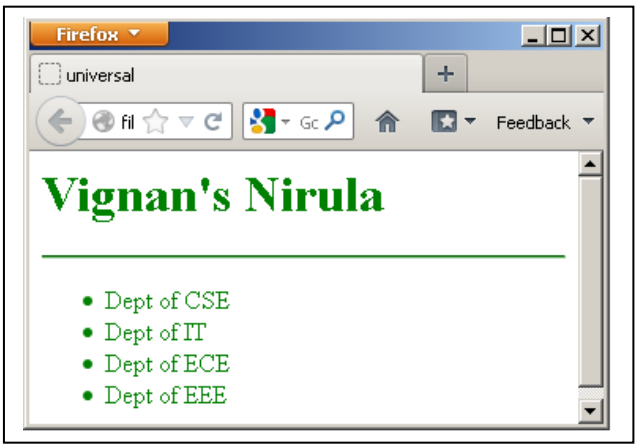
3.7.1 Universal Selector

The *universal selector* is an asterisk; it is like a wildcard and matches all element types in the document. *{}

If you want a rule to apply to all elements, you can use this selector. Sometimes it is used for default values, such as a font-family and font-size, that will apply to the whole of the document.

Example:

```
<html>
<head>
<title>universal</title>
<style type="text/css">
* {
color:green
}
</style>
</head>
```



```
<body>
<h1> Vignan's Nirula</h1>
<hr></hr>
<ul type="disc">
<li>Dept of CSE</li>
<li>Dept of IT</li>
<li>Dept of ECE</li>
<li>Dept of EEE</li>
</body>
</html>
```

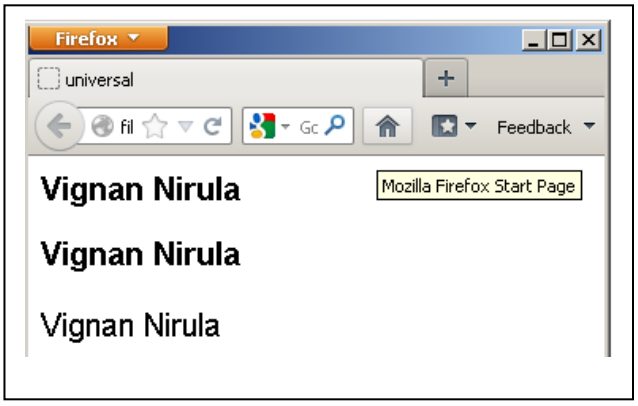
3.7.2 Type Selector

The type selector matches all of the elements specified in the comma-delimited list. It allows you to apply the same rules to several elements. For example, the following would match all h1, h2, and p elements.

```
h1, h2, p { font-family:Arial;
font-size:20px;
}
```

Example:

```
<html>
<head>
<title>type selector</title>
<style type="text/css">
h1, h2, p { font-family:Arial;
font-size:20px;
}
</style>
<body>
<h1>Vignan Nirula</h1>
<h2>Vignan Nirula</h2>
<p>Vignan Nirula</p></body></html>
```

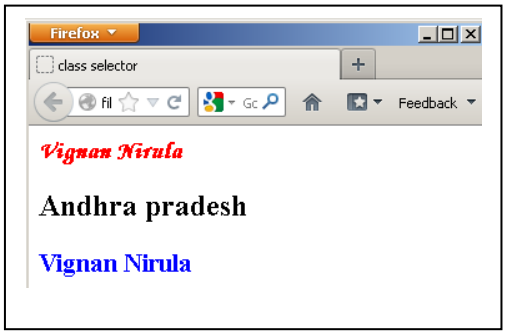


3.7.3 Class Selector

The class selector can assign different styles to the same element. These different styles appear on different occurrences of that element.

Example:

```
<html>
<head>
<title>class selector</title>
<style type="text/css">
h1.redtext
```



```
{
font-family:monotype corsiva;
color:red;
font-size:14pt;
}
h1.bluetext
{
font-family:times new roman;
color:blue;
font-size:16pt;
}
</style>
</head>
<body>
<h1 class="redtext">Vignan Nirula</h1>
<h2> Andhra pradesh</h2>
<h1 class="bluetext">Vignan Nirula</h1>
</body>
</html>
```

3.7.4 ID Selector

The ID selector is similar to the class selector but only the difference between the two is that class selector can be applied to more than one elements where as using the ID selector the style can be applied to one specific element.

The syntax is as follows.

#name_of_ID {property:value list;}

Example:

```
<html>
<head>
<title>ID selector</title>
<style type="text/css">
#top
{
font-family:monotype corsiva;
color:red;
font-size:14pt;
}
</style>
</head>
<div id="top">
```

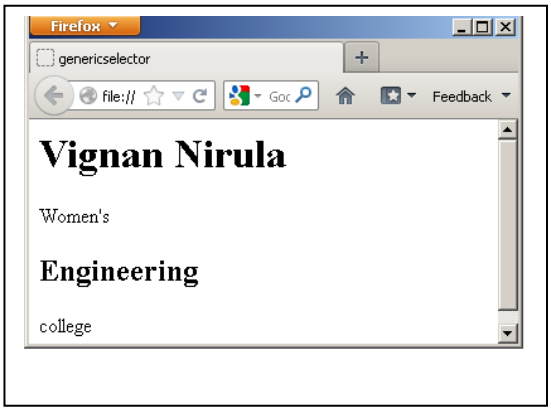
Vignan Nirula

 Women's
Engineering college
</body>
</html>

3.7.5 Generic selector

We define a class in generalized form. In the sense, that particular class can be applied to any tag.

Example:

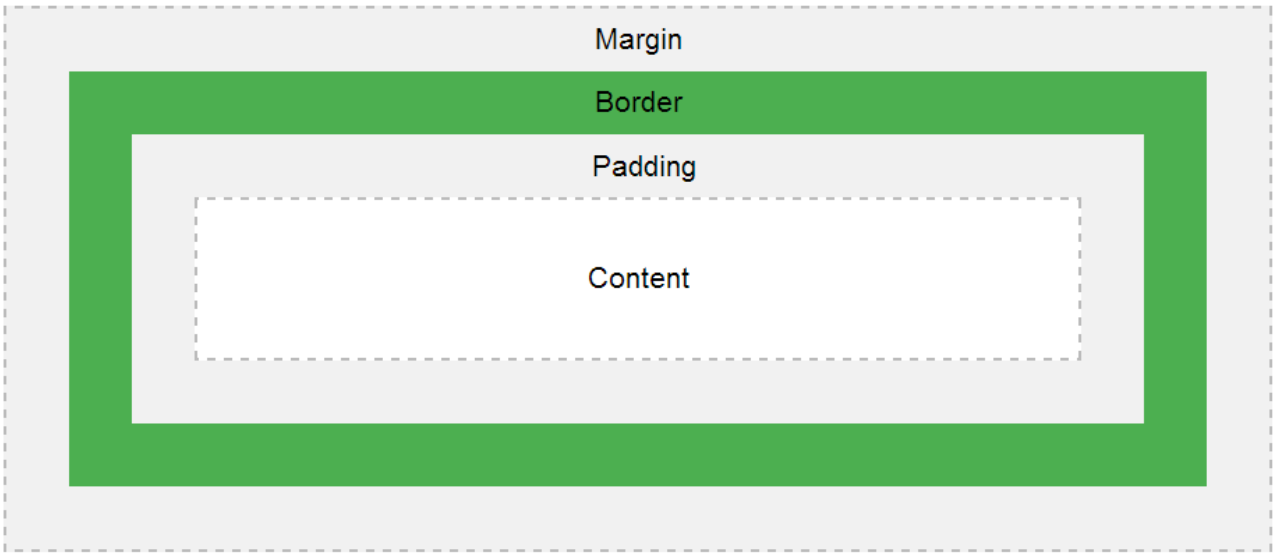



```
<html>
<head>
<title>genericselector</title>
<style type="text/css">
text
{
font-family:Arial;
color:black;
}
</style>
</head>
<body>
<h1 class="text">Vignan Nirula</h1>
<p> Women's</p>
<h2 class="text">Engineering</h2>
<p class="text">college</p>
</body>
</html>
```

The 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:



- **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

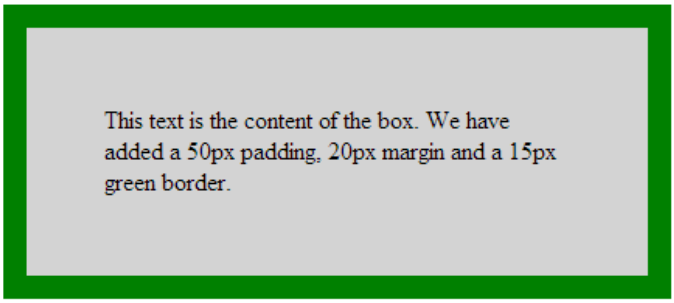
The box model allows us to add a border around elements, and to define space between elements.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
</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 content of the box. We have added a 50px padding, 20px margin and a 15px green border..</div>
</body>
</html>
```

Demonstrating the Box Model

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



Conflict Resolution

What is Specificity?

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out.

Think of specificity as a score/rank that determines which style declarations are ultimately applied to an element.

The universal selector (*) has low specificity, while ID selectors are highly specific!

Specificity Hierarchy

Every selector has its place in the specificity hierarchy. There are four categories which define the specificity level of a selector:

Inline styles - An inline style is attached directly to the element to be styled.

Example: <h1 style="color: #ffffff;">.

IDs - An ID is a unique identifier for the page elements, such as #navbar.

Classes, attributes and pseudo-classes - This category includes .classes, [attributes] and pseudo-classes such as :hover, :focus etc.

Elements and pseudo-elements - This category includes element names and pseudo-elements, such as h1, div, :before and :after.

How to Calculate Specificity?

Memorize how to calculate specificity!

Start at 0, add 1000 for style attribute, add 100 for each ID, add 10 for each attribute, class or pseudo-class, add 1 for each element name or pseudo-element.
Consider these three code fragments:
A: h1
B: #content h1
C: <h1 style="color: #ffffff">Heading</h1>
The specificity of A is 1 (one element)
The specificity of B is 101 (one ID reference and one element)
The specificity of C is 1000 (inline styling)
Since 1 < 101 < 1000, the third rule (C) has a greater level of specificity, and therefore will be applied.

Specificity Rules
Equal specificity: the latest rule counts - If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled, and therefore will be applied:

```
<html>
<head>
<style>
h1 {background-color: yellow;}
h1 {background-color: red;}
</style>
</head>
<body>
<h1>This is heading 1</h1>
</body>
</html>
```

ID selectors have a higher specificity than attribute selectors - Look at the following three code lines:

```
<html>
<head>
<style>
div#a {background-color: green;}
#a {background-color: yellow;}
div[id=a] {background-color: blue;}
</style>
</head>
<body>
<div id="a">This is a div</div>
</body>
</html>
```

the first rule is more specific than the other two, and will be applied.
Contextual selectors are more specific than a single element selector - The embedded style sheet is closer to the element to be styled. So in the following situation

From external CSS file:
#content h1 {background-color: red;}

In HTML file:
<style>
#content h1 {
  background-color: yellow;
}
</style>

A class selector beats any number of element selectors - a class selector such as .intro beats h1, p, div, etc:
<html>
<head>
<style>
.intro {background-color: yellow;}
h1 {background-color: red;}
</style>
</head>
<body>
<h1 class="intro">This is a heading</h1>
</body>
</html>

WEB TECHNOLOGIES

UNIT 2

Learning JavaScript

Introduction

JavaScript was developed by Brendan Eich, a member of Netscape in 1995. At that time its name was Live Script. After several years Sun Micro Systems joined the Netscape and they started developing Live script. And later on its name was changed to **JavaScript.**

JavaScript has three parts.

a) **Core:** It includes operators, expressions, statements and sub programs.
b) **Client-side:** It is a collection of objects using which one can have control over the browser. And user-browser interaction is possible.
c) **Server-side:** It is a collection of objects using which one can access the database on the server.

Differences between Java & JavaScript

Java	**Java script**
1. Java is a Programming language	1. Java script is a scripting language
2. Java is an object-oriented language	2. Java script is an object-based language
3. Objects in Java are Static. i.e., the number of data members and method are fixed at compile time.	3. In JavaScript the objects are dynamic. i.e., we can change the total data members and method of an object during execution.
4. Java is strongly typed language, and type checking is done at compile time.	4. Java script is loosely typed language. And type checking is done at run time.
5. Difficult to understand	5. Java script is comparatively easier to understand.

Uses of JavaScript (Applications)

- JavaScript can be used as an alternative to Java applets. The java applets need to be downloaded separately even though they are embedded in XHTML, however JavaScript need not be downloaded and used as embedded in XHTML document.
- JavaScript can get embedded in XHTML.
- JavaScript can be effectively used for interaction with the users. The JavaScript support forms such as buttons, textbox, menus…. Simple web applications such as **calculator, calendar** can be developed using JavaScript.
- Using DOM (Document object model) JavaScript can access and modify the properties of CSS and contents of XHTML. Hence static web document becomes dynamic.
- JavaScript can be used to **detect visitor's browsers** and can load the page accordingly.

- JavaScript can be used to create **cookies.**
- JavaScript is used to validate the **data** on the web page before submitting it to the server.

How to develop a JavaScript?

Script type

In web programming two types of scripts are popularly used.

- text/css
- text/javascript

We can ember the JavaScript directly or indirectly in XHTML document.

Hence we must write the tag

<script type="text/javascript">

</script>

If we want to embed the JavaScript indirectly, that means if the script is written in some another file and save it as **MyPage.js (**example). We want to embed that script in the XHTML document then we must write the script tag as follows.

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

</script>

Advantages of indirectly embedding of JavaScript

- Script can be hidden from the browser user.
- The layout and presentation of web document can be separated out from the user interaction through the JavaScript.

Disadvantages of indirectly embedding of JavaScript

- If small amount of JavaScript code has to be embedded in XHTML document then making a separate JavaScript file is meaningless.
- If the JavaScript code has to be embedded at several places in XHTML document then it is complicated to make the code for it from the XHTML document.

Identifiers

Identifiers are the names given to the variables. These variables hold the data value.

Following are the issues to handle JavaScript identifiers.

- Identifiers must begin with either letter or underscore or dollar sign. It is then followed by any number of letters, underscore, dollars or digits.
- There is no limit on length of identifiers.
- The letters in identifiers are case-sensitive. i.e., the identifier INDEX, Index, index, index are considered to be distinct.
- Programmer defined variable names must not have uppercase letters.

Reserved words

Reserved words are the special words associated with some meaning. Various reserve words that are used in JavaScript are as follows.

Break	continue	delete	for	in	return	var
With	case	default	else	function	switch	try
Void	catch	do	if	new	this	while

Comments

JavaScript supports following comments

- The // i.e. a single line comment can be used in JavaScript
- The /* and */ can be used as a multi-line comment
- The XHTML <!--> and <--> is also allowed in JavaScript

Semicolon

While writing the JavaScript the web programmer must give semicolon at the end of the statements.

Writing the First JavaScript

```
<html>
<head>
<title> My first JavaScript <title>
</head>
<body>
<center>
<script type=”text/javascript”>
/* this is first JavaScript */
document.write(“Welcome to first page of JavaScript”);
</script>
</center>
</body></html>
/* and */ comments supports in javascript only, not in XHTML.
```

Primitive Operations and Expressions

Variables are used to store data. To store information in a variable, you can give the variable a name and put an equal sign between it and the value you want it to have. Variables names are case - sensitive. Avoid giving two variables the same name within the same document as one might override the value of the other, creating an error.

Primitive types

JavaScript defines two entities **primitives** and **objects**. The primitives are for storing the values where as the object is for storing the reference (address) to actual values.

The primitive types used in JavaScript are **Number, String, Boolean, Undefined, Null**

The predefined objects used in JavaScript are **Number, String, Boolean**

These objects are called **wrapper objects.** These wrapper objects provide properties and methods can be used by primitive types.

- **Boolean:** values are **true** and **false**. These values can be compared with the variables.
- **Null:** Null value can be assigned by using reserved word **null**. It means no value.
- **Undefined:** If a variable is explicitly declared and not assigned any value to it then it is undefined value.

Literals

There are two types of literals used in JavaScript and those are numeric literals and string literals.

- The numeric literals are called **numbers.** These numbers can include integer values, floating point or double precision values.
  For example 10, 10.3, 10.0, 10E3, 10E6
- The String literals are the sequence of characters. It can be written in double quotes " " or in single quotes ' '
  For example "Vignan"

Variable declaration

A variable is a named memory location which can store some value. The following is general syntax for creating variables in java script.

var variblename;

We can declare the variable using reserved word **var.**

Scoping rules

Following are the variable scoping rules used in JavaScript

- **Script level scope:** If a variable is declared with a **var** and if it is declared outside any function then it has **script level scope.** It is called **global variable**.

- **Function level scope:** If a variable is declared with a **var** and if it is declared inside a function then it has **function level scope.** It is called **local variable**.
- **Auto-declaration:** If a variable is declared without a **var** declaration statement, it will be automatically declared with script level scope, and it becomes global variable.
- **Collision:** If a variable is explicitly defined in a function has the same name as a variable defined outside the function, then the variable outside the function cannot accessible with in this function.

Operators

Operators are the symbol which operates on value or a variable. For example: + is a operator to perform addition. C programming language has wide range of operators to perform various operations.

Operators
Arithmetic Operators
Increment and Decrement Operators
Assignment Operators
Relational Operators
Logical Operators
Conditional Operators
Bitwise Operators
Special Operators

Arithmetic Operators

Arithmetic operators are user to perform arithmetic operations on two variables.

Operator	Meaning Of Operator
+	addition or unary plus
-	subtraction or unary minus
*	Multiplication
/	Division
%	remainder after division(modulo division)

Relational Operator

Relational operators checks relationship between two operands. If the relation is true, it returns value 1 and if the relation is false, it returns value 0. For example:

```
a>b
```

Here, > is a relational operator. If a is greater than b, a>b returns 1 if not then, it returns 0.

Relational operators are used in decision making and loops

Operator	Meaning Of Operator	Example
==	Equal to	5==3 returns false (0)
>	Greater than	5>3 returns true (1)
<	Less than	5<3 returns false (0)
!=	Not equal to	5!=3 returns true(1)
>=	Greater than or equal to	5>=3 returns true (1)
<=	Less than or equal to	5<=3 return false (0)

Logical Operators

Logical operators are used to combine expressions containing relation operators. In C, there are 3 logical operators:

Operator	Meaning Of Operator	Example
&&	Logial AND	If c=5 and d=2 then,((c==5) && (d>5)) returns false.
||	Logical OR	If c=5 and d=2 then, ((c==5) || (d>5)) returns true.
!	Logical NOT	If c=5 then, !(c==5) returns false.

Explanation

For expression, ((c==5) && (d>5)) to be true, both c==5 and d>5 should be true but, (d>5) is false in the given example. So, the expression is false. For expression ((c==5) || (d>5)) to be true, either the expression should be true. Since, (c==5) is true. So, the expression is true. Since, expression (c==5) is true, !(c==5) is false.

Assignment Operators

The most common assignment operator is =. This operator assigns the value in right side to the left side. For example:

```
var=5  //5 is assigned to var
a=c;   //value of c is assigned to a
5=c;   // Error! 5 is a constant.
```

Increment and decrement operators

++ and -- are called increment and decrement operators respectively. Both of these operators are unary operators, i.e, used on single operand. ++ adds 1 to operand and -- subtracts 1 to operand respectively. For example:

Let a=5 and b=10

```
a++; //a becomes 6
a--; //a becomes 5
++a; //a becomes 6
--a; //a becomes 5
```

Difference between ++ and -- operator as postfix and prefix When i++ is used as prefix(like: ++var), ++var will increment the value of var and then return it but, if ++ is used as postfix(like: var++), operator will return the value of operand first and then only increment it.

Bitwise Operators

A bitwise operator works on each bit of data. Bitwise operators are used in bit level programming.

Operators	Meaning Of Operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

Conditional operators (?:)

Conditional operators are used in decision making in C programming, i.e, executes different statements according to test condition whether it is either true or false.

Syntax :

conditional_expression?expression1:expression2

If the test condition is true, expression1 is returned and if false expression2 is returned.

Ex: c=(a>b)?a:b;

Special Operators

Comma Operator

Comma operators are used to link related expressions together. For example:

int a,c=5,d;

The sizeof operator

It is a unary operator which is used in finding the size of data type, constant, arrays, structure etc.

String concatenation operator

Two strings can be concatenated using the + operator. A variable can be concatenated with the string using + operator also.

Ex:

```
<html>
<head>
<title>String concatenation</title>
</head>
<body>
<script type=”text/javascript”>
var str;
str=”vignan”;
document.write(“<h3>”+str+”Engineering college”+”</h3>”);
</script>
</body>
</html>
```

typeof operator

typeof operator returns the primitive type of the variable. If the variable is of object type or null then it returns object. If a variable is declared and is not used then its type is undefined.

Ex:

```
<html>
<head>
<title>typeof operator</title>
</head>
<body>
<script type=”text/javascript”>
var str=”computer”;
var num=100;
var myobj;
document.write(“String=”+typeof(str)+”<br/>”);
document.write(“Number=”+typeof(str)+”<br/>”);
document.write(“Object=”+typeof(str)+”<br/>”);
</script>
</body>
</html>
```

Implicit type conversion

JavaScript supports implicit type conversion. The lower data type is automatically converted to higher data type. Such a data type is known as implicit type conversion (Coercion).

Ex:

1."Survey_number"+10

The result of above operation will get a string

"Survey_number 10"

2. 5*"12"

Second operand which is a String will automatically converted to number.

The result is 60.

3. 5*"Hello"

Second operand which is a String cannot be converted to number.

The result is NaN –Not a number.

Ex:

```
<html>
<head>
<title>Implicit typeconversion</title>
</head>
<body>
<script type="text/javascript">
var a,b;
a=5*"12";
b=5*"hello"
document.write("a+"<br/>");
document.write("b");
</script>
</body>
</html>
```

Explicit type conversion

One can forcefully perform the type conversion in JavaScript. The numeric value can be converted to string type using **tostring** method.

Ex:

```
var  mynum=2;
var mystr;
mystr=mynum.toString();
```

Now, value of mynum="2".

In JavaScript the explicit type conversion can be done as
Boolean(val)→converts val to boolean type
Number(val)→converts val to number type
String(val)→converts val to string type.

To separate the numeric data from an alphanumeric data we use two methods.
ParseInt()→separate the int value.
parseFloat()→separate the float value.

Screen Output & Keyboard Input

Document.write property

For displaying the message on the web browser the basic method being used is **document.write**. To print the desired message on the web browser we write the message in double quotes.

Ex: document.write("Welcome to Vignan!");

Popup boxes

There are three types of popup boxes by which the user can interact with the browser.

1.alert box: Some message will be displayed

2.confirm box: The message about confirmation will be displayed. Hence it should have two buttons. **Ok** and **cancel.**

3.prompt box: which displays a text window in which the user can enter something. Hence it should have two buttons. **Ok** and **cancel.**

Ex:

```
<html>
<head>
<title>Input/output</title>
</head>
<body>
<script type="text/javascript">
if(confirm("Do you agree?"))
 alert("You have agreed");
else
 input=prompt("Enter some text here");
alert("Hi"+input);
</script>
</body>
</html>
```

Control Statements

Control Structure is the essential part of any programming language which is required to control logic of program.

Conditional Statements

Conditional statements allow you to take different actions depending upon different statements. There are three types of conditional statements you will learn about here:

- if statements, which are used when you want the script to execute if a condition is true
- if...else statements, which are used when you want to execute one set of code if a condition is true and another if it is false
- switch statements, which are used when you want to select one block of code from many depending on a situation

if Statements

if statements allow code to be executed when the condition specified is met; if the condition is true then the code in the curly braces is executed. Here is the syntax for an if statement:

```
if (condition)
{
code to be executed if condition is true
}
```

if . . . else Statements

When you have two possible situations and you want to react differently for each, you can use an if...else statement. This means: " If the conditions specified are met, run the first block of code; otherwiserun the second block. " The syntax is as follows:

```
if (condition)
{
code to be executed if condition is true
}
else
{
code to be executed if condition is false
}
```

Ex:

```
<html>
<head>
<title>if/else</title>
</head>
<body>
<script type="text/javascript">
var a,b,c;
a=10;b=20;c=30;
if(a>b)
{
if(a>c)
```

```
document.write("a is largest number"+a);
else
document.write("c is largest number"+c);
}
else if(b>c)
document.write("b is largest number"+b);
else
document.write("c is largest number"+c);
</script>
</body>
</html>
```

switch Statements

A switch statement allows you to deal with several possible results of a condition. You have a single expression, which is usually a variable. This is evaluated immediately. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code will execute.

Syntax:

```
switch (expression)
{
case option1:
code to be executed if expression is what is written in option1
break;
case option2:
code to be executed if expression is what is written in option2
break;
case option3:
code to be executed if expression is what is written in option3
break;
default:
code to be executed if expression is different from option1, option2,
and option3
}
```

You use the break to prevent code from running into the next case automatically.

Ex:

```
<html>
<head>
<title>switch</title>
</head>
<body>
<script type="text/javascript">
```

```
var d=new Date();
ch=d.getMonth();
switch(ch)
{
case 0:document.write(“January”);
        break;
case 1:document.write(“february”);
        break;
case 2:document.write(“march”);
        break;
case 3:document.write(“april”);
        break;
case 4:document.write(“may”);
        break;
case 5:document.write(“june”);
        break;
case 6:document.write(“July”);
        break;
case 7:document.write(“august”);
        break;
case 8:document.write(“september”);
        break;
case 9:document.write(“october”);
        break;
case 10:document.write(“november”);
        break;
case 11:document.write(“december”);
        break;
default: document.write(“Invalid choice”);
}
</script>
</body>
</html>
```

Loop Control statements

Loop statements are used to execute the same block of code a specified number of times.

- A while loop runs the same block of code while or until a condition is true.
- A do while loop runs once before the condition is checked. If the condition is true, it will continue to run until the condition is false. (The difference between a do and a do while loop is that do while runs once whether or not the condition is met.)
- A for loop runs the same block of code a specified number of times

While

In a while loop, a code block is executed if a condition is true and for as long as that condition remains true. The syntax is as follows:

```
while (condition)
{
code to be executed
}
```

Ex:

```
<html>
<head>
<title>while</title>
</head>
<body>
<script type=”text/javascript”>
var i=1;
while(i<=10)
{
 document.write(“Number”+i+”It’s square”+(i*i)+”<br/>”)
i++;
}
</script>
</body>
</html>
```

do . . . while

A do ... while loop executes a block of code once and then checks a condition. For as long as the condition is true, it continues to loop. So, whatever the condition, the loop runs at least once.

Here is the syntax:

```
do
{
code to be executed
}
while (condition);
```

Ex:

```
<html>
<head>
<title>do/while</title>
</head>
<body>
<script type=”text/javascript”>
var i=1;
do
{
 document.write(“Number”+i+”It’s square”+(i*i)+”<br/>”)
i++;
}
while(i<=10);
</script>
</body>
</html>
```

For

The for statement executes a block of code a specified number of times. You use it when you want to specify how many times you want the code to be executed

Here is the syntax.

```
for (initialization; test condition; increment/decrement)
{
code to be executed
}
```

Ex:

```
<html>
<head>
<title>for</title>
</head>
<body>
<script type=”text/javascript”>
var i;
for(i=1;i<=10;i++)
{
 document.write(“Number”+i+”It’s square”+(i*i)+”<br/>”)
}
</script>
</body>
</html>
```

Break statement

The break statement is used to break the loop from running many times.

Ex:

```
<html>
<head>
<title>break</title>
</head>
<body>
<script type=”text/javascript”>
var i;
for(i=10;i>=1;i--)
{
if(i==5)
break;
}
 document.write(“My lucky Number is:”+i”)
</script>
</body>
</html>
```

Continue statement

The continue statement is used in a loop in order to continue (skip).

```
<html>
<head>
<title>continue</title>
</head>
<body>
<script type=”text/javascript”>
var i;
for(i=10;i>=1;i--)
{
if(i==5)
{
x=i;
continue;
}
document.write(i+”<br/>”);
}
 document.write(“My missed Number is:”+x”)
</script>
</body>
</html>
```

Objects

- In JavaScript object is a collection of **properties.** These properties are nothing but the members of the classes from Java or C++. For example, in javascript the object Date () is used which happens to the member of the classes in Java.
- The property can be two types either a data type or a function type. The data properties are sometimes called as **properties** and method properties are called as **methods** or **functions.**
- The data properties are further classified into two types-primitive values and reference to other objects.
- Primitives are simplest data types for example number, string, Boolean are some primitives used to define type the simple data.
- The objects in the JavaScript are accessed using the variables. These variables acts like a reference to the object. The property of the object can be accessed with the help of object name using the dot operator. For example if there is an object named *person* then its property *salary* can be accessed as *person*.salary
- **Object** is a root object in JavaScript. All other objects are formed from this **object** using **prototype inheritance**. All other objects inherit the methods of this **object**. In JavaScript the object appears internally and externally. It is there in the form of property-value pair. The properties are the names and values are data values or functions.

String Object:

The string object allows you to deal with strings of text. Before you can use a built - in object, you need to create an instance of that object. You create an instance of the string object by assigning it to a variable like so:

myString = new String('Here is some bold text")

The string object now contains the words " Here is some bold text " and this is stored in a variable called myString .

Example:

```
<html>
<head>
<title>String object</title>
</head>
<body>
<script type="text/javascript">
var s1="Hello";
var s2="Vignan";
document.write("First string is:"+s1+"<br/>");
document.write("Second string is:"+s2+"<br/>");
document.write("Concatenation:"+s1.concat(s2)+"<br/>");
```

document.write("Length of string s1 is:"+s1.length+"
");
document.write("Length of string s2 is:"+s2.length+"
");
document.write("character at 3rd position of string s1 is:"+s1.charAt(3)+"
");
document.write("string s1 in lowercase is:"+s1.toLowerCase+"
");
</script>
</body></html>

Date Object:

The date object helps you work with dates and times. You create a new date object using the date constructor like so:
var d=new Date();
You can create a date object set to a specific date or time, in which case you need to pass it one of four parameters:

- milliseconds : This value should be the number of milliseconds since 01/01/1970.
- dateString : Can be any date in a format recognized by the parse() method.
- yr_num , mo_num , day_num : Represents year, month, and day.
- yr_num , mo_num , day_num , hr_num , min_num , seconds_num , ms_num : Represents the years, days, hours, minutes, seconds, and milliseconds.
- This Date and Time value is based on computer's local time (system's time) or it can be based on GMT(Greenwich Mean Time) which is also known as UTC(Universal Coordinated Time). This is basically a world time standard.

The following table shows some commonly used methods of the date object.
Many of the methods in the table that follows were then added offering support for the universal (UTC) time, which takes the format Day Month Date, hh,mm,ss UTC Year .

Example:

```
<html>
<head>
<title>Date object</title>
</head>
<body>
<script type="text/javascript">
var mydate=new Date();
document.write("The Date is:"+mydate.toString()+"<br/>");
document.write("Today's Date is:"+mydate.getDate()+"<br/>");
document.write("Current year is:"+mydate.getFullYear()+"<br/>");
document.write("Minutes is:"+mydate.getMinutes()+"<br/>");
document.write("Seconds is:"+mydate.getSeconds()+"<br/>");
</script>
</body>
</html>
```

Math Object:

The math object helps in working with numbers. It has properties for mathematical constants and methods representing mathematical functions such as the Tangent and Sine functions. The following example rounds pi to the nearest whole number (integer) and writes it to the screen.

```
numberPI = Math.PI
numberPI = Math.round(numberPI)
document.write (numberPI)
```

Example:

```
<html>
<head>
<title>Math object</title>
</head>
<body>
<script type=”text/javascript”>
var number=100;
document.write(“Square root of number is:”+Math.sqrt(num));
</script>
</body>
</html>
```

Object creation and modification:

The web designer can create an object and can set its properties as per his requirements. The object can be created using **new** expression. Initially the object with no properties can be set using following statements.

```
Myobj=new Object();
```

Then by using dot operator we can set the properties for that object. The object can then be modified by assigning values to that objet.

Example:

```
<html>
<head>
<title>Object creation</title>
</head>
<body>
<script type=”text/javascript”>
var student;
student=new Object();
student.id=10;
student.name=”Vignan”;
document.write(“The ID is:”+student.id);
document.write(“The Name is:”+student.name);
</script>
</body></html>
```

Functions

A function is made up of related code that performs a particular task. For example, a function could be written to calculate area given width and height.

There are three parts to creating or defining a function:

- Define a name for it.
- Indicate any values that might be required; these are known as arguments.
- Add statements to the body of the function.

It is standard to define function in the **head** section and call that function from the **body** section.
The keyword **function** is used to define the function.

function name_of_function(arg_1,arg_2,……,arg_n)
{

----
statements
}

Example:

```
<html>
<head>
<title>Defining a function</title>
<script type=”text/javascript”>
function name()
{
document.write(“welcome to vignan”);
}
</script>
</head>
<body>
<script type=”text/javascript”>
document.write(“Department of IT”);
name();
</body>
</html>
```

Returning values from function

We can return values from the function using a keyword **return.**

```
<html>
<head>
<title>Defining a function</title>
<script type=”text/javascript”>
function name()
{
str=”vignan”;
return str;
}
```

```
</script>
</head>
<body>
<script type=”text/javascript”>
document.write(“Department of IT”);
name();
</body>
</html>
```

Passing parameters to Function

We can pass some parameters to function. In the following program, we have passed two arguments to the function and returning some value from the function.

Example:

```
<html>
<head>
<title>Defining a function</title>
<script type=”text/javascript”>
function add(x,y)
{
return x+y;
}
</script>
</head>
<body>
<script type=”text/javascript”>
var a,b;
a=10;b=20;
c=add(a,b);
document.write(“Addition of two numbers:”+c);
</script>
</body>
</html>
```

Arrays

Arrays is a collection of similar type of elements that can share a common name. Any element in an array is referred by **array name** followed by “[“ followed by **position** of the element followed by “]”. The position of the element in array is called array **index** or **subscript.**

In JavaScript the array can be created by using **Array** object.

Creating an array is as follows.

```
var ar=new Array(10);
```

Using **new** operator we can allocate the memory for the arrays dynamically.

var denotes the name of the array. The size of the array is mentioned in brackets.

Example:

```
<html>
<head>
<title>Defining a function</title>
```

```
</head>
<body>
<script type=”text/javascript”>
var a=new Array(5);
for(i=0;i<=5;i++)
{
a[i]=i;
document.write(a[i]+”<br/>”);
}
</script>
</body>
</html>
```

Passing An Array to Function

We can pass an entire array as a parameter to the function. When an array is passed to the function the array name is simply passed.

Example:

```
<html>
<head>
<title>passing  array to  a function</title>
<script type=”text/javascript”>
function display(a)
{
document.write(“The array elements are:”);
i=0;
for(i in a)
{
document.write(a[i]+”<br/>”);
i++;
}
}
</script>
</head>
<body>
<script type=”text/javascript”>
var arr=new Array(10);
for(i=0;i<=9;i++)
{
arr[i]=i;
}
display(arr);
</script>
</body>
</html>
```

JavaScript Constructors

Constructors are like regular functions, but we use them with the new keyword.

There are two types of constructors: built-in constructors such as Array and Object, which are available automatically in the execution environment at runtime; and custom constructors, which define properties and methods for your own type of object.

A constructor is useful when you want to create multiple similar objects with the same properties and methods. It's a convention to capitalize the name of constructors to distinguish them from regular functions. Consider the following code:

```
function Book()
{
  // unfinished code
}
var myBook = new Book();
```

The last line of the code creates an instance of Book and assigns it to a variable. Although the Book constructor doesn't do anything, myBook is still an instance of it. As you can see, there is no difference between this function and regular functions except that it's called with the newkeyword and the function name is capitalized.

Determining the type of an instance

To find out whether an object is an instance of another one, we use the instanceof operator:

```
myBook instanceof Book    // true
myBook instanceof String  // false
```

Note that if the right side of the instanceof operator isn't a function, it will throw an error:

```
myBook instanceof {};
// TypeError: invalid 'instanceof' operand ({})
```

Another way to find the type of an instance is to use the constructor property. Consider the following code fragment:

```
myBook.constructor === Book;   // true
```

The constructor property of myBook points to Book, so the strict equality operator returns true.

All objects inherit a constructor property from their prototype which point to the constructor function that has created the object:

```
var s = new String("text");
s.constructor === String;      // true
"text".constructor === String; // true
var o = new Object();
o.constructor === Object;      // true
var o = {};
o.constructor === Object;      // true
var a = new Array();
a.constructor === Array;       // true
```

The Object.defineProperty() method can be used inside a constructor to help perform all necessary property setup. Consider the following constructor:

```
function Book(name) {
```

```
  Object.defineProperty(this, "name", {
     get: function() {
       return "Book: " + name;
     },
     set: function(newName) {
       name = newName;
     },
     configurable: false
   });
}

var myBook = new Book("Single Page Web Applications");
console.log(myBook.name);    // Book: Single Page Web Applications
// we cannot delete the name property because "configurable" is set to false
delete myBook.name;
console.log(myBook.name);    // Book: Single Page Web Applications
// but we can change the value of the name property
myBook.name = "Testable JavaScript";
console.log(myBook.name);    // Book: Testable JavaScript
```

The JavaScript language has nine built-in
constructors: Object(), Array(), String(), Number(), Boolean(), Date(), Function(),Error() and RegE xp(). When creating values, we are free to use either object literals or constructors.

```
var obj = new Object(5);
obj.toFixed(2);     // 5.00
// we can achieve the same result using literals
var num = 5;
num.toFixed(2);     // 5.00
// a string object
// strings have a slice() method
var obj = new String("text");
obj.slice(0,2);     // "te"
```

Pattern Matching Using Regular Expressions

What Is a Regular Expression?

A regular expression is a sequence of characters that forms a **search pattern**.

When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of **text search** and **text replace** operations.

Syntax: /*pattern*/*modifiers*;

Ex:

```
var patt = /w3schools/i;
```

/w3schools/i is a regular expression.
w3schools is a pattern (to be used in a search).
i is a modifier (modifies the search to be case-insensitive).

Using String Methods
In JavaScript, regular expressions are often used with the two **string methods**: search() and replace().
The search() method uses an expression to search for a match, and returns the position of the match.
The replace() method returns a modified string where the pattern is replaced.

Using String search() With a String
The search() method searches a string for a specified value and returns the position of the match:

```
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>Search a string for "W3Schools", and display the position of the match:</p>
<p id="demo"></p>
<script>
var str = "Visit W3Schools!";
var n = str.search("W3Schools");
document.getElementById("demo").innerHTML = n;
</script>
</body>
</html>
```

Using String search() With a Regular Expression
Use a regular expression to do a case-insensitive search for "w3schools" in a string:

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>Search a string for "w3Schools", and display the position of the match:</p>
<p id="demo"></p>
<script>
var str = "Visit W3Schools!";
var n = str.search(/w3Schools/i);
document.getElementById("demo").innerHTML = n;
</script>
</body>
</html>
```

Using String replace() With a String
The replace() method replaces a specified value with another value in a string:

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

```
<h2>JavaScript String Methods</h2>
<p>Replace "Microsoft" with "W3Schools" in the paragraph below:</p>
<button onclick="myFunction()">Try it</button>
<p id="demo">Please visit Microsoft!</p>
<script>
function myFunction() {
  var str = document.getElementById("demo").innerHTML;
  var txt = str.replace("Microsoft","W3Schools");
  document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

Use String replace() With a Regular Expression

Use a case insensitive regular expression to replace Microsoft with W3Schools in a string:

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>Replace "microsoft" with "W3Schools" in the paragraph below:</p>
<button onclick="myFunction()">Try it</button>
<p id="demo">Please visit Microsoft and Microsoft!</p>
<script>
function myFunction() {
  var str = document.getElementById("demo").innerHTML;
  var txt = str.replace(/microsoft/i,"W3Schools");
  document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

DHTML: Positioning Moving and Changing Elements

DHTML is a collection of features that together, enable your web page to be dynamic. "Dynamic" is defined as the ability of the browser to alter a web page's look and style after the document has loaded. "DHTML is the combination of several built-in browser features in fourth generation browsers that enable a web page to be more dynamic".

Suppose there is a picture of monkey on your elements on the web page. If you simply display monkey's picture and display *"monkey likes jumping"*, then it is an old style of web designing. However, if you click on the picture of monkey and then if your web page could show that monkey is jumping saying that *"monkey likes jumping"* – then it adds up the value in your design. This effect on the web page is known as **Dynamic effect.** With **DHTML** (**D**ynamic **H**yper

Text **M**arkup **L**anguage) a web developer can control how to display and position HTML elements in a browser window.

Differences between HTML and DHTML

HTML (HyperText Markup Language) is the actual markup language web pages are displayed in. It creates static content on web page. .

DHTML stands for Dynamic HTML and it's using HTML, CSS and JavaScript together to create web pages that are not static displays only, but the content is actually dynamic.

HTML does not allow altering the text and graphics on the web page unless the web page gets changed.

DHTML allows altering the text and graphics of the web page and for that matter you need not have to change the entire web page.

Creation of HTML is simplest and less interactive.

Creation of DHTML is complex but more interactive.

HTML

1. It is referred as a static HTML and static in nature.
2. A plain page without any styles and Scripts called as HTML.
3. HTML sites will be slow upon client-side technologies.

DHTML

1.It is referred as a dynamic HTML and dynamic in nature.
2. A page with HTML, CSS, DOM and Scripts called as DHTML.
3. DHTML sites will be fast enough upon client-side technologies.

Dynamic Style

Dynamic styles are a key feature of DHTML. The DHTML's object model allows us to change the style of the object using **style** attribute. This style is basically known as **dynamic style** of object.

For example, to highlight the text in a heading when the user moves the mouse pointer over it, you can use the **style** object to enlarge the font and change its color, as shown in the following simple example.

Example:

```
<html>
<head>
<title>Dynamic Styles</title>
<script language="JavaScript">
function doChanges(e)
{
   e.style.color = "green";
   e.style.fontSize = "20px";
```

```
}
</script>
</head>
<body>
<h3 onmouseover="doChanges(this)" style="color:black;font-size:18px">Welcome to Dynamic
HTML!</h3>
<p>You can do the most amazing things with the least bit of effort.</p>
</body>
</html>
```

Changing Font

Using the **style.font** the font style can be changed. For changing the font style some event must be occurred.

Example:

```
<html>
<head>
<title>changing font</title>
</head>
<body>
<p>
<a onmouseover="this.style.color='blue';this.style.font='20px Arial';"
   onmouseout="this.style.color='black';this.style.font='12px normal';">Before you speak
</a>listen
<br/>
<a onmouseover="this.style.color='blue';this.style.font='20px Arial';"
   onmouseout="this.style.color='black';this.style.font='12px normal';">Before you write
</a>think
</p>
<br/><br/>
<p>
<em>By Vignan Nirula </em>
</p>
</body>
</html>
```

Reacting to a Mouse click

When we click on mouse button it goes down and then come back. These activities can be caught using onmousedown and onmouseup events respectively.

In the following example we will display one image when user presses mouse button and another image gets displayed when user releases the mouse button.

Example:

```
<html>
<head>
<title>Reaction to a mouse click</title>
<script language="JavaScript">
function abc()
{
img1.src=”vignan.jpg”;
}
function xyz()
{
img1.src=”nirula.jpg”;
}
</script>
</head>
<body>
<h1>Two wonders of the world</h1>
<img id=”img1” onmousedown=”abc()” onmouseup=”xyz()” src=”nirula.jpg” width=”200” height=”200”/>
<p>click this image to change it</p>
</body>
</html>
```

WEB TECHNOLOGIES

UNIT 3

Working with XML:

Introduction to XML:

- XML stands for Extensible Markup Language.
- XML is a markup language much like HTML.
- XML was designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is designed to be self-descriptive.
- XML is a W3C Recommendation.
- XML was designed to transport and store data where HTML was designed to display data

Rules for writing XML Document:

- In XML the basic entity is an element. The elements are used for defining the tags. The elements typically consists of opening and closing tag.
- XML is a case sensitive language. For example
  < **n**ame > Naresh < /**N**ame > is not allowed
- In XML each start tag must have matching end tag.
- The XML elements in XML must be properly nested. For example

```
<name>
    <firstname></firstname>
    <lastname></lastname>
</name>
```

- A space or tab character is not allowed in the element name or in attribute name.
- To insert **"<" or ">" or " ' " or "&" or " "" "** characters with in text simply use **"&lt", "&gt", "&apos", "&amp", "&quot"** respectively.
- The syntax for writing comments in XML is similar to HTML. For example
  <!--This is a comment line -->

Advantages of XML:

1. XML document is human readable and we can edit in any XML document in any text editors.
2. The XML document is language neutral.
3. Every XML document has a tree structure, hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.
5. XML is useful in exchanging data between the applications.
6. XML document is used to extract data from data base.

Goals of XML:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML
4. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

5. XML documents should be human-legible and reasonably clear.
6. The XML design should be prepared quickly.
7. The design of XML shall be formal and concise.
8. XML documents shall be easy to create.
9. Terseness in XML markup is of minimal importance.
10. It shall be easy to write programs which process XML documents.

Namespace in XML: The process of creating two elements with same name is known as namespace in XML. For example

```
<vehicles>
        <car>
                <price>200000</price>
                <color>red</color>
        </car>
        <car>
                <price>300000</price>
                <color>green</color>
        </car>
</vehicle>
```

Here <vehicle> is a root element. Root element is an element which is parent of all other elements in the XML document. The other elements are called the container elements. The container elements may contain one or more elements, called as child elements. In above xml document <car> Is a container element, <price> and <color> are child elements.

XML CSS: We can also apply styles to XML document using CSS. This can be done as follows.

library.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="library.css"?>
<catalog>
  <book>
    <title>XML Bible</title>
    <author>Winston</author>
  </book>
  <book>
    <title>AI</title>
    <author>S.Russel</author>
  </book>  </catalog>
```

library.css

```
catalog
{
        font-family:arial;
        color:red;
        font-size:16pt;
}
book,author
{
        font-family:times new roman;
        color:black;
        font-size:14pt;
}
```

Advantages of XML:

1. XML schemas are more specific
2. XML schema provides the support for data types.
3. XML schema supports namespace
4. XML schema has large number of built-in data types.

Disadvantages of XML:

1. XML schema is complex to design and hard to learn
2. XML document cannot be present if the corresponding schema is not found
3. Maintaining the schema for large and complex operations causes less execution speed

Document Type Definition: The Document type definition is used to define the basic building blocks of any XML document. Using DTD we can specify the various elements types, attributes. DTD specifies set of rules for structuring data in XML file. Various building blocks of XML are as follows:

1. Elements: The basic entity is element. The elements are used for defining tags. The elements consist of opening tag and closing tag.
2. Attribute: The attributes are used to specify the values of the element. These are specified with in double quotes.
3. CDDATA: CDDATA stands for character data.
4. PCDATA: PCDATA stands for Parsed Character Data.

Types of DTD: DTD's are classified into two types. They are

1. **Internal DTD**
2. **External DTD**

1. **Internal DTD:** Using an internal DTD, the code is placed between the DOCTYPE tags. For example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,marks)>
<!ELENENT name (#PCDATA)>
<!ELENENT address (#PCDATA)>
<!ELENENT marks (#PCDATA)>
]>
<student>
   <name>naresh</name>
   <address>gutur</address>
   <marks>68</marks>
</student>
```

2. **External DTD:** In this type, create an external DTD file and specify the file name in the corresponding XML document. For example

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student SYSTEM "student.dtd">
<student>
   <name>naresh</name>
   <address>Guntur</address>
   <marks>68</marks>
</student>
```

student.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT student (name,address,marks) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT address (#PCDATA) >
<!ELEMENT marks (#PCDATA) >
```

DTD and CSS: We can also include both DTD and CSS files to XML document. This can be done as follows:

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="student.css"?>
<!DOCTYPE student SYSTEM "student.dtd">
<student>
   <name>naresh</name>
```

student.css

```
name
{
   font-family:arial;
   color:black;
   font-size:12px;
}
address
{
   font-family:cursive;
   color:blue;
   font-size:14px;
```

```
  <address>Guntur</address>
  <marks>68</marks>
</student>
```

Merits of DTD:

1. DTD's are used to define the structural components of XML document
2. DTD's are simple and compact
3. DTD's can be defined inline

Demerits of DTD:

1. DTD's are not suitable for complex documents
2. Various IDE's cannot support DTD's
3. DTD's cannot define the type of data
4. Some XML processors cannot understood DTD's
5. DTD's does not support namespace concept.

Differences between XML and DTD:

XML	**DTD**
1. XML schema's are suitable for large applications	1. DTD's are suitable for small applications
2. XML schema's support for defining the type of data.	2. DTD's does not support
3. XML schema's support namespace	3. DTD's does not support namespace.
4. XML schema has large number of built in and derived data types.	4. DTD's has limited number of data types

XML Schema: The XML schemas are used to represent the structure of XML document. The purpose of XML schema is to define the building blocks of an XML document. This is an alternative of DTD. The XML schema language is called as XML Schema Definition Language (XSD). XML schema defines elements, attributes, elements having child elements, order of child elements, fixed and default values of elements and attributes. The following is an example XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="student">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name"></xs:element>
        <xs:element name="address"></xs:element>
        <xs:element name="marks"></xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Data Types: Various data types that can be used to specify the data types of an element are

1. **String**
2. **Date**
3. **Numeric**
4. **Boolean**

1. **String Data Type:** The string data type is used to define the element containing characters. For example

student.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="student" type="xs:string"></xs:element>
</xs:schema>
```

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">Naresh </student>
```

2. **Date Data Type**: Date data type is used to specify the date. The format of this date is yyyy-mm-dd denotes year, month and date. For example

student.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="dob" type="xs:date"></xs:element>
</xs:schema>
```

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<dob xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">2001-09-01
</dob>
```

3. **Numeric Data Type:** To represent decimal or integer values Numeric Data Type is used. For example

Example1:

student.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="salary" type="xs:decimal"></xs:element>
</xs:schema>
```

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<salary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">9876.54
</salary>
```

4. **Boolean Data Type:** for specifying true or false Boolean data type is used. For example

student.xsd

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="flag" type="xs:boolean"></xs:element>
</xs:schema>
```

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<flag                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd"> true
</flag>
```

Document Object Model:

The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model. It is the W3C recommendation for handling the structured documents. In DOM everything from XML is treated as a node.

What the Document Object Model is

The DOM is a programming API for documents. It is based on an object structure that closely resembles the structure of the documents it models.

DOM History and levels

- A Simple Document Object Model, was implemented in Netscape 2.0 browser. In this implementation, **write()** method was introduced. This method was useful to add the contents dynamically do a document. Using this method, the developers were able to access form controls that could be changed dynamically by JavaScript document. But this version of DOM was unable to modify images present in the document.
- Later on Netscape 3.0 introduced newer DOM version which was able to handle image and other elements of browser.
- Then Netscape navigator 4.0 and Internet Explorer came up newer versions of DOM both independently. But both these versions had lot of differences within them. Hence developers found it difficult to handle documents using DOM when browser was getting changed.
- Hence W3C came up to resolve these differences and created the standard initial DOM version in 1997.
- Various levels of DOM was listed in the following table.

Level	Description
DOM0	This model is supported by early browsers. This level could support JavaScript.

	This was implemented in Netscape 3.0 and Internet Explorer 3.0 browsers.
DOM1	This version was issued in 1998 which was focused on XHTML and XML.
DOM2	This version was issued in 2000 that could specify the style sheet. It also support the event model and traversal with in documents.
DOM3	This is current release of DOM specification published in 2004. This version could deal with XML with DTD and Schema, document validations, document views and formatting.

- Basically Document Object Model (DOM) is an application programming interface (API) between XML and application programs
- The DOM contains set of interfaces for document tree node type. These interfaces are similar to the Java or C++ interfaces. They have objects, properties, methods which are useful for respected node type of web document.

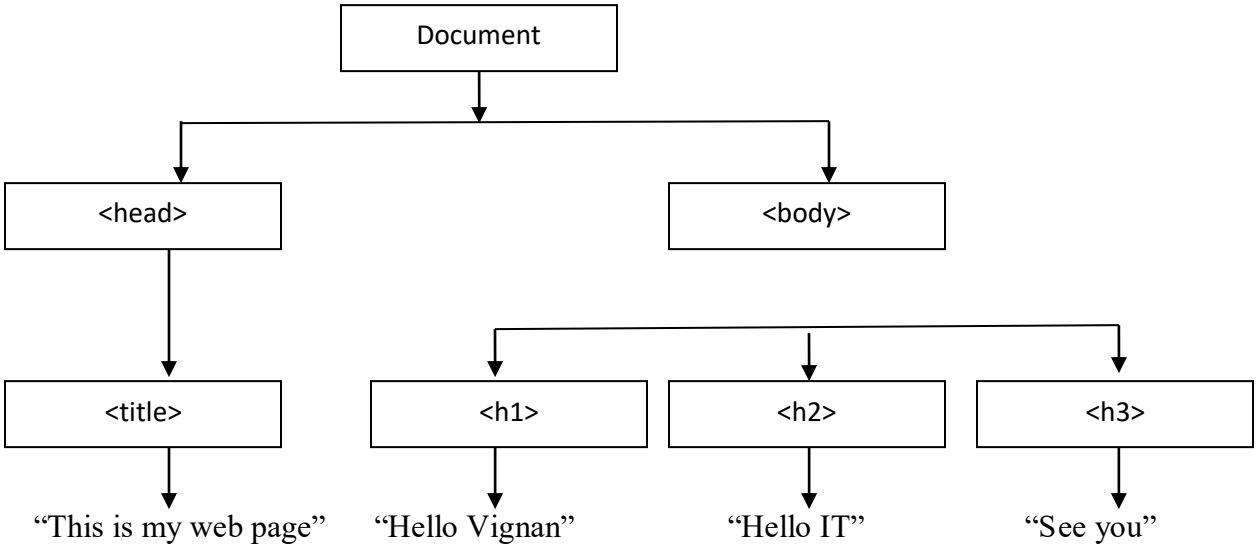
DOM Tree

The documents in DOM are represented using a tree like structure in which every element is represented as a node. Hence the tree structure is also referred as DOM tree.

Example: Consider the following XHTML document

<html>
<head> <title>This is my web page</title> </head> <body>
<h1>Hello vignan</h1> <h2>Hello IT</h2>
<h3>See you</h3> </body> </html>

The DOM tree for the above example document is as follows.



We can describe some basic terminologies used in DOM tree as follows.

1. Every element in the DOM tree is called **node**.
2. The topmost single node in the DOM tree is called as **root**.
3. Every child node must have a parent **node**.
4. The bottommost nodes that have no children are called **leaf nodes**.
5. The nodes that have common parents are called as **siblings.**

Element access in JavaScript

There are several ways by which we can access the elements of web document.

Consider the following document to understand the ways of accessing elements in JavaScript.

Example:

```
<html>
<head>
<title>This is my web page</title>
</head>
<body>
<form name=”form1”>
    <input type=”text” name=”myinput”/>
</form>
</body>
</html>
```

Method 1:

Every document element is associated with some address. This address is called **DOM address.** The document has the collection of **forms** and **elements**. Hence we can refer the element as

```
var DOM_Obj=document[0].elements[0];
```

But this is not a suitable method of addressing the elements. Because if we change above script as

```
…….
<form name=”form1”>
    <input type=”button” name=”mybutton”/>
    <input type=”text” name=”myinput”/>
</form>
……
```

Then index reference gets changed. Hence use another method.

Method 2:

In this method we can make use of **name** attribute in order to access the desired element.

var DOM_Obj=document.form1.myinput;

But this may cause a validation problem because XHTML1.1 does not support **name** attribute to the form element. Hence use another method.

Method 3:

We can access the desired element from the web document using JavaScript method **getElementById.** This method is defined in DOM1. The element access can be made as follows.

var DOM_Obj=document. getElementById("myinput");

But if the element is in particular group, that means if there are certain elements on the form such as radio buttons or checkboxes then they normally appear in groups. Hence to access elements we make use of its index. Consider following sample document.

```
<form id="Food">
    <input type="checkbox" name="vegetables" value="spinatch"/> Spinatch
    <input type="checkbox" name="vegetables" value="FenuGreek"/> FenuGreek
    <input type="checkbox" name="vegetables" value="Cabbage"/> Cabbage
</form>
```

For getting values of these checkboxes we can write the following code.

```
var DOM_Obj=document. getElementById("Food");
for(i=0;i<Dom_Obj.vegetables.length;i++)
document.write(Dom_Obj.vegetables[i]+"<br/>");
```

XSLT

XSL stands for EXtensible Stylesheet Language, and is a style sheet language for XML documents. XSL transform XML documents into other formats.

XSLT stands for XSL Transformations. In this we will learn how to use XSLT to transform XML documents into other formats, like XHTML.

Example:

XML Code:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
    <cd>
        <title>Empire Burlesque</title>
        <artist>Bob Dylan</artist>
        <country>USA</country>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
    </cd>
    <cd>
        <title>Hide your heart</title>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler

```
        <artist>Bonnie Tyler</artist>
        <country>UK</country>
        <company>CBS Records</company>
        <price>9.90</price>
        <year>1988</year>
    </cd>
</catalog>
```

By using XSL editor we can convert the XML into XSL source code.

XSLT Code:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

SAX (Simple API for XML): SAX is an event-based parser for XML documents. Unlike a DOM parser, a SAX parser creates no parse tree. SAX is a streaming interface for XML, which means that applications using SAX receive event notifications about the XML document being processed an element, and attribute, at a time in sequential order starting at the top of the document, and ending with the closing of the ROOT element.

- Reads an XML document from top to bottom, recognizing the tokens that make up a well-formed XML document.

- Tokens are processed in the same order that they appear in the document.
- Reports the application program the nature of tokens that the parser has encountered as they occur.
- The application program provides an "event" handler that must be registered with the parser.
- As the tokens are identified, callback methods in the handler are invoked with the relevant information.

When to Use?

You should use a SAX parser when −

- You can process the XML document in a linear fashion from top to down.
- The document is not deeply nested.
- You are processing a very large XML document whose DOM tree would consume too much memory. Typical DOM implementations use ten bytes of memory to represent one byte of XML.
- The problem to be solved involves only a part of the XML document.
- Data is available as soon as it is seen by the parser, so SAX works well for an XML document that arrives over a stream.

Disadvantages of SAX

- We have no random access to an XML document since it is processed in a forward-only manner.
- If you need to keep track of data that the parser has seen or change the order of items, you must write the code and store the data on your own.

Given the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentElement param="value">
   <FirstElement>
      &#xb6; Some Text
   </FirstElement>
   <?some_pi some_attr="some_value"?>
   <SecondElement param2="something">
      Pre-Text <Inline>Inlined text</Inline> Post-text.
   </SecondElement>
</DocumentElement>
```

This XML document, when passed through a SAX parser, will generate a sequence of events like the following:

- XML Element start, named *DocumentElement*, with an attribute *param* equal to "value"
- XML Element start, named *FirstElement*
- XML Text node, with data equal to "¶ Some Text" (note: certain white spaces can be changed)
- XML Element end, named *FirstElement*
- Processing Instruction event, with the target *some_pi* and data *some_attr="some_value"* (the content after the target is just text; however, it is very common to imitate the syntax of XML attributes, as in this example)
- XML Element start, named *SecondElement*, with an attribute *param2* equal to "something"
- XML Text node, with data equal to "Pre-Text"
- XML Element start, named *Inline*
- XML Text node, with data equal to "Inlined text"
- XML Element end, named *Inline*
- XML Text node, with data equal to "Post-text."
- XML Element end, named *SecondElement*
- XML Element end, named *DocumentElement*

AJAX A NEW APPROACH

Introduction to AJAX

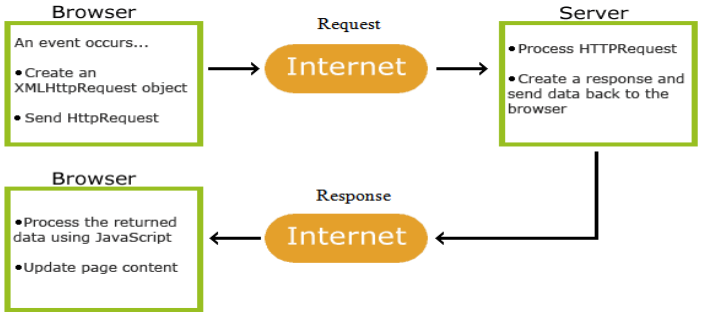
The AJAX is involved in both client and server communication. The Perl and CGI scripts normally run at server side and process the request obtained from the client side through browser.

AJAX is a Asynchronous Java Script and XML. It is not a new programming language but it is a kind of web document which adopts certain standards. AJAX allows the developer to exchange the data with server and updates the part of web document without reloading the web page.

How AJAX works?

When user makes a request, the browser creates a object for the Http request and a request is made to the server over an internet. The server processes this request and sends the required data to the browser. At the browser side the returned data is processed using java script and the web document gets updated accordingly.

Following fig illustrates this working:

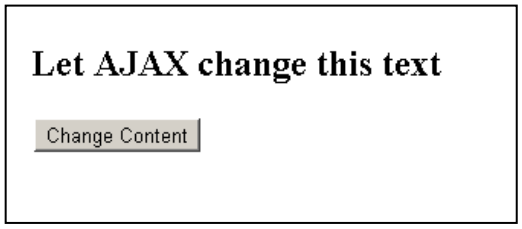


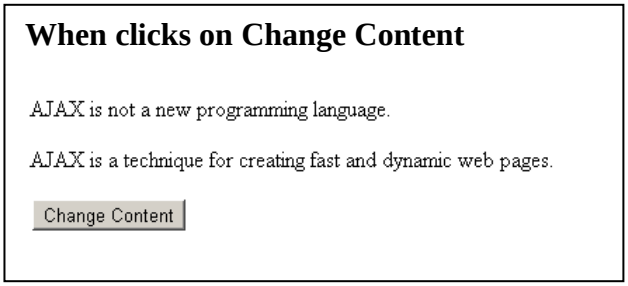
Let us understand how AJAX works with the help of programming example.

```
<html>
<head>
<script>
function myfun()
{
var req;
if (window.XMLHttpRequest)
  {
  req=new XMLHttpRequest();
  }
else
  {
  req=new ActiveXObject("Microsoft.XMLHTTP");
  }
req.onreadystatechange=function()
  {
  if (req.readyState==4 && req.status==200)
    {
    document.getElementById("myDiv").innerHTML=req.responseText;
    }
  }
req.open("GET","ajax_info.txt",true);
req.send();
}
</script>
</head>
<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="myfun()">Change Content</button>

</body>
</html>
```





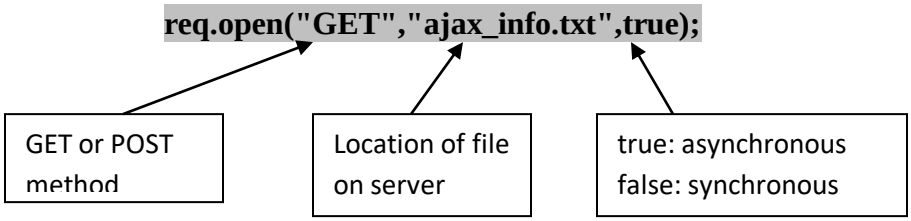
In above sript,we have written some text which can be replaced by some another text on button click. On button click a function MyFun is invoked. In this function

1.**XMLHttpRequest** object is used to exchange data with a server. The object allows the user to change/update the parts of the webpage without reloading it fully. The modern web browsers such as IE7+,firefox,chrome have built in **XMLHttpRequest** but old web browsers make use of **ActiveXObject**.

2. When a request to a server is sent,the onreadystatechange event is triggered.

The **readyState** property holds the status of the **XMLHttprequest**. The **readystate**=4 means request is finished and response is ready . The **status**=200 means "OK"

3. The request can be sent to the server by using two functions **open()** and **send().**



Asynchronous communication allows fast processing of the data

The **ajax_info.txt** file contains some updating text using which our web page can be updated.

The **send()** method sends the request to the server.

Integrating PHP and AJAX

We can use Ajax along with XML and PHP. In the following example we will discuss how, HTM L file along with java script communicates with XML and PHP. It will work as follows:

1. HTML displays the form that contains a drop down list. User can select his/her friend`s name from the dropdown list.
2. When user selects some name, a function named **showNames** will be triggered. This function is defined in java script file.
3. This function in java script file will send the name as a query string to some PHP file. The name of the PHP file is considered as URL
4. The PHP file makes use of DOM. It will load XML file using DOM. Using DOM object we go through each node of XML file and retrieve the contents.
5. These contents are then returned to the HTML using the **innerHTML**. Hence on browser we can get the details of the friend whose name we have selected.

Step1: Create an HTML document for displaying the form.

AJAXDemo.html

```
<html>
<head>
<script src="testing.js"></script>
</head>
<body>
<form>
Select a Name:
<select name="names"onchange="showNames(this.value)">
<option value="chitra">chitra</option>
```

```
<option value=”priyanka”>priyanka</option>
<option value=”Raj”>Raj</option>
</select>
</form>
<p>
<div id=”txtHint”><b>Friend Details:</b></div>
</p>
</body>
</html>
```

Step2: The java script will be as follows. It would contain function **showNames**.

Testing.js

```
var xmlHttp;
function showNames(str)
{
xmlHttp=GetxmlHttpobject();
if(xmlHttp==null)
{
alert(“Browser does not support HTTP Request”);
return;
}
var url=”getInfo.php”;
url=url+”?q=”+str;
url=url+”&sid=”Math.random();
xmlHttp.onreadystatechange=statechanged;
xmlHttp.open(“GET”,url,true);
xml.Http.send(null);
}
function statechanged()
{
if(xmlHttp.readystate==4||xmlHttp.status==200)
{
document.getElementBYId(“txtHint”).innerHTML=xmlHttp.responseText;
}
}
```

```
function GetxmlHttpobject()
{
var xmlHttp=null;
try
{
//Firefox,opera 8.0+,safari
xmlHttp=new XMLHttpRequest();
}
catch(e)
{
//Internet Explorer
try
{
xmlHttp=new ActiveXobject("Msxml2.XMLHTTP");
}
catch(e)
{
xmlHttp=new ActiveXobject("Microsoft.XMLHTTP");
}
}
return  xmlHttp;
}
```

Step3: The PHP script that normally runs on the server side is as given below. It will make use of DOM to load and handle the XML file.

getinfo.php

```
<?php
$q=$_GET["q"];
$xmlDoc=new DOMDocument();
$xmlDoc->load("FriendNames.xml");
$a=$xmlDoc->getElementsByTagName('name');
for($i=0;$i<=$a->length-1;$i++)
{
//going through elements
if($a->item($i)->nodeType==1)
{
```

```
 if($a->item($i)->childNodes->item(0)->nodevalue==$q)
 {
   $b=($a->item($i)->parentNode);
}
}
}
$fr=($b->childNodes);
for($i=0;$i<$fr->length;$i++)
{
//going through other elements
if($fr->item($i)->nodeType==1)
{
echo("<b>".$fr->item($i)->nodeName.":</b>");
echo($fr->item($i)->childNodes->item(0)->nodevalue);
echo("<br/>");
}
}
?>
```

Step4: The XML file which is handled by the PHP in the above step is:

FriendNames.xml

```
<?xml version="1.0"?>
<Friend>
<Info>
<name>chitra</name>
<phone>1111111111</phone>
<email>chitra_abc@gmail.com</email>
<hobby>singing</hobby>
</Info>
<Info>
<name>priyanka</name>
<phone>2222222222</phone>
<email>pr123@rediffmail.com</email>
<hobby>Reading</hobby>
</Info>
<Info>
```

<name>Raj</name>
<phone>333333333</phone>
<email>Raj_2008@hotmail.com>/email>
<hobby>photography</hobby>
<Info>
</Friend>
Step5: for getting the output we will open the HTML file (Created in step 1) in browser window