WEB TECHNOLOGIES

UNIT 1

1. History of the Internet and World Wide Web

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.

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.

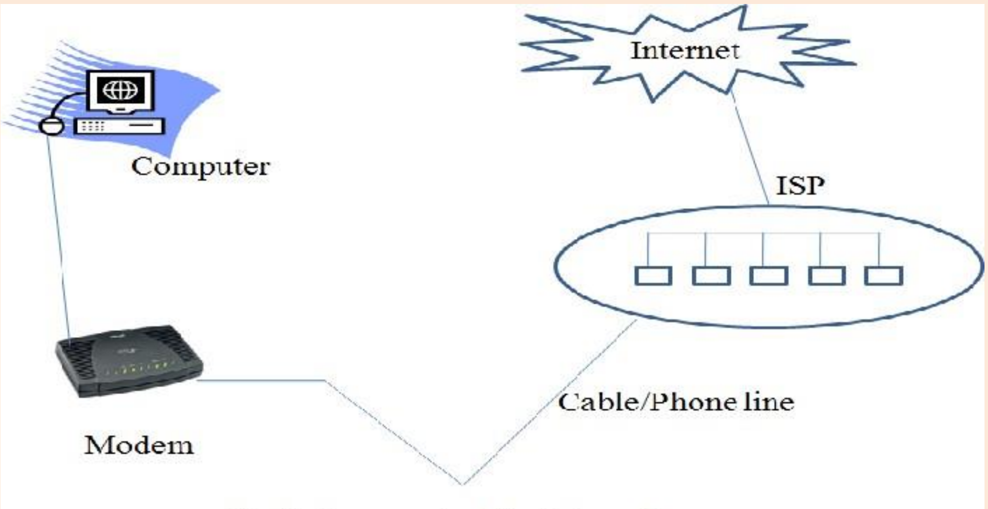


Fig: Setup required for Internet

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.

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.

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

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:\KHITs\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

<b> </b>

8. **Italic tag:**

This implements italic effects on the text.

<i>..........</i>

9. **strong tag:**

This tag is used to always emphasized the text

<strong>...............</strong>

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.

<sub> ....</sub>
<sup> ....</sup>

12. Break tag:

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

<br>

13. &amp &lt &gt &nbsp &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> <a href=" path of the file"> some text </a> 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> KHIT Home page </title>
</head>
<body >
<h1  align="center">              KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY </h1>
<h2 align="center"> CHOWDAVARAM, Guntur</h2>
<basefont size=40>
<p> This college runs under the <tt>management</tt> of <font size=5> <b><i>&quot
KES Education Society &quot &amp The Institute is a dream come true of its founder Sri. KALLAM HARANADHAREDDY - an famous Industrialist 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"> CSE Department</a><br>
</body>
</html>
```

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>
      <li> Text to be displayed</li>
      </ul>
```

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” > …..</ul>

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”>.....</ol>

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>
```

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>
```

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> KALLAM HARANADHAREDDY  Institute of Technology <b></p>
<img src="e:\KHIT.jpg" width="225" height="151" alt="khit">
</body>
</html>
```

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>
```

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=”sates list” 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.

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 </a> 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 <a href=" ">.

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 </a>

Example:

```
<html>
<head>
<title>Use of links on webpage>
</title>
</head>
<body>
<a href=" https://khitguntur.ac.in/"><b>Click here</b></a><b> to open KHIT's Home page</b>
</body>
</html
```

If we click on the hyperlink then we will get the KHIT'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> KALLAM HARANADHAREDDY  Institute of Technology </h1>
<a href="https://khitguntur.ac.in/ ">
<img src="E:\KHIT.jpg" alt="KHIT" />
</a>
<br> KHIT 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**.

```
<img src="d:\KHIT.jpg" alt="KHIT" align=”right”/>
```

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.

→ Bernarlly introduced html in the year 1991

→ HTML doccument is a collection of HTML Element :

→ Tag is Enclosed b/w < >

→ Element is collection of open tag, content, closed tag.

<tag> content </tag>

→ Attribute is also called as property.

<u>structure</u> :-

title
style
script
meta data } head part contains

```
<html>
      <head>
            <title> My Webpage </title>
      </head>
      <body>
            <h1> xy </h1>
            <h2> pq </h2>
<b><i><u> web Technology </u></i></l
 <br>
<b><del> web </del> <b>
<b> <s> web <Is> <b>
```

```
<big> cse </big> <br>
<b> x <sub> 2 </sub> </b> <br>
<b> x <sup> 2 </sup> </b>
<b> <bdo dir= ltr > hello </bdo>
 </body>
</html>
```

bdo → biderectional text.

  → ~~space b/w words~~ non-brealing spaces

& copy → ©

& reg → ®

<marquee> web </marquee > — to move text

& cent - ¢

& pound - £

& Empty - ∅

<body bgcolor= "RED">

<body background = "path">

Self ending tag

In font tag default size is 3

<font space = "fontname" size="7" color="green">

<s>
<del>
<strike> } used for same purpose

→ <strong> web </strong >
(or)
bold

→ double quotes → <q> content </q>

→ headings → <h1> KHIT </h1>

→ <ins> (or) <u> → for underlining.

→ <del> (or) <s> (or) <strike> → for striking.

<img>: image tag is used to insert the image into web page.

Tag name	Attribute	Des
<img>	1) src 2) height 3) width 4) alt	used to insert the image into web page.

alt - alternative text.

Note: <img is self closing tag

```
<!DOCTYPE html>
<html>
    <head>
      <title> my image </title>
    </head>
    <body>
```

```
<img src = "d:images\csea.jpeg"
     width = "70%" height = "70%"
     alt = "csea image" >

</body>
</html>
```

<list> :-

lists are of 3 types in html — 1) Ordered list <ol>

child tag is <li> — 2) unordered list <ul>

3) definition list <dl>
- → definition term <dt>
- → definition description <dd>

Eg:- ordered list

```
<!DOCTYPE html>
<html>
   <head> ~~My list~~ ~~</head>~~
      <title> My list </title>
   </head>
```

```
<body>                    "i" → roman numbers
   <ol type = "A" start= "5" >
      <li> CSE </li>
      <li> ECE </li>

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

* type & start are the attributes of ordere list.

Unordered list

Eg :-

```
<!DOCTYPE html>
<html>
  <head>
     <title> My list </title>
  </head>
  <body>              → default type
  <ul type = "disc" >        types - circle
                                   - square
                                   - disc
     <li> CSE </li>
     <li> ECE </li>
  </body> </ul>
</html>
```

definition list

Eg:-
```
<!DOCTYPE html>
<html>
  <head>
    <title> My list </title>
  </head>
  <body>
    <dl>
      <dt> Computer </dt>
        <dd> It is an Electronic device </dd>
      <dt>
          ≡
    </dl>
  </body>
</html>
```

Table tag :-

→ html table allows to arrange the data like text, image, link, other tables Etc.., into rows and columns.

→ html tables are created using <table> tag

Eg:-
```
                        "1" → thick border
<table border = "0">
                  ↳ no border
  <tr>
    <th> sno </th>
    <th> sna </th>
  </tr>
  <tr>
    <td> 101 </td>
    <td> ABC </td>
  </tr>
</table>
```

→ attribute of table - bgcolor, background, rowspan, Columnspan, height, width, ~~self~~ cell spacing, cell padding.

Cell ~~padding~~ spacing :- It defines the width of the cell border.

Cell padding :- It represents the distance b/w the border and the content with in a cell

Column span:- <Col span>

It is used to merge 2 or more columns into a single column.

Row span:-

It is used to merge 2 or more rows into a single row.

```
Eg:- <html>
       <body>
         <table>
           <caption> student </caption>
         <tr><th>sno</th> <th>sna</th></tr>
         <tr><td> 101 </td> <td> ABC </td></tr>
         <tr> <td colspan="2"> </td> </tr>
         </table>
       </body>
     </html>
```

marquee:- <marquee>

→ Attributes of <marquee> are

- * width
- * height
- * direction (up, down, left, right)
- * behavior (scroll, side, alternate)
- * scroll delay
- * scroll amount
- * loop
- * bgcolor
- * hspace
- * vspace

Html comment:-

<!-- Comment -->

Eg <! -- Html -->

→ Browser discards the html comments.

→ Comments are of 2 types:- Invalid commen
valid "

Marked text:- <mark>

Used to highlight the content in a documen

→ Default color is light yellow color.

Eg:- <mark> JNTUK </mark>.

→ <code> - - </code> - for code font

→ <samp> --- </samp> - for o/p font

Paragraph :-

<p> — </p>

<p align="center"> - - - </p>

→ Default alignment is left.

→ <kbd> – keyboard font

→ <div> – division of webpage

Form:-

→ Forms are used to collect data from the user.

eg :- Registration form, feed back form, login form Etc..)

→ By using forms we can read the data from the user and sent the data to the backend application such as CGI, ASP, PHP

→ Backend application to perform required operation / processing on the passed data based on the defined business logic inside the application.

→ There are various form Elements

- <form>
-

<form> :- The HTML <form> tage is used to create an HTML form.

→ Attributes of <form> are action & method.

⟹ action value is the url of the script/ server location we are using url of the server.
To specify the server application location

⟹ Method attributes has get method value for unsecured data and post method value for secured data which doesn't appear in the url.

→ Syntax for <form> is

```
<form action = "Script URL" method = "GET | POST">

</form>
```

Note:- In form if we are not using any method default method is get method.

Eg :-

```
<!DOCTYPE html>
<html>
 <body>
  <form action="http:\ ..." method="post">
  <label for= "cse"> Name</label>
   Name : <input type="text". name="f_name"
                    placeholder="Enter name" id="cse"/>
  <br>
  Password : <input type="password" name="password"
                                          />
  <br>
  <input type="submit" name="submit"
                          value="submit"/>
 </form>
 </body>
</html>
```

⇒ <textarea>:

```
<textarea row="5" cols="50" name="description">
     Enter your self in 500 words </textarea>.
```

⇒ <input> : of <form> tag.

→ Element

→ Self closing tag

→ attributes of input are type, value, name, id, placeholder, size, maxlength.

→ Button & submit are different. In submit without value attribute the data is submitted to the server.

→ But in case of button without value attribute it displays a Empty textbox and a click on this will never submit the details to the server.

```
<input type="text/password/submit/radio/checkbox/button/mail(@.)etc../>
```

text →
name=" "
value=" "
id=" "
size=" "
maxlength=" "
placeholder=" "

password →
name=" "
id=" "
size=" "
maxlength=" "
minlength=" "

submit →
Name=" "
Size=" "
Value=" "

radio →
name=" "
Checked

checkbox →
name=" "
value=" "
checked

button →
name="
value="
Size="

Form Eg :-

```
<!DOCTYPE html>
<html>
 <head>
   <title> Fronform < title>
 <head>
```

```
<body>
    <form action="url" method="get/post">
Name:
~~[illegible]~~: <input type="text" name="~~[illegible]~~" placeholder="enter ur
                                  ~~name~~ name" />
    <br>
Surname
~~Password~~: <input type="~~password~~ text" name="~~password~~ surname" placeholder="enter
                                  ur ~~password~~ surname" />
<label> Qualification </label>
<br>
SSC <input type="checkbox" name="ssc" value="ssc"
                                                checked>
Inter
 Degree
 PG
<label> Gender </label> <br>
<input type="radio"> female male <br>
<input type="radio"> female.
<input type="radio"> others.
<label> Address </label> <br>
 <textarea row="5" cols="20" name="Address">
        Enter your Address </textarea>
```

```
<input type="submit" value="Submit" name="submit">
   </form>
  </body>
 </html>.
```

Selection:

```
<select name="Days">
   <option name="SUN"> SUN </option>
   <option name="MON"> MON </option>
         ⋮
   <option name="SAT"> SAT </option>
</select>
```

Frames:-

→ Introduced by Netscape cooporation

→ Collection of frames appeared in the window is called frameset.

→ To display more than one site at a time frames are used. i.e., to divide the window into parts.

Drawbacks:-

→ Navigation is not possible.

→ All browsers are not guaranteed to work with frame

→ If display size is small frames appearance is not clear.

→ From HTML5 version onwards frames are not available.

→ Tags that are using in frames are:-

1) <frameset>
2) <frame>
3) <noframes>

→ No need of writing frames in body tag.

Eg:-

```
<html>
  <head>
    <title> ExonFrame </title>
  </head>
  <frameset row="50%, 50%">
     <frame src="url"  name="  ">
           (By default frame having scroll behavior)
     <frame src="url"  name="  ">
  </frameset>
</html>
```

→ <frameset> attributes are

	Attributes	parameters
1.	cols	% (or) pixels
2.	rows	% (or) pixels
3.	border	pixels
4.	bordercolor	Color Name or Hash code or rgb function

→ <frame> attributes are

	Attribute	parameter
1.	Name	Any Name
2.	src	Url (or) filepath
3.	scrolling	Yes (or) No (or) default ↓ by default

→ <noframes>

```
<p align="center">
    This. Browser is not supporting frames
</p>
</no frames>
```

whenever the browser failed to execute the frame in that case only noframes content is displayed on the web page otherwise not display

4/12/1

Content + style = HTML Document without css

CSS (Cascading style sheets) :—

→ Introduced by Hakon wiem lie in 1994.

→ Content & style are Seperated in Css

Content + style (css) = Document

→ Before cascading style sheets web pages are developed by the developers like the following manner

Content + Style = HTML Document

→ In HTML content of the web page and style of the web page are prepared within a single document.

→ without css change in (or) modification in the document is a difficult task.

→ By using css, Content of the webpage & style of the webpage are Separated.

→ one webpage developer focuses on the modification of webpage content and another webpage developer focuses on the modification of webpage style.

→ Extension used is .css

file.css

style of the webpage

→ In general css are classified into 3 types

1) External css

2) Internal css

3) Inline css

External css :—

step 1:

file.css

style of the webpage

step 2:

file.html

web page content
<link>

```
<html>
  <head>
  <lint href = " path ">
  ~~</link>~~
  </head>
  < body>
  </body>   </html>
```

→ <link> must be in head part.

→ Tag definitions are available In .css file.

Eg:- step1 : Example.css

```
body {
  background-Color:Green;
  }
h1 {
  Color: Red;
  }
P {
  Color: Blue;
  }
```

step 2: Ex1.html

```
<!DOCTYPE  html>
<html>
  <head>
  <link rel="stylesheet",
        href=" path">
   </head>      d:/Example.css
   <body>
 <h1> First Heading </h1>
  <p> First Paragraph </p>
 </body>
<html>
```

Eg:- step 1: example.css

```
.red {
   Color: Red;
   }
.green {
   color: Green;
   }
.thick {
   font-size: 20px;
   }
```

step 2: Ex2.html

```
<!DOCTYPE  html>
<html>
  <head>
   <link rel="stylesheet",
         href = "d:/example.css">
   </head>
   <body>
  <p Class="red"> Red Color </p>
  <p Class = "green"> Green color </p>
  <p class = "thick green"> Thick green</p>
  </body>
</html>
```

Internal css / Embedded style sheets :—

→ It is also called as Embedded style sheet.

→ Here we are not creating any .css file

→ we are using <style> tag to create the style which is placed in the head part of the html document.

Eg:

```
<html>
  <head>
    <style>
      body{ background-color : Green;}
       h1{ color: Blue ;}
       p { Color : Red ;}
    </style>
  </head>
  <body>
<h1> This is Internal css </h1>
 <p> Here we are using style tag to
          Create styles </p>
  </body>
</html>.
```

Eg2:

```
<html>
   <head>
     <style>
       .Red { color : Red ;}
       .Thick { font-Size : 20px ;}
       .Green { Color : Green ;}
     </style>
   </head>
  <body>
   <p class = "Red" > Examples on Internal css </p>
   <p class = "Thick Green"> Internal css </p>
   </body>
 </html>
```

Inline css :—

→ we can apply style sheet rules directly to any html Element using style attribute of the relevant Tag in inline style sheet.

Note:- This should be done only when we are interested to make a particular change in any html Element.

Eg:-
```
<html>
  <head>
    <title>Inline css</title>
  </head>
  <body>
  <p style="color:green";> Inline style sheet Ex</p>
  <p style ="font-size:30px"> Ex on font </p>
  </body>
</html>
```

Imported Style Sheets :-

Another way of importing a style sheet is to use @import.

→ It works like link tag. It allows importing a style sheet from another style sheet also.

→ we should write import statement within the style tag in html document.

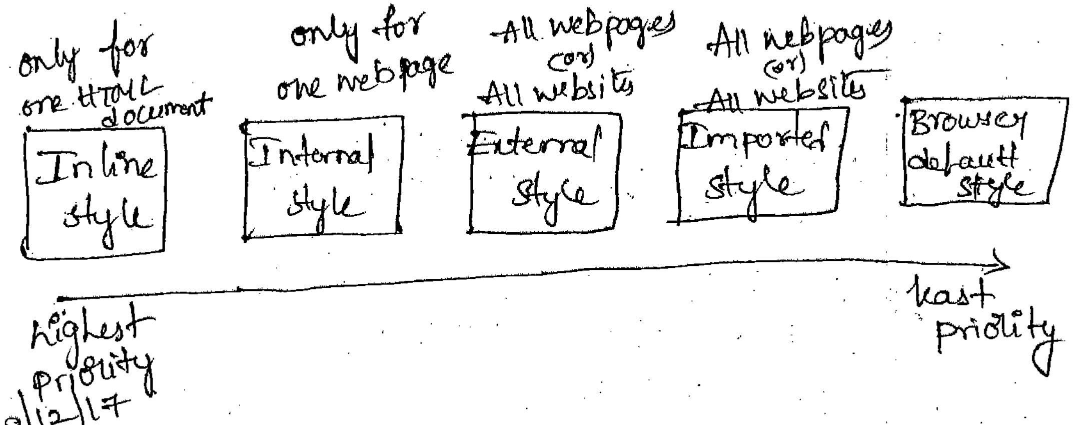
Syntax: @import url("path");

Eg:- Step 1: Ex.css

```
body {background-color:green;
     }
.Red {
     Font-size:20px;
       color:Red;
     }
```

Step 2: Example.html

```
<html>
 <head>
  <style>
   @import url("d:\Ex.css");
  </style>
 </head>
 <body>
 <P class ="red"> Imported style sheet
                                   </P>
  </body>
</html>
```



only for one HTML document

Inline style

only for one webpage

Internal style

All webpages (or) All websits

External style

All webpages (or) All websites

Imported style

Browser default style

highest priority

least priority

9/12/17

Selectors :

Selector is nothing but a name which contains one or more declarations.

→ Declaration is a property - value pair.

→ Types of Selectors

1. Element Selector
2. ID "
3. Class "
4. Element specific Class "
5. Universal "
6. Descendent " / Contextual Selector
7. Child "
8. Grouping "

→ The element Selector selects elements based on the element name.

Eg:- P{ } h1{ } body{ }

```
P {
  text-align: Center;
   Color : Red;
 }
```

ID Selector :-

→ ID Selector uses attribute of HTML Element.

→ selector appears with '#' symbol called ID selector

```
#para1 {
   text-align: center;
     color : Red;
   }
```

Eg:- <p id = "para1" > Hello </P>

Class selector :-

→ Class selector selects elements with a specific Class attribute

→ Appears with period (.)

```
.Center {
    text-align : Center;
      Color : Green;
   }
```

Eg:- <p class = "center" > Hello </P>

Element specific class Selector :-

```
P.Center {
  }
```

Eg:- <P class = "center" > ______ </P>

Grouping Selector:

If you have Elements with the same style difinition

like
```
h1 {
text-align: center;
  color: green;
     }
   h2 {
   text-align: center;
    color: green;
     }
```
We can group these selectors to minimize code
```
h1,h2 {
    text-align: centers
      color: green;
    }
```

Universal Selector:

```
*{
  background-color: green;
 }
```
→ Applies to all

Descendent Selector: / Contextual selector

```
div p {    → p is descendent of div

     }
```

Child Selector:-

Parent > Child

Java Script:-

Java Script is a object based scripting language

→ Object based scripting languages has built in object.

→ Where every client side validations are required Java script is used.

→ Java Script is a cross-platform scripting language.

→ Introduced in the year 1995 by netscape communications. by Brendend Eich

1/12/17 → present Version 1.8.5

Structure of Java Script:-

```
<script>
   Explanation part
 Java Script code {Execution part}
   Embedded code

</script>
```

Explanation part — 1. // - single line documentation

2. /* --- */ Multi line "

3. <!-- ---- --> Html Documentatio

JS Variables:-

A Javascript variable is simply a name of storage location.

→ Two types of variable :- local variable
Global ,,

→ There are some rules

* Names must start with letter (or) underscore (or) dollar
* After first letter we can use digits
* Js variables are case sensitive.

→ window. Variable name = value; is treated as global variable with in the block also.

Data-types :-

Js is a dynamic language. so we needn't to mention the type. After assigning the value their type is decided.

→ primitive DT – string
Number
Boolean
Undefined
Null

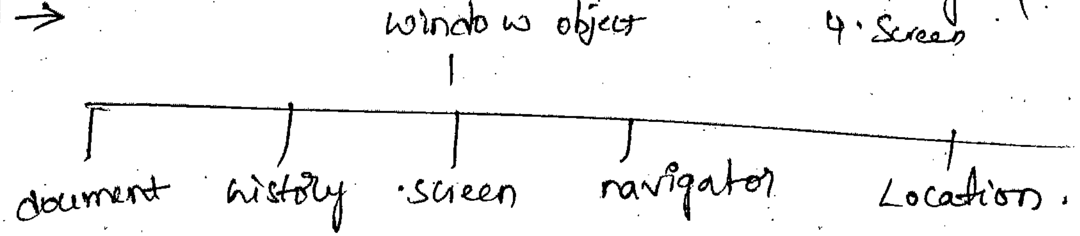
→ Non-primitive D.T – Object
Array
RegExp

Browser Object Model

→ BOM is used to interact with browser
→ Default object is window.

1. window
2. History
3. Navigator
4. Screen

→



→ Methods in window are :-
alert(), confirm(), prompt(), open(), close(), setTimeout()

History:

→ History objects stores url in the form of array visited by the user.

→ Methods in History are :-
forward(), back(), go().

Navigator:

→ Used for browser detection.

→ Methods are javaEnabled(), taintEnabled()
↓
depricated.

Screen:

→ Holds information of browser screen.

Document Object Model

→ The document object represents the whole html document.

→ When html is loaded in the browser, it becomes a document object.

→ Methods available are
write( ), writeln( )
getElementById( ), getElementByName( )
getElementByTagName( ), getElementByClassName( ).

18/12/17

UNIT-II
XML (Extensible Markup Language).

→ XML is a universal language used for communication b/w applications (different)

→ XML is case sensitive.

→ Introduced in the late 1990s by W3C.

→ XML document is divided into 2 parts
1) prolog and
2) body.

→ XML is a language independent & platform independent.

XML document content:

Contains
1. ELEMENTS
2. ATTRIBUTES
3. Entity References.
4. Mixed of All

<open tag> Text data </close tag>

Text data contains Entity references.

→ XML provides 5 built in Entity references.

1. &lt; — <
2. &gt; — >
3. &amp; — &
4. &quot; — " "
5. &apos; — ' '

Restrictions :-

To provide restriction in xsd the following constraints are used.

1. minInclusive
2. maxInclusive
3. minExclusive
4. maxExclusive
5. totalDigits
6. fractionDigits
7. pattern
8. whiteSpace
9. enumeration
10. length
11. minLength
12. maxLength.

Syntax:

```
<Element "elementname">
 <simpleType>
 <restication base = "datatype"> </restriction>
 </simpleType>
</Element>
```

Eg:

```
<Element name= "salary">
 <simpleType>
 <restriction base = "decimal">
 <minInclusive value = "10000"/>
 <maxInclusive value= "40000"/>
 </simpleType>
 </element>
```

Eg on pattern:

```
<element name ="gender">
 <SimpleType>
 <restriction base = "string">
 <pattern value =[MFO]"/> (or) <pattern value =[A-z a-z][]-
                                              --[]*">
                                              (or)
 </restiction>                          "Male/Female/others"
 </Element>
```

in XML:

```
<gender> M </gender>
      (or)
<gender> F </gender>
      (or)
<gender>O</gender>
```

User defined data types:-

There are of 2 types

1. Atomic (similar to class concept in java)
2. Non-Atomic — list / union

Atomic DataType:-

Eg:

```
<simpleType name= "salarytype">
 <restrictions base = "decimal">
 <min Inclusive value = "10000"/>
 <maxInclusive value = "60000"/>
 </restrictions>
 </simpleType>
```

In xsd:

```
<Schema><element name = "employee">
 <ComplexType>
  <Sequence>
   <Element name = "eno" type = "int" />
   <Element name = "Salary" type = "SalaryType" />
  </sequence>
 </complexType>
</schema>
```

Non-Atomic :—

List:

⇒ List is used to Specify multiple values for same Element

Syn:
```
<xs: simpleType name = "~~my~~Contactlist">
   <xs: list itemType = "xs:int" />
  </xs: simpleType>
```

In xsd:

```
<xs: Schema >
 <xs: Element name = "Employee">
 <xs: ComplexType>
 <xs: sequence>
 <xs: element name = "Eno" type = "xs:int" />
 <xs: Element name = "mbno" type = "Contactlist" />
</xs: sequence>
</xs: complexType>
```

Union: To combine more than one atomic data types as a single data type.

Eg:
```
       <xs: Schema>
       <xs: simpleType name = "EmailID">
          <xs: restriction base = "xs: string">
             <xs: pattern value = "a-z A-z 0-9 . @" >
          </xs: restriction>
       </xs: simpleType>
```
Atomic data type 1

```
       <xs: simpleType name = "mbno">
        <xs: restriction base = "xs: int">
         <xs: length value = 10 >
        </xs: restriction>
       </xs: simpleType>.
```
Atomic data type 2

```
  <xs: SimpleType name = "MyUnion">
   <xs: union memberType = "emailID mbno"/>
   </xs: SimpleType>
 <xs: Element name = "custInfo" type = "MyUnion"/>
 </xs: schema>
```

<custInfo>1234567890 </custInfo>

<custInfo> abcdefgh23@gmail.com < /custInfo>

Parsers:-

Parser is used to convert xml document into object oriented form.

→ parser is a type of software

→ 3 types of parsers 1) DOM parser (constructed by using JAXP API)
2) SAX parser (Simple API for xml parser)
3) STAX parser (streaming API for xml parser)

→ DOM parser works by using tree based structure

→ operations performed by the parser are
1) read operation
2) parse "
3) validate operation

→ By using factory methods like newInstance(), newDocumentBuilder() to create instances for abstract classes & interfaces also which are in factory classes.

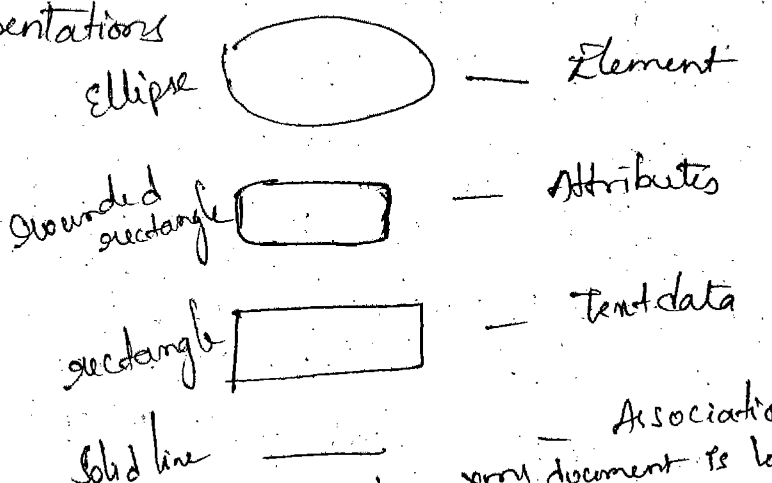
→ JAXP API consist Interfaces, classes, methods, constants.

→ In DOM parser steps required are
* creation of DocumentBuilderFactory object
* creation of DocumentBuilder object
* parse the xml document.

→ DOM parser creates tree based structures (xml sparse Tree)

→ 12 types of nodes are available.

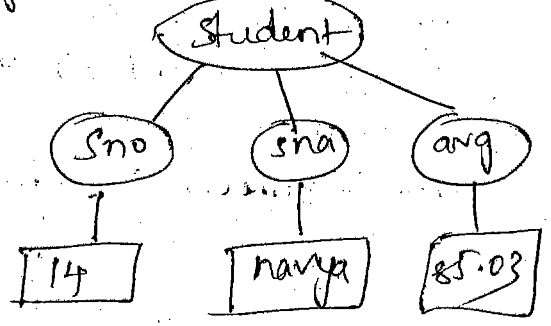
→ Representations



(At a time xml document is loaded into memory & sparse tree creation in memory)

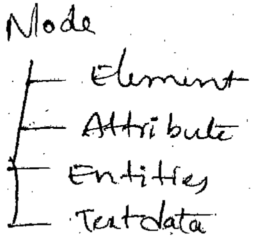
XML document

<Student>
<sno>14</sno>
<sna>narya</sna>
<avg>85.03</avg>
</Student>



→ DOM parse needs more mmory if the XML document size is more which is the drawback of DOM parser.

→ org.w3c.dom.Node
↳ interface.



→ 12 types of Nodes are

(priority)

1. Element ... 1
2. Attr -2
3. Entity -6
4. Text -3
5. Entity Reference -5
6. Document -9
7. Document Type -10
8. Document Fragment -11
9. processing Instruction -7
10. Comment -8
11. CDATA Section -4
12. NOTATION -12

→ DOM parser performs read and write operations.

→ DOM parser allows top to bottom & bottom to top retrieval.

→ getDocumentElement( ) ~~return~~ is used to retrieve root Element.

→ get ~~Element~~ Node Name( ) is used to print the Element.

Eg:
```
import javax.xml.parsers.*;
import org.w3c.dom.*;
import java.io.IOEXCEPTION;
class ExDomParser{
public static void main(String[] args) throws
 ParserConfigurationException{
DocumentBuilderFactory dbf = DocumentBuilderFactory.
                                     newInstance()

DocumentBuilder db = dbf.newDocumentBuilder()
try{
 Document d = db.parse("student.xml");
 Element e = d.getDocumentElement( );
System.out.println("root node :" +e.get~~Element~~NodeName(
NodeList nl = e.getChildNode();
 for(i=0; i<nl.getLength(); i++)
 {
  Node n= nl.item(i);
  if(n.getNodeType()==n.ELEMENT_NODE)
  {
  s.o.p("node.name is:"+n.getNodeName());
  s.o.p("text is:"+n.getTextContent());
  }
 }
}
 Catch(Exception ex)
 {
 ex.printStackTrace();
}}}
```

SAX Parser :-

Drawbacks in Dom parser :-

1. If xml document size is big, sparse tree requires more space in memory.
2. DOM parser is slow

DOM	SAX
1. Document Object Model parser	1. Simple API for Xml parser
2. Follows tree based Structure	2. Follows Event based model
3. Allows read & write operations	3. Allows only read operations
4. Allows Sequential & random accessing.	4. Allows only sequential access that too top to bottom
5. It occupies more memory if the xml document size is big.	5. It occupies less amount of mmry compared to DOM parser.
6. DOM is slow in performance	6. SAX parser is faster

ContentHandler (Interface)

methods - Events

* void startDocument()
* " endDocument()
* " startElement()
* " EndElement()
* " characters()
* " startPrefixMapping()
* " EndPrefixMapping()
* " skippedEntity()
* " ignoreWhiteSpaces()

Attributes (Interface)

* getAttribute()
* getLength()
* getValue(int index)
* getValue(String, String)

8/1/18

UNIT- II

AJAX

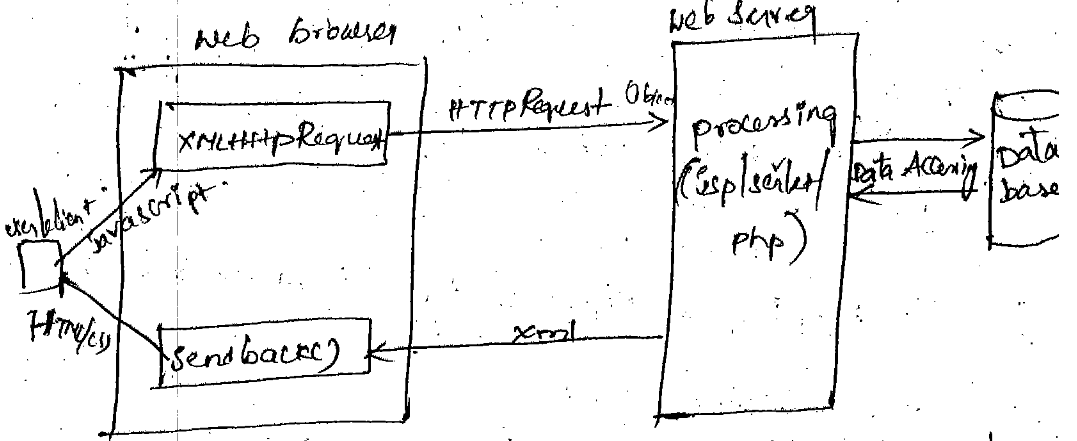
→ Asynchronous Javascript And XML

→ By using AJAX we can modify a portion of the web page ~~inorder~~ instead ~~of~~ reload the entire web page.

→ Applications of AJAX are facebook, gmail, google maps, google etc.,

→ AJAX is not a technology it is a technique.

→ It is a Combination of HTML, CSS, JavaScript, XML, DOM & XMLHttpRequest Object'S.



→ To establish the Connection b/w webbrowser & web server we need to calla Open() which is overloaded method and is present in XMLHttpRequest.

Open(method, url, async)
open (method, url)

by default false

Open(method, url, asy, username psnwd)
Open (method, url, asyn, username)

method can be = GET/POST / HEAD

→ Send(content) method is used to send the request to the Server. overloaded method send ( )

→ Set RequestHeader (header, value) — It adds request headers

→ status & StatusText are the attributes of XMLHttpRequest

status = 404 — unsuccessful
= 200 — Successful.

statusText (404) — "Not found"
(200) — "OK".

→ abort( ) method cancels the current request.

→ getResponseHeader (headerName) – returns the value of the specified header.

Properties of ~~XMLHttp~~ XMLHttpRequest Objects

→ onreadystatechange is the attribute of XMLHttpRequest
An Event handler for an Event that fires at Every state Change.

→ readyState is the attribute of XMLHttpRequest & having 5 states starts from 0 – 4

0 – XMLHttpRequest object Created, request not initialized

1 – open( ) method called

2 – Send( ) " " , HEADERS_RECEIVED

3 – request is in process

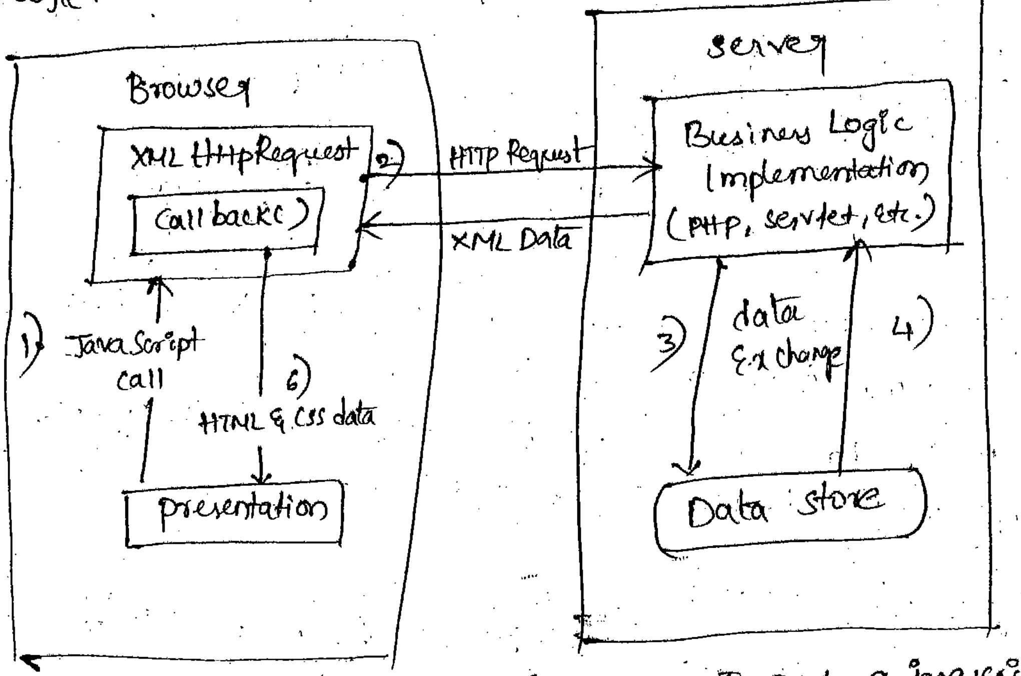
4 – request is completed

→ responseText – return response as text.

→ response XML – return response as XML

How AJAX works? :-

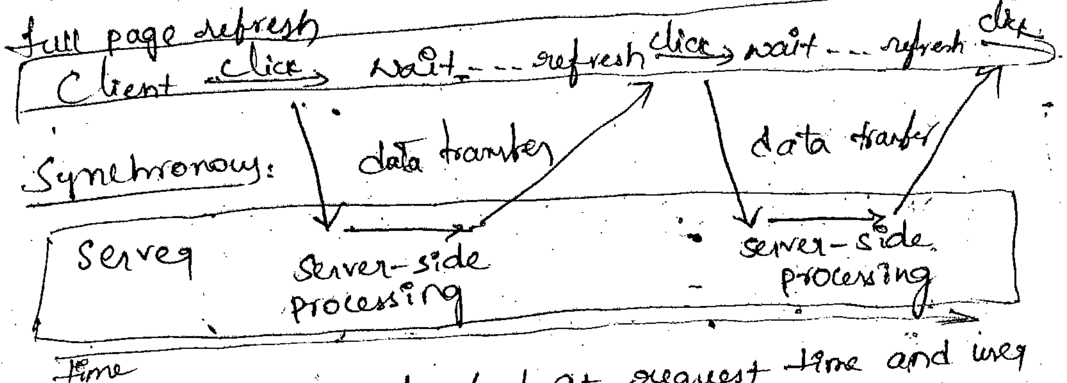
AJAX Communicates with the server using XMLHttpRequest Object



1) User sends a request from the UI and a javascript
call goes to XMLHttpRequest Object
2) HTTP Request is sent to the server by XMLHttpRequest
Object
3) Server interacts with the database using JSP, PHP,
Servlet, Asp.net Etc.
4) Data is retrieved
5) Server sends XML data or JSON data to the
XMLHttpRequest callback function
6) HTML and CSS data is displayed on the browser.

Synchronous mode :

full page refresh



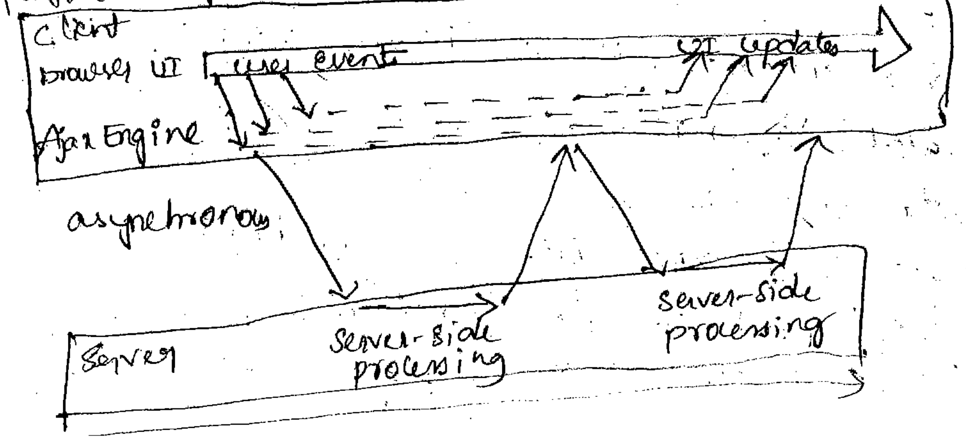
→ Full page is refreshed at request time and user is blocked until request completes.

→ A synchronous request blocks the client until operation completes i.e.., browser is unresponsive.

Asynchronous mode :

An asynchronous request doesn't block the client i.e.., browser is responsive. At ~~that time~~, user can perform another operations also.

Partial UI updates



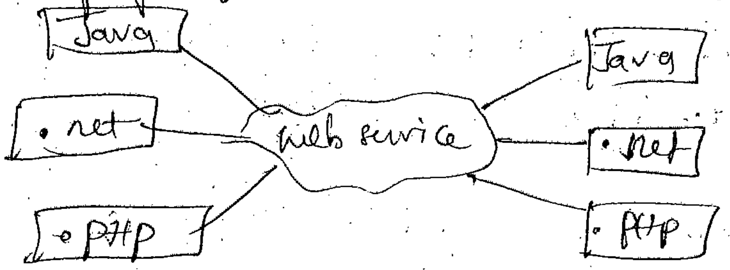
→ Full page is not refreshed at request time and user gets response from ajax Engine.

Eg:
```
<html>
  <head>
   <script type = "text/javascript">
            showHint(str)
    function changeContent() {   if(str.length==0) {
                                  document.getElementById("
 else { var x = new XMLHttpRequest();   innerHTML = "";
                                        return; }
      x.open("GET", "login.php", true);
      x.send();
      x.onreadystatechange = function() {
   if(this.readystate == 4 && this.status==200)
   {
   document.getElementById("change").
              innerHTML= this.responseText;

     }
   } }
  </script>
 </head>
 <body>
<h1> Ajax Example </h1>
<p id="change"> this is Ajax </p>
<input type= "button" onclick="ChangeContent()"
                onkeyup= "showHint()"
           value = "change content" />
<p> Suggestions: <span id="change"> </span> </p>
 </body>
</html>
```

Web Services:—

* Is a Client server application or application component for communication
* method of communication b/w two devices over network.
* is a s/w system for interoperable machine to machine communication
* is a collection of standards or protocols for Exchanging information b/w two devices or application



→ web services are of 2 types

1. SOP (simple Object Access protocol) — is a protocol
2. RESTful web service — is a architecture

→ web components are of 3 types

* SOAP
* WSDL — (web services description language)
* UDDI — (universal description discovery Integration)

SOAP: XML — based protocol for accessing web services

WSDL: XML document containing information about web services. such as method name, method parameter.

→ part of UDDI

UDDI: XML based framework for describing, discovering & integrating web services.

Types:—

1. SOAP:—

→ Acronym for Simple object Access protocol.

→ SOAP is a protocol.

→ Xml based web service

Advantage
→ SOAP is platform independent & also language independent

→ It is slower because it takes time when converting.

Advantage
→ SOAP provides its own Security namely WS security.

Drawbacks:

→ Slow since it is using Xml based which needs conversion

→ It uses WSDL and doesn't have any other mechanism to discover the service.

Eg on SOAP:

Request:

```
Post/item HTTP/1.1
HOST: 189.123.345.239
Content-Type: application/soap+xml; charSet=UTF-8
Content_Length: 200
<?xml version="1.0"?>
<Soap: Envelope (who sends, what it contains, Is it mandatory or not)
Xmlns: soap = "http:// www.W3c.org/2001/12/Soap-Envelope"
Soap: Encoding Style = "http:// www.N3c.org/2001/12/soap-Encoding">
        ↳ (define rule for Exchanging instance of application)
<Soap Body Xmlns: m = "http:// www.abc.org/item">
<m: GetPrice>
<m: name> shoes </m: name>
</m: GetPrice>
</Soap: Body>
</soap: Envelope>.
```

Response:

```
HTTP:/1.1 200 OK
Content-Type: application/Soap+xml; CharSet=UTF-8
Content_Length: nnn
<? xml version="1.0"?>
<soap: Envelope
xmlns: soap = "http://www.w3c.org/2001/12/Soap-Envelope"
Soap: Encoding Style = "http:// www.N3c.org/2001/12/soap_Encoding">
<soap: Body xmlns: m = "http:// www.abc.org/item">
    <m: GetPrice Response>
    <m: price> 300.00 </m: price>
    </m: GetPrice Response>
</soap: Body>
</soap: Envelope>
```

UNIT-4

PHP.

Php stands for Hypertext preprocessor. Lerdorf

→Introduced in the year 1994 by Rasmus Lerdorf

→ PHP is a server side scripting language.

→ PHP is an interpreted language ie., no need of compilation.

→ PHP is an object-oriented scripting language.

→ PHP is case sensitive.

Features:-

1. Performance :- script written in PHP Executes much faster than those scripts such as JSP & ASP.

2. Open-Source software:- PHP source code is available on the web, you can developed all the version of PHP according to requirement.

3. platform Independent:- PHP are available for WINDOWS, MAC, LINUX & UNIX OS.

4. Compatibility :- Compatible with almost all local servers like Apache, IIS Etc.,

5. Embedded :- PHP code can be Easily Embedded within HTML tags and scripts.

→ PHP ~~are~~ is used to interact with databases.

→ PHP performs system operations like read, writing, open, close, creating Etc.,

→ Here we are using AMP software (Apache, MYSQL, PHP)
server database s/w

WAMP — for windows
LAMP — " LINUX
MAMP — " Mac
SAMP — " Solaris
FAMP — " FreeBSD
XAMPP — (Cross, Apache, MYSQL, PHP, Perl)
X A M P P

Syntax:

```
<?php
    //code
    ?>
```

Eg:

```
<html>
<body>
 <?php
   Echo "Hello!";
  ?>
</body>
</html>
```

→ Echo is a coding standard. It is a language construct not a function.

→ If we want to use more than one parameters, it is required to use paranthesis.

Eg void Echo(string $arg1 [, string $---])

→ Declaring a variable is

$ variablename.

PHP Print :— It is a language construct~~or~~. It works like an Echo

→ print returns integer value (int) but Echo doesn't return any value (void).

Eg: int print $a+$b;

Variables :—

→ Variables are declared using dollar symbol ($)

→ PHP doesn't force to specify data type ie.., loosely coupled.

→ First charactey should be a alphabet (or) underscore.

→ " " shouldn't " " number.

→ Identifier is case sensitive.

→ Doesn't use keywords as Variables.

Eg: $a; $a=10;

Syn $variablename;

Eg:
```
<?php
  $X="Welcom to php variable";
  $Y=10;
  $Z=20;
  Echo $X."<br>";
  print($Y+$Z);
?>
```

Data types :—

→ php data types are used to hold the different values like integers, strings, float, boolean etc..,

→ php contains 8 different data types categorized into 3 groups :— 1) scalar types
2) compound "
3) special "

Scalar types:—
Integer, string, float, boolean are scalar data types.

Compound types:—
Array and object are compound types.

Special types:—
RESOURCE, NULL are special types.

Constants :—

→ Constant values are never changed during program execution.

→ To declare constants we use

1. define function
2. const keyword.

→ Built in constants (magic constants) are begins & Ends with double underscore's

- __LINE__ (current line no)
- __FILE__ (full path & filename)
- __DIR__ (file directory path)
- __FUNCTION__ (fun name)
- __CLASS__ (class n)
- __METHOD__
- __NAMESPACE__ (name of current namespace)
- __TRAIT__ (name of the trait)

define function :—

Syn: define (NAME, value, case-insensitive);
↳ default false (boolean)

→ Name of the constant

→ value of the constant.

→ so if boolean value is false it is case-sensitive.

Eg: define("AGE", 28, true); define(College, "KHIT");

const Keyword :—

Syn :- const const_name = value;

Eg: const a=10;

Operators :—

Operator is a symbol it performs operation on the operands.

→ operators are in generally classified into 3 types

1. Unary operator : which performs operation only on one operand. Eg $a++
2. Binary operators : which performs operation on two operands. Eg: $a + $b
3. Ternary operators : / Conditional operator.

    $a ? $b : $c;

    $a > $b ? "a is true" : "b is true";

4. Increment & Decrement operators : --, ++

Arithmetic Operators :— +, -, *, /, %, **(PHP 5.6) ↳ Exponentiation

Logical operators :— ~~!~~, &&, ||, and, or, xor

Relational operators :— >, <, >=, <=, ==, !=, ===, <>(not equal) checks type.

Bitwise operators :— &, |, ^ ↳(xor), ~ ↳(NOT), <<, >>

1 0 - 1
0 - 1

Eg:
```
<?php
$a = 10;
$b = 5;
($a > $b) ? Echo "a is big" : Echo "b is big";
?>
```

Other operators :—

1) string operators (., .=)
2) Array operators.
3) Type operators. (instanceof)
4) Execution operators (back ticks (``)) Eg: $a = 'ls -l';
5) Error Control operators. (@)

Comments in php :- Documentation is an Explanationary part.

// — Single line comment.

— " " "

/* - - - - */ - Multiple line documentation.

Conditional statements :-

1. Simple if
2. If ... Else
3. Nested if
4. Else...if laddeq

Eg:- **Operators**,

```
<?php
 $a=2;
 $b=3;
Echo "sum is:" ($a+$b);
Echo "<br>";
Echo "sum is:" ($a-$b);
Echo "<br>";
 ?>
```

String operators :-

. → Concatenation

.= → Complex.

Eg:
```
<?php
  $a = "KHIT";
  $b = "CSE";
   $a. = $b;
 # $a = $a.$b;
 print $a."<br>";
 ?>
```

Eg:- Comparision Operators

```
<?php
  $a=10;
  $b=10.2;
  if($a===$b)
  Echo "a and b are same";
  Else
  print "a and b are not same";
  ?>
```

Eg:-

```
<?php
    $b=2;
    Echo $b<<1;   ← (no.of bits)
    ?>
```

Arrays:-

1. Indexed
2. Associated
3. Multidimensional

Conditional statements Examples:-

Eg: (simple if)

```
<?php
    $x = 50;
    if($x>30)
      Print $x." is big";
      print "conditional statements";
  ?>
```

Eg: if ... Else

```
<?php
    $x = 50;
    if($x>20)
      Print $x."is big";
    Else
      print $x."is not big";
?>
```

Eg: Nested if

```
<?php
  $x = 40;
  if($x>20)
  {
      if($x>18)
        print "Eligible for vote";
      Else
        print "not eligible";
  }
?>
```

Eg: Else... if ladder.

```
<?php
  $x=20;
  $y=10;
  if($x==$y)
    Print "Both are Equal";
  Else if($x>$y)
    Print $x." is big";
  Else
    Print $y." is big";

?>
```

Switch :-

Syn:
```
switch($case Variable)
{
  Case 'name'/value : statement1;
                      |
                      Statement 2;
                      break;
            |
            |
  Case 'name'/value : statement1;
                      Statement 2;
                      |
                      break;
  default : statement;
}
```

Eg :
```
$a=2;
switch($a){
Case 1 : Echo "ONE";
         break;
Case 2 : Echo "Two";
         break;
Case 3 : Echo "Three";
         break;
default : Echo "Invalid";
}
```

Control statements :-

1. While loop
2. do... While loop
3. for loop
4. for... Each

Eg: (While)
```
<?php
  $a=10;
   While($a>0)
   {
    print $a. "<br>";
    $a++;
   }
?>
```

Eg : do... While

```
<?php
 $a=10;
 do
 {
  print $a. "<br>";
  $a++;
 }While($a>0);
?>
```

Eg: for

```
<?php
  $a = 2;
  for($i=1; $i<=10;$i++){
   Echo $a."*".$i."=".($a*$i);
   Echo "<br>";
?> }
```

Eg:—

Syn: foreach($arrayname as $variable)

```
{
   ≡
}
```

Eg: <?php

```
  $sub = array("NT", "DAA", "DWDM");
  foreach($sub as $i)
    Echo $i."<br>";
  ?>
```

Break:—

Whenever a break appears it comes out from the current working block.

```
<?php
 $sub = array("NT","DAA","BWDM", "CN");
 foreach($sub as $K)
 {
    if($K == "CN")
      break;
    Echo $k;
  }
 ?>
```

Continue:—

Whenever continue appear the control transfer to the starting of the block.

Eg: <?php

```
 $sub=array("NT", "DAA", "BNDM", "CN","SE");
 foreach($sub as $K)
 {
    if($k=="CN")
      Continue;
    echo $k;
  }
 ?>
```

Functions :—

- → Function is a part of program or subprogram
- → Code reusability, Error free code available & development time reduced.
- → Functions have name and we may call by that name
- → functions can have parameters.
- → Functions may return values.
- → keyword used to define is "function"
- → Functions are of 2 types — Built in / User defined.

Built-in :—
The functions which are already created by php developers are string handling, array and mathematical function.

Userdefined :—
The functions which are defined by the user are called user defined functions.

Eg:
```
<?php
function fact($n)
   {   $f=1;
     while($n>0)
       {
     $f=$f*fact($n-1); // recursive
       }
   } return $f;

   Echo fact(5);
?>
```

```
$f *= $n; // non-
$n--;       recursive
```

→ Various ways to declare a function

1) passing parameters to the function.
2) Default arguments to the function
3) variable no:of arguments
4) Recursive function.
5) Function within the function

Passing Parameters to the function :—

Syntax : (passing parameter without return value)
```
function funname(parameter list)
{
  ≡
}
```

Passing parameter with return value :
```
<?php
function fun($x, $y)
 {
  $z = $x+$y;
  return $z;
 }
 Echo sum(10,20);
?>
```

Default argument to the function:

Eg:
```
<? php
   function sum($x,$y=10)
   {
      $z=$x+$y;
      return $z;
   }
   Echo sum(2);
   echo sum(20,30);
?>
```

Variable no.of arguments:

```
<? php
 function sum(...$x)
 {
    $sum=0;
  foreach($x as $k){
     $sum+=$k;
    return $sum;
 }
   Echo sum(2);
   Echo sum(2,3);
   Echo sum(2,3,5);
?>
```

0 1
X [2|3]

$sum = 0+2
$ = 2+3

Recursive function:

```
<? php
 function fact($n)
 {
  if(n==0 || n==1)
   return 1;
  else
  return $n * fact($n-1);
 }
?>
```

Function within the function:

```
<? php
 function Sum($x, $y)
 {
   $z = $x +$y;
   return $z;
  function diff($x, $y){
    $z = $x - $y;
    return $z;
   }
   diff(5,3);
 }
   sum(2,3);
?>
```

Array:-

Array is a collection of similar data type elements.

Creation of array:

1. By using array function.

   Syn: array(parameters).

Types of arrays:-

1. Indexed array —creation→ array( )
   → $ ~~Variable~~ array name [index] = value.
2. Associated array (Key ⇒ value)
3. Multidimensional "

Eg:-
```
<? php
    $ Sub = array("Name" => "Navya", "Age" => 20);
    Echo  $sub["Name"];
    ?>
```

(or)

```
    <? php
  $sub= array("WT" => 40, "OWDM" => 60, "CN" => 50);
   foreach($sub as $k)
     Echo  $K . "<br>";
    ?>
```

Multidimensional

Eg:-
```
<?php
  $sub= array(
            array(1, "WT", 35),
            array(2, "CN", 40),
            array(3, "DAA", 45)
         );
  for($row=0; $row<3; $row++) {
  for($col=0; $col<3; $col++) {
   Echo  $sub[$row][$col] . " ";
  }
    Echo  "<br>";
  }
 ?>
```

Array Handling functions:-

1. array( )
2. array_chunck($array, no:of chuncks) - divides array

Eg: $sub= array("y1", "y2", "y3", "y4");

```
        array_chunck($sub, 2);
     $sub[0]
            $sub[0] = "y1"
            $sub[1] = "y2"
     $sub[1]
            $sub[0] = "y3"
            $sub[1] = "y4"
```

3. Count($array) - Count no:of Elements

Eg: $a = array(1,2,3,4,5);

Count($a); o/p: 5

4. sort($array) - arrange elements in sorted order.

```
<?php
    $a= array( 5,4,3,2,1);
   $b= sort( $a);    o/p: 1  2  3  4  5
   forEach($b as $k)
    Echo $k;
  ?>
```

5. array_reverse($array) - reverse the elements

```
 <? php
    $a = array ( 10,50,90,45,2);
     $b= array_reverse($a);
     forEach($b as $k)
       Echo $k." ";
    ?>
                    o/p: 2  45  90  50  10
```

6. array_search($Element, $array);

```
  $a = array(10, 50,90,45, 2);
   Echo array_search( 50,$a);
```

7. array_intersect ($array1, $array2):

```
<?php
 $a= array(10,50,90,45,2);
 $b= array(5,9,85,50,100,10,900);
 $c = array_intersect($a,$b);
  forEach($c as $k)
   Echo $k." ";
  ?>
```

8. array_change_key_case($array, CASE_LOWER):
CASE_UPPER

```
Eg: <? php
      $a = array("Age"=>20, "NAME"=>"NAVYA");
      $b= array_change_key_case($a, CASE_LOWER);
      forEach($b as $k)
       Echo $k." ";
     ?>
```

Sorted functions:—

1. sort(): sort arrays in ascending order

```
 <?php
  $cars = array("volvo", "BMW", "Toyota");
   sort($cars);
   $clength= count($cars);
  for( $x=0; $x<$clength; $x++){
```

```
Echo $cars[$x];
Echo "<br>";
}
?>
```

rsort(): sort arrays in descending order.

```
<?php
$x= array(1,2,3,4,5);
$y= rsort($x);
forEach($y as $k)
  Echo $k;
?>
```

o/p: 5 4 3 2 1

asort(): sort associative array in ascending order, according to the value.

```
<?php
 $sub= array("WT"=> 50, "DAA" => 40);
  ~~sort~~ asort($age);
  forEach($age as $x => $x_value){
    Echo "key=" . $x . ", value=". $x_value;
    Echo "<br>";
  }
?>
```

ksort(): sort associative array in ascending order, according to the key.

```
<?php
$age = array("xyz"=> 20, "abc" => 15, "pqr"=> 30)
 ksort($age);
foreach($age as $x => $x_value)
{
 echo "key=" . $x . ", value=" . $x_value;
 Echo <br>";
}
?>
```

o/p: key=abc value=15
key=pqr value=30
key=xyz value=20

arsort(): Sort associative arrays in descending order, according to the value.

```
<?php
$age=array("xyz"=>20,"abc"=> 15,"pqr"=>30);
 arsort($age);
foreach($age as $x =>$x_value)
{
Echo "key =". $x . ", value =". $x_value;
Echo "<br>";
}
?>
```

o/p: key=pqr value=30
key=xyz value=20
key=abc value=15

krsort(): sort associative array in descending order, according to the key.

Super globals in php:-

1. $GLOBALS
2. $_SERVER
3. $_POST
4. $_GET
5. $_FILES
6. $_ENV
7. $_COOKIE
8. $_SESSION
9. $_REQUEST

→$GLOBALS is a php super global variable which is used to access global variables from anywhere in the php script (within the function (or) method).

→ php stores all global variables in an array called $GLOBALS['index'], here index holds variable name.

Eg:-
```
<?php
        $a=10;
        $b=20;
        function sum()
        {
$GLOBALS['$c'] = $a + $b;
        }
          Sum();
        Echo $c;
      ?>
```

→ In the above Example $c is declared within the function but can be accessed outside the function with the help of $GLOBALS.

$_POST:-

login.html:
```
   <html>
     <head>
        <title> Login </title>
      </head>
     <body>
                                          GET
       <form name="login"   method="POST"
                         action=" login.php">
   Name:<input type="text"  name="uid">
  password:<input type="password"  name="pwd">
     <input type="submit"  value="submit">
   </form>
  </body>
</html>
```

login.php:-

```
<?php
                   GET
$name = $_POST['UID'];
                   GET
$password = $_POST['psnd'];
Echo "User name is :".$name;
Echo "password · is :",$psnd;
?>
```

→ $_POST is widely used to collect form data after submitting an html form with method = "POST".

$_GET:-

$_GET is widely used to collect form data after submitting an html form with method = "GET".

$_SERVER:-

→ Total 23 to 24 values are passed to $_SERVER (super global ~~const~~

Server softwares are:- Tom cat

IIS

JBoss

weblogic

webspheare.

→ values passed to the $_SERVER super global are:

1. $_SERVER['PHP_SELF'] - It returns file name of the currently executing script.

2. $_SERVER['SERVER_NAME'] - It returns name of the host server.

3. " ['SERVER_ADDR'] - It returns Ip address of the host server.

4. " ['SERVER_SOFTWARE'] - It returns server identification string.
(i.e Tomcat, IIs, JBoss Etc..)

Eg: Apachae Tomcat/2.2.24

5. " ['SERVER_PROTOCOL'] - It returns name and version of the information protocol.

Eg: http://1.1

6. " ['SERVER_ADMIN'] - It returns the value given to the server admin in the web server.

7. " ['SERVER_PORT'] - It returns the port number of the server machine.

$_COOKIE:-

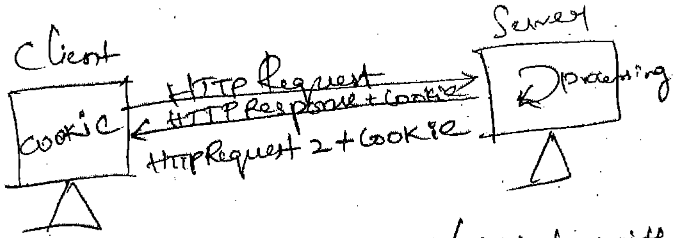
→ Cookie is created at server side and stored at client side/client web browser.

→ Cookie contains key value pair.

→ php cookie is a small piece of information.

→ It is used to recognize the user.

→ Each time when client sends request to the server cookie is embedded with request.



→ cookies are created/setted with setcookie() function.

Eg:- boolean setcookie("csea");

Note:- How to acces cookie?

To retrieve/acces cookie we are using $_COOKIE. It is a super global of php.

Eg:-
```
<?php
 setcookie("user","csea");
?>
```

Eg. html:
```
<html>
 <body>
  <?php
  if(!(isset($_COOKIE['user']))
   Echo "cookie is not set";
  Else
   Echo "Cookie name is".
          $_COOKIE['user'];
  ?>
 </body>
</html>
```

→ set Cookie() must appear before the html tag.

$_SESSION:-

→ php session is used to store and pass information from one web page to another web page temporarily (upto close the website)

→ php session Technique is widly used in shopping webits where we need to store and pass cart information.

Eg:- product name, product code, user name, price etc.,

→ php session creates unique userid for each browser. To recognize the user and avoid conflicts b/w multiple browsers.

How to start session?

To start the session we are using the following

SESSION_START. — It starts new or resumes existing session. It returns existing session if session is already created. If session is not created it creates a new session.

syn:- boolean session_start(void)

→ $_SESSION is an associative array it contains all session variables. It is used to set and get session variable values.

Eg:- $_SESSION['user'] = "csea";

→ Get session values

Echo $_SESSION['user'];

Note:- To destroy all Session Variables session_destroy( ) function is used.

$_FILES:—

→ It contains all information of files.

→ With the help of $_FILES we get file name, file type, file size and temporary file name and errors associated with the file name.

Eg:

```
$_FILES['filename']['SIZE']~~['TYPE']~~;
$_FILES[file_name]['TYPE'];
$_FILES[file_name]['temp_file']
```

$_ENV:-

→ It is used to return the Environmental variables from the web server.

Eg:-

```
<?php
    Echo $_ENV['user'];
?>
```

$_REQUEST:—

Data Base Connectivity:—

Working with DataBases:—

If we want to ~~Connect~~ work with database we need to establish connection to the database.

→ mysql_connect( ) is a ~~method~~/function used to connect to the mysql database but it is depricated.

→ By using the following ~~methods~~/functions we may connect to the mysql database from php 5.5 version onwards

1. mysqli_connect( )
2. PDO::_connect( )

mysqli_connect( ):- This ~~method~~ function returns database resource if connection is established succesfully. Otherwise It returns NULL.

→ For this function we may pass arguments like servername, username, password, databasename.

Eg:-

```
<?php
    $server = "localhost";
    $username = "KAit";
    $password = "csea";
  $con = mysqli_connect($server, $username, $password);
    if(!$con) {
  die("connection is not established". mysqli_error());
```

```
Echo "connection successfully established";
    mysqli_close($con);
?>
```

mysqli_close() :- This function is used to disconnect with mysql database if connection is successfully closed otherwise it returns false.

Syn:- boolean mysqli_close(resource);

Eg:- mysqli_close($con);

mysqli_query():-

Create database in mysql

Since php 4.3 mysql_create_db() function is depricated.

→ ~~After 4.3~~ From php 4.3 onwards two functions are available for creating database

1. mysqli_query()
2. PDO::_query()

mysqli_query() is used to execute the query on the database.

→ It returns true if query executed succesfully orelse it returns false.

→ For this function we need to pass two parameters

1. Database connection.
2. Executing query.

Eg:-
```
<?php
    $serveq = "localhost";
    $username = "KhIt";
    $password = "csea";
$con=mysqli_connect($serveq, $username, $password);
  if(!$con){
die("Connection not Established".mysqli_error());
  }
  Echo "Connection established";
  $query = "CREATE database CSEA;";
  if(mysqli_query($con, $query)){
   Echo "DataBase created";
   }
  Else
   echo "DataBase not created". mysqli_error();
  mysqli_close($con);
?>
```

Creating table in mysql using php :—

```
<?php
    $server = "localhost";
    $username = "khit";
    $password = "cse";
      $db = "CSEA";
$con= mysqli_connect($server, $username, $password,
                                          $db);
    if(!con){
 die("connection not established".mysqli_error());
    }
 Echo "connection Established";
 $query= "Create table student(sno INT,
                     sna VARCHAR2(20));";
 if(mysqli_query($con, $query))
 {
   Echo "Table Created";
  }
 else
   Echo "Not Created";
  mysqli_close($con);
?>
```

→ From php 5.5 ~~onwards~~ version mysql_query() function is depricated.

→ In this version alternatively two functions are introduced

1. mysqli_query()
2. PDO::_query()

Inserting a record into a table in mysql using php :—

```
 $query= "insert into student values(514,"ravya"
                                           ";
  if(mysqli_query($con, $query))
  {
   Echo "1 row Created";
  }
  Else
     Echo "not Created". mysqli_Error();
```

Working with oracle database :—

→ For connection we use oci_connect()

→ To know the Error we use oci_Error()

→ For preparing an oracle statement for Executi

we use oci_parse()

Eg: $x=oci_parse($con, 'select * from student');

→ To Execute a statement we use oci_Execute(

Eg:- oci_Execute($x);

→ To close the connection we use oci_close()

Working with db2:-

→ For connection db2_connect( );

→ db2_autocommit() returns or sets the AUTOCOMMIT state for a database connection.

* Parameters passed are connection and value.

* Values are DB2_AUTOCOMMIT_OFF – Turns off
" " –ON – " on

→ db2_pconnect() – returns a persistent connection to database.

→ For Executing query db2_exec( );
Eg:- $r = db2_exec($con, 'select * from student");

→ For closing the connection db2_close();

UNIT-V

PERL.

→ PERL is a server side scripting language.

→ It is cross-platform independent.

→ support any database like mysql, sql, db2 etc

→ supports frame works.

→ starting version is 1.0 in 1987 by LARRY WALL.

→ Latest version is perl 5.4.3 introduced in 2016.

→ perl interpreter is used to convert the perl code into unicode.

Features:

1. Platform independent
2. supports by all databases
3. Used to create frameworks
4. Interpreter is used
5. supports unicode
6. supports both pop & oop.

Eg:

```
my $x;
   $x=10;
my $y=20;
my $z=$x+$y;
say("sum is → $z");
```

Datatypes :— 3 types

1. $ Scalars — stores only one value
2. @ Arrays — stores any no:of values of same type
3. % Hashes. — store associated arrays.

→ $, @, % symbols doesn't use in between identifiers.

→ say() and print() both are used for display message on console.

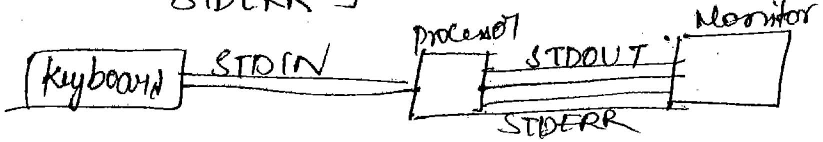
→ But the difference is say() by default provides '\n'.

```
print("welcome");       say("welcome");
print ("To");           say("To");
print("perl");          say("perl");

welcomeToperl              welcome
                             To
                            perl.
```

→ Default streams provides by any programming/ Scripting languages are

STDIN → source (keyboard)
STDOUT } Monitor
STDERR



→ If we want to read data from keyboard we use <STDIN>

Eg:—
```
my $x;
   say ("Enter x value");
     $x = <STDIN>; // $x = 23
   print("value of x is → $x");
```

Perl Special literals :—

1. __FILE__ (represents current file name)
2. __LINE__ (represents current line no).
3. __PACKAGE__ (" " executing package)

Operators :—

1. Numeric operators(+, −, *, /, %)
2. String operator/concatenation operator(.)
3. Logical operators(&&, ||, !)
4. Bitwise operators(&, |, ^, ~, <<, >>, <<=, >>=, &=, |=)
5. Special operators(++)
6. Comparision operators (>, <, <=, >=, ==, !=)
7. Assignment operators (=, +=, −=, *=, /=, %=, &=, |=)

```
Eg:- my $x="Naya" (or)
     my $x;
     my $y; ~~say~~
*     $x=<STDIN>; // $x=< >
      $Y=<STDIN>; // $y=< >
     ~~Say("Before c.~~ say("Before Concatenation $x & $y");
      say("After Concatenation $x.$y");
```

```
Eg:- my $x;
     my $y;
     $x=< >;
     $y=< >;
     if($x>$y)
     say("x is big");
     Else
      say("y is big");
```

```
Eg:- my $x;
     my $y;
     $x=< >;
```

```
say("left shift of x is -> $x<<3");
say("right shift of y is -> $y>>4");
```

Conditional statements:-

1. simple if
2. if... Else
3. Nested if
4. Else... if ladder.

```
Eg:- my $n;
     Say("Enter n value");
     $n=<STDIN>;
     if($n%2==0)
      say("$n is Even");
     Else
      say("$n is odd");
```

```
Eg:- my $n;
     my $m;
     say("Enter n value");
     $n=<STDIN>;
     say("Enter m value");
```

```
if(n>=80 && m<=100) {
   if(n>90)
    say("First class");
   Else
    say ("second class");
  }
  Else
   say ("Third class");
```

Else...if :-

```
   if(Condition){
     statements;
    }
   Elsif (Condition) {
     statements;
   }
  else {
    statements;
   }
```

Loops in perl :-

1. while loop
2. do...while "
3. for "
4. for each "

5. until loop

Eg:-

```
my $x=10;
   my $i=0;
    while($x>10){
     say( value of x is: $x);
     $ i++;
    }
      (or)
```

factor of 2 :-

```
     my $x=0;
     my $n;
      $n=< >;
      while($n>0) {
       if($n%2==0)
        $n/=2;
       Else
       {
        say("Not a factor of 2");
        $x++;
        break;
       }
     }
     if ($x==0)
       print("factor of 2");
```

do... while:-

```
my $sum=0;
my $n;
$n=< >;
$m= $n;
do
{
  $r= $n%10;
  for($f=1; $r>1; $r-=1)
     $f*=$r;
  $sum+=$f;
  $n/=10;
}while($n>0);
if( $sum==$n)
 say("Strong number");
Else
 say("not");
```

until :

prints the stmnts if the condition is false.

```
my $x=20;
my $i=5;
until($x<$i){
  say("false");
  i++;
}
```

foreach:-

syn

```
foreach $variable (@array){
   statements;
}
```

Eg: Perl program to display array elements using foreach loop.

```
my $x;
@number=(501,502,503);
foreach $x (@number)
   print "$x \n";
```

Arrays:-

syn:-

```
@arrayname=(element1, element2, .... elementn);
```

Eg:-

```
@numberarray = (10,15,20,25,30);
```

10	15	20	25	30

How to access array Elements:-

Eg:-

```
@names =("KHIT", "CSE", "JNTUK");
    for(i=0; i<=2 ; i++)
     print "#names[i] \n";
```

Array size (or) length :-

```
$ size= @names;  (3)
```

To know the max-index we are using $#arrayname.

```
$max_index = $#names;  (2)
```

Another way to know the length is

```
$length = $#names+1;  (3)
```

Array functions :-

→ To add and remove Elements from an an we are using the following built in functions

1) Push() - Used to add Element at the last index of an array.
2) pop() -
3) shift() -
4) unshift() -
5) splice() - (6) split() (7) join() 8. sort().

Eg.

```
push @names, "NBA";
```

Syn: Push @arrayname, value;

Write a perl program to read an elements into an array and display after calling of push function.

```
@courses =("CSE", "ECE", "EEE");

 ~~say(~~
 $size = @courses;
 for(i=0; i<=size; i++)
  say ("~~#~~courses[i]");
  push @courses, "MECH";
  $length = @courses+1;
 for(i=0; i<=length; i++)
   print ("#courses[i]\n");
```

(or)

```
my $c;
my $i;
my $n;
my @colors;
say("Enter n value");
$n = <STDIN>;
```

```
for($i=0; i<n; i++){
  say("Enter any color");
  $c=<STDIN>;
  push(@colors, $c);   (or) #colors[$i] = $c;
  }
  print @colors;
  push(@colors, "Green");
  print @colors;
```

Pop():-

To remove elements from the array which is the last Element.

Syn:- pop(@arrayname);

```
Eg:- @sub = ("WT", "DWDM", "CN", "DAA");
     print @sub;
     pop (@arrayname);
     print @sub;
```

shift():-

This function is used to shift (remove) the first Element

```
Eg:- @sub=("WT","DWDM","CN");
       print @sub;
       shift(@sub);
       print @sub;
```

unshift():- Inserts element at the starting of the array

```
my $c;
my $i;
my $n;
my @numbers = (10,20,80,90);
 Say("Enter n value");
  $n = <STDIN>;
 for(i=1; i<=$n; i++){
   say("Enter ur number");
    $c = <STDIN>;
    unshift (@numbers, $c);
 }
```

Multidimensional array:-

```
@a1 = (10,20,30);
@a2 = (40,50,60);
@a3 = (70,80,90);
@ay = (@a1, @a2, @a3);
```

00	01	02	10	11	12	20	21	22
10	20	30	40	50	60	70	80	90

(or)

```
@ay = ([10,20,30], [40,50,60], [70,80,90]);
```

Eg:-
```
for($i=0; $i<=$#ay; $i++)
    for($j=0; $j<=$#ay; $j++)
        print #ay[$i][$j];
```

Strings:-

→ string is a scalar type in perl.

→ strings are declared by using '$' symbol.

→ strings are represented by using Either single quote (or) double quote.

String operators:-

1. Concatenation operator(.)
2. Repetition " (x)

Eg:-
```
$name = "ABC";
print $name;
$multi = $name x2;
print $multi;      o/p: ABCABC.
```

String handling functions in perl:-

1. length( ) - this function is used to find the length of the given string
2. substr( ) - This function is used to truncate the strings. For this we need to provide half set value, length of the half set value.

Syn:- substr(original string, half set value, length).

Eg:-
```
$mystr = "navya";
$len = length($mystr);
say $mystr;
say $len;
```

Eg:-
```
$mystr = "navya";
$len = length($mystr);
$x = substr($mystr, 2 , $len);
say $x;
```

string comparision:-

used to compare strings.

Eg:-
```
$s1 = "navya";
$s2 = "navyab";
if($s1 == $s2)  (or)  if($s1 Eq $s2)
    say("same");
Else
    say("not same");
```

Concatenation of two strings using complex assignment

Eg:-
```
$s1 = "navya";
$s2 = "madhuri";
$s1. =$s2;
say $s1;
```

Formatting characters in string:- (Escape sequences)

1) \a - bell
2) \b - gives back space
3) \e - Escape next character
4) \l - Transforms the next letter into lower case.
5) \L - " all the characters " " "
6) \n - goes into a new line.
7) \r - carriage return.
8) \t - gives a tab space.
9) \u - converts next letter into upper case.
10) \U - " all the characters " " "

Difference b/w single quote and double quote:-

-> strings can be placed within a double quote or single quote but they have little different behaviour

Eg:-
```
$s1 = 'JNTUK';
$s2 = 'University at kakinada';
print 'welcome to $s1 students $s2';
```

o/p:- welcome to $s1 students $s2

Eg:-
```
$s1 = "JNTUK";
$s2 = "University at kakinada";
print "welcome to $s1 students $s2";
```

o/p:- welcome to JNTUK students university at kakinada.

Reason:-

1. In single quotes are the characters are interpreted as it is.
2. In double quotes variable values are displayed.

Write a perl program to check the given sub string is available in the specified string.

Match operator (=~):- This operator is used to find a substring from a given string

Eg:-
```
$s1 = "university at kakinada";
if($s1 =~ /at/)
    print "matched";
else
    print "not matched";
```

Functions and subroutines :-

→ Functions and subroutines are used to reuse the code.

→ Difference b/w function and subroutine

→ subroutine is created with the keyword sub and it returns a value.

→ You can divide your code into seperate subroutines.

Syn:- sub nameofthe subroutine (~~arguments~~)

```
{
    Body;
}
```

calling :- nameofthe subroutine(Parameter)

write the subroutine to the factorial of a given number.

```
Say ("Enter value");
$n = <STDIN>;
sub fact(~~$n~~)
{   $x = $_[0];
   my $f=1; if($n==0 || $n==1)
Else $f = $n x fact($n-1); return 1;
   return $f;
}
 ~~$~~ $Y = fact(
```

Eg:-

```
Sub display{
   $x = $_[0];
   $Y = $_[1];
   print ("Addition is → $x+$Y");
}
   display(10,20);
```

→ we can pass any no:of arguments inside a subroutine.

→ parameters are passed as a list in the special @_list array variables.

→ so the first argument to the function will be $_[0] and next $_[1] - - -

```
Sub fact{  $n= $_[0];
 if($n==0 || $n==1)
   return 1;
 else
   return $n * fact($n-1)
 }
print "Enter n value.";
$n= <STDIN>;
 fact($n);
```

```
while($n>=2)
{
 $f *= $n;
 $n -=1;
}
```

Eg:- sub display {

```
    @abc = @_;
  print "The list of array Elements -> @abc";
    }
  my @y = (10, 15, 25, 30);
  my @x = ("CSE", "ECE", "EEE");
    display(@x);
    display(@y);
```

Hashes:-

Hashes are used to store group of key ⇒ value pairs.

key - string type {must be unique

value - scalar type

→ Represented by %.

Syn:- % hashname ("key" ⇒ "value");

Note:- what is the difference b/w hashes and arrays?

→ Arrays are orderd but hashes are unorderd.

→ Here array Elements are accessed by using index

but hash values are accessed by using key.

Eg:- % branch("CSE" ⇒ 05);

% branch ("CSE" ⇒ "Computers");

→ For multiple declarations

```
% hashname ("key1" ⇒ value1
       :
       :
   "keyn" ⇒ valuen);
```

Accessing Hash elements:-

Hash elements are accessed by using '$'.

Syn: $hashname{"key"};

$branches {"CSE"}

Write a perl script to access hash values.

```
  % branches( "CSE" ⇒ 05,
              "ECE" ⇒ 04,
              "EEE" ⇒ 02);  → it moves to next key ⇒ value pair
  While (($key, $value) = Each(% branches))
  {
   print "key.⇒.value";     (automatically it increments)
  }
```

Write a peal script to soort the elements.

```
  % courses( "Btech" ⇒ 4,
             "Degree" ⇒ 3,
             "BA" ⇒ 3,
             "BSc" ⇒ 3,
             "MA" ⇒ 2,
             "MTech" ⇒ 2);
```

```
foreach $key (sort keys %courses) {
    print "$key. => . Value";
                      $courses{"key"}
}
```

→ For sorting hashes syntax is

```
Sort keys % hash
```

Write a perl program to check whether the hash key is Existed or not

```
%courses ("Btech" => 4,
          "Mtech" => 2,
          "BA" => 3,
          "MA" => 2);
if( Exists($ ~~hash~~ courses{"key" (BA)}))
{
   print " key found";
}
Else {
   print "Not found";
}
```

Hash slices:—

Eg:—

Command line arguments:—

The arguments which are passed to the perl program through Command line is called as Command line arguments.

→ @ARGV holds command line arguments and @ARGV is like normal array.

→ To acess the arguments from the @ARGV we are using the following procedue ie..,

$ARGV[0], $ARGV[1], ... $ARGV[n-1]

Eg: step1:

```
$x = $ARGV[0];            $z = #$AGRV+1;
$y = $ARGV[1];            if($z != 2) {
print "sum is → . $x+$y";  print "Insufficient no.of arguments";
                           breaks
                          }
```

step2: Execution

Command prompt filename.pl arg1 arg2

Perl DBI (Data Base ~~Interaction~~ Independent Interface);

DSN — Database Source Name.

DBH — Database handle object.

→ Database is accessed within a perl script using DBI module. It provides an abstraction layer b/w perl code and database.

→ DBI takes all sql commands through API and forward them to the appropriate driver for the Execution.

Eg:-

```
Use DBI;
my $driver = "mysql";
my $database = "ceea";
my $dsn = "dbi: $driver: database=$database";
my $username = "root";
my $password = " ";
my $dbh=DBI->Connect( $dsn, $username, $password,
            {
             printError => 0,
             RaiseError => 1,
             AutoCommit => 1,
           FetchHashkeyName => 'NAME', }
            );
    $dbh -> disconnect;
```

Insert into database :- Through insert operation we will pass records into the table (student)

Eg:-

```
Use DBI;
my $driver = "mysql";
my $database = "ceea";
my $dsn = "dbi: $driver: database = $databasename";
my $username = "root";
my $password = " ";
my $dbh = DBI->Connect($dsn,$userna "
$dbh -> prepare("insert into student value(14, "nargo", 87)");
$dbh -> Execute( );
$dbh -> finish( );
```

→ prepare ( ) function is used to prepare the sql query
→ Execute ( ) function " " " execute " "
→ finish( ) " " " close the code.

UNIT-IV

RUBY Programming

→ RUBY is a pure object oriented programming language introduced by Yukihiro Matsumoto in 1993

Features :-

1. Ruby is an open source and is freely available on the web browser.
2. It is a interpreted programming language.
3. Server side scripting language similar to python.
4. can be used to write Common Gateway Interface (CGI).
5. Can be Embedded in HTML.
6. It is a general purpose programming language.
7. It provides built in ~~functions~~ methods as well as used to develop user defined methods.
8. Can be Connected to DB2, MYSQL, Oracle & Sybase.

→ Extension is .rb.

→ "Puts" is used to print the message on console.

Comments :-

is used to put comments for single line comment.

(=begin / =End) is used for multi line Comment

Reserve words :-

BEGIN
END
alias
and
begin
break
def
defined?

elif
Ensure
ief
in
module
for
Ensure
End

not
or
redo
rescue
retry
return
self
unless
undef

Operators :-

1. Arithmetic :- +, −, *, /, %, ** (Exponential)

2. Comparision :- ==, != , <, >, <=, >= , <=> (used for comparing values), === (used to test Equality)

   Eg :- (1...10) === 5 returns true.

3. Complex Assignment :- +=, −=, *=, /=, **=, %=

4. *parallel Assignment :-

   Eg :- a, b, c = 10, 20, 30
   a, b = b, c.

5. Bitwise Operators :- &, |, ^, ~, <<, >>

6. Logical operators :- and, or, &&, ||, !, not

7. Ternary : ?

Range operator :-

... — Excludes upper bound Eg:-(1...10) Excludes 10

.. — includes upper bound Eg:-(1..10) includes 10.

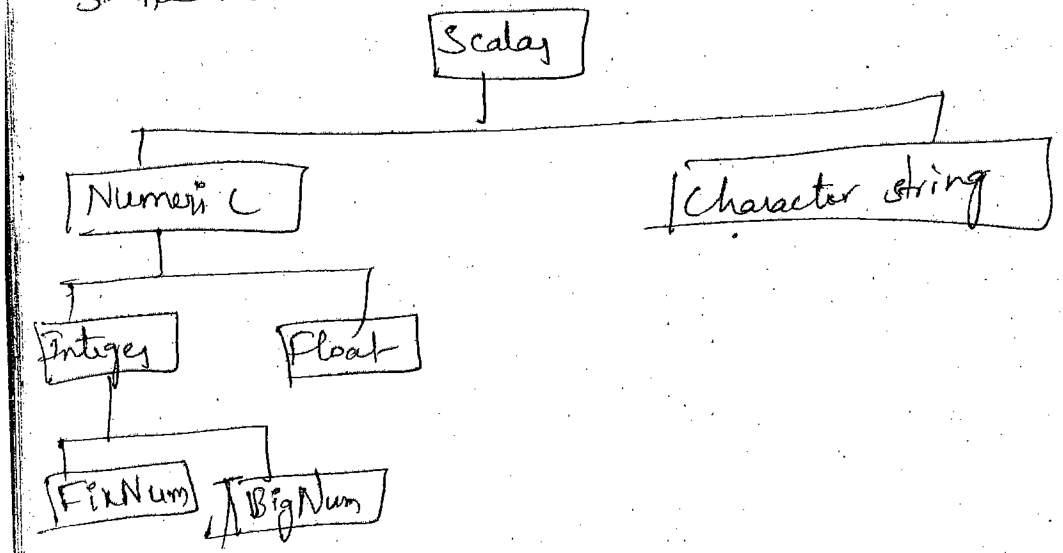
defined? operator :-

defined? variable # True if variable is initialised.

Eg
```
foo = 42
defined? foo   # true
defined? puts  # method.
```

Datatypes :-

1. Scalar
2. Arrays
3. Hashes.



Variably :-

1. Local Variables — declared by using lowercase (or) underscore
2. Instance " — @ variablename
3. class " — @@variablename
4. Global " — $
5. Constants — declared in uppercase

Eg:-
```
$gl_va = 10.
Class Ex
  def print_gl
    Puts "Global variable is #gl_va"
  End
End.
obj = Ex.new
obj.print_gl
```

→ Initialized is a method to initialize object

```
Class ~~Student~~
  def initialized(id, name)
    @stud_id = id
    @stud-name = name
End
```

```
def display_details()
   puts " student id #@stud_id "
   puts " student name #@stud_name "
  End
End
# Creating objects
  s1 = Student.new("14", "navya")
  s1. display_details()
```

→ puts() — defaultly allows new line character.

→ putc() — prints only one character.

→ print — prints message without newline character

Conditional statements

1. if
2. Elsif
3. unless

Case statements

Arrays :-

Creating Arrays :-

```
arrayname = Array.new
      (or)
arrayname = Array.new(size)
```

Eg :-
```
names = Array.new(20)
puts names.size
puts names.length
```

Eg :-
```
names = Array.new(4, "mac")
puts "#{names}"
```

Built-in methods :-

1. array.first
2. array.last
3. array.length
4. array.size
5. array.push(obj, ---) → inserts at last index
6. array.unshift(obj, --) → " " first index
7. array.pop → delets at last position
8. array.shift → " " starting "
9. array.sort
10. array.reverse
11. array.slice(index)(or)
    array.slice(start index, length)
12. arry & other_array → Returns array with no duplicates.
13. " + "
14. " − "
15. " <=> "
16. array.at(index) → to retrieve particular index value.

17. array. clear

18. array. concat(other_array)

19. array. delete_at(index)

20. array. each { |item|block}

Eg :- a = Array. new(10); # a= Array. new

```
Puts "Enter n value"
 n= gets. to_i
Puts "Enter Elements into array"
 a[i] for i in (0...n) do
 a[i]= gets. to_i
  End.
 Puts " Array elements are:"
  for i in (0...n) do
  puts "#{a[i]}";
  End
  Puts " First Element --> . #{a.first}"
  puts " Last Element --> . #{a.last}"
a.clear #puts " clear array --> . #{a.clear}"
  Puts " Array Elements are:"
  for i in (0...n) do  # puts "#{a}"
  Puts "# {a[i]}";
```

Hashes:-

→ perl forces us to make key as a string but in ruby we make it into any scalar type.

Eg:- hashname= Hash. new

Eg:- months = Hash. new("month");

```
 puts "#{months[0]}"
 puts "#{months[72]}"
```

O/p :- month
month

Eg :- months = {"1" => "Jan", "2" => "Feb"}

```
 K = months . keys.
 puts "#{K}"
```

Builtin methods in Hashes:-

1. hash. [key]
2. hash. [key] = value
3. hash. clear
4. hash. delete[key)
5. has. each { |key, value| block}
6. hash. keys
7. hash. values.
8. hash. length
9. hash. merge(other_hash)
10. hash. sort.
11. hash. size

9) methods:—

→ Method name must begin with small letter with is recommended.

Syn
```
def method-name
   Expr
End
      (or)
def method-name(v1, v2)
   expr
End
   (or)
def method name(v1=val1, v2=val2)   (default values)
   Expr
end.
```

Eg:—
```
def test(a1="ruby", a2="perl")
    puts "#{a1}"
    puts "#{a2}"
End
   test "c", "c++"        #o/p:- c. c++
   test                   #o/p- ruby perl
```

Return values:—

```
def test
   i=1
   j=2
   k=3
  return i, j, K.
End
var = test
puts var
```

Eg:—
```
def sample(*test)     ← no:of arguments
   for i in 0...test.length
      puts "#{test[i]}"
   End
End
sample "Navya", "14", "F"
sample "2", "3", "4", "5"
```

Ruby Iterators.—

1. Each
2. Collections.

Eg
```
ary=[1,2,3,4,5]
ary.each do |i|
  puts i
end
```

Regular Expressions :-

→ Spelled as regexp which matches patterns against string

Eg :- /llo/ =~ 'Hello'.
↳ pattern.

→ match() is a method checks whether the given pattern is available in the string or not.

Eg :- /point/. match ("Javapoint")

#<MatchData "point">

Metacharacters & Escapes :-

(,), ?, +, -, *, {, }, [,] used to perform pattern matching.

Eg :- /Hello\, welcome/. match("Does Hello, welcome.")

o/p #<MatchData "Hello, welcome">

Eg :- /2*2\+3\-1=\?/. match("Doe 2*2+3-1=?")

→ Character classes are Encircled within square brackets.

[ab] [ab]

Eg [ab] means a or b which is the opposite of [ab] means a and b.

Eg /j[aeiou]va/. match("Java").

o/p #<MatchData "java">

→ [a-d] is Equivalent to [abcd]

→ [^a-d] represents Except a to d.

[ ]

Repetitions :-

[ab]

* : zero or more times

[a-d].

+ : one or more "

? : zero or one time (optional)

[^

{n} : Exactly n times

{n,m} : At least n and at most m times

{,m} : m or less times

{n,} : n or more times