ANJUMAN
COLLEGE OF ENGINEERING & TECHNOLOGY
(MANAGED BY : ANJUMAN HAMI-E-ISLAM, NAGPUR)

Computer Science
& Engineering
Department

Roll No:_____
Name:____________________
Sem:________Section_______

ANJUMAN COLLEGE OF ENGINEERING & TECHNOLOGY

ESTD. 1999

Approved by A.I.C.T.E. New Delhi, Recognized by DTE, Mumbai, Affiliated to RTM Nagpur University, Nagpur.

CERTIFICATE

Certified that this file is submitted by

Shri/Ku.____

Roll No.________a student of_________year of the course ___________________

____ as a part of PRACTICAL/ORAL as prescribed by the Rashtrasant Tukadoji Maharaj Nagpur University for the subject_______________________________________ in the laboratory of ____________________________________during the academic year __________________________ and that I have instructed him/her for the said work, from time to time and I found him/her to be satisfactory progressive.

And that I have accessed the said work and I am satisfied that the same is up to that standard envisaged for the course.

Date:-	Signature & Name of Subject Teacher	Signature & Name of HOD

Anjuman College of Engineering and Technology

Vision

- To be a centre of excellence for developing quality technocrats with moral and social ethics, to face the global challenges for the sustainable development of society.

Mission

- To create conducive academic culture for learning and identifying career goals.
- To provide quality technical education, research opportunities and imbibe entrepreneurship skills contributing to the socio-economic growth of the Nation.
- To inculcate values and skills, that will empower our students towards development through technology.

Vision and Mission of the Department

Vision:

- To achieve excellent standards of quality education in the field of computer science and engineering, aiming towards development of ethically strong technical experts contributing to the profession in the global society.

Mission:

- To create outcome based education environment for learning and identifying career goals.
- Provide latest tools in a learning ambience to enhance innovations, problem solving skills, leadership qualities team spirit and ethical responsibilities.
- Inculcating awareness through innovative activities in the emerging areas of technology.

Program Educational Objectives (PEOs)

- The graduates will have a strong foundation in mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problem in their career.
- Graduates will be able to create and design computer support systems and impart knowledge and skills to analyze, design, test and implement various software applications.
- Graduates will work productively as computer science engineers towards betterment of society exhibiting ethical qualities.

Program Specific Outcomes (PSOs)

- Foundation of mathematical concepts: To use mathematical methodologies and techniques for computing and solving problem using suitable mathematical analysis, data structures, database and algorithms as per the requirement.
- Foundation of Computer System: The capability and ability to interpret and understand the fundamental concepts and methodology of computer systems and programming. Students can understand the functionality of hardware and software aspects of computer systems, networks and security.
- Foundations of Software development: The ability to grasp the software development lifecycle and methodologies of software system and project development.

COURSE DATA SHEET

PROGRAM: COMPUTER SCIENCE	DEGREE: B.E
COURSE: COMPUTER WORKSHOP-1 LAB	SEMESTER: III CREDITS:
COURSE CODE: BE3S6 REGULATION:	COURSE TYPE: LAB COURSE
COURSE AREA/DOMAIN: WEB	CONTACT HOURS: 2 hours/Week.
CORRESPONDING LAB COURSE CODE (IF ANY): BE3S6	LAB COURSE NAME (IF ANY): COMPUTER WORKSHOP LAB

SYLLABUS:

Practical-No	**DETAILS**	**HOURS**
I	Study of HTML and different HTML Tags	2
II	➢ Design a webpage to print “Good Morning” message in bold and Italic font making colorful background. ➢ Create HTML file which displays three images at left, right and center in the browser ➢ Design a webpage file which contains Hyperlink	2
III	Design HTML document with an example of Table to print your Bio-data	2
IV	Develop a complete webpage using Frames and Frameset which gives information about a Hospital	2
V	Study & Create a webpage using different Form Tags	2
VI	Create a form to fill student Information in detail	2
VII	To Study VB Script.	2
VIII	A) Fibonacci Series program in VB Script. B) VB Script Program for Beginners to generate Date and Time in Different Format.	2
IX	To Study Java Script.	2
X	Java Script example program for Fibonacci series. Java Script program to check odd or even numbers with example.	2
XI	To study about Unix Operating System	
XII	To Study about Linux Operating System	
	TOTAL HOURS	20

TEXT/REFERENCE BOOKS:

T/R	BOOK TITLE/AUTHORS/PUBLICATION
	HTML Programming, Freeman and Robson, Oreilly publications

COURSE PRE-REQUISITES:

C.CODE	COURSE NAME	DESCRIPTION	SEM

COURSE OBJECTIVES:

1	➢ To enumerate basics of Web Programming with HTML
2	➢ To enumerate basics of Web Programming with CSS
3	➢ To design web pages with basic as well as advanced tools including HTML & CSS

COURSE OUTCOMES:

S.NO	**DESCRIPTION** *Upon Completion of course students will be able TO*	**Bloom Taxonomy Level**
CO1	**Describe different tools for Web Programming.**	**2**
CO2	**Demonstrate basics of HTML Tags**	**3**

CO3	**Develop a dynamic webpages by the use of HTML including forms, syntax, headings, linking, images, special characters and horizontal rules, lists, tables, forms.**	**6**
CO4	**Study the concepts of VB Script & its usage in Web Development.**	**4**
CO5	**Develop web pages using Java Script**	**6**
CO6	**Working under Unix / Linux Operating Systems.**	**3**

COURSE OUTCOMES VS POS MAPPING (DETAILED; HIGH:3; MEDIUM:2; LOW:1**):**

SNO		PO1	PO2	PO3	PO4	PO 5	PO6	PO7	PO 8	PO9	PO10	PO 11	PO 12	PSO 1	PSO2	PSO 3
CO1		**2**	**1**	**1**		**1**			**1**	**1**	**1**	**2**	**1**			
CO2		**2**	**1**	**3**		**3**			**1**	**1**	**1**	**2**	**1**		**2**	**1**
CO3		**2**	**1**	**3**		**3**			**1**	**1**	**1**	**2**	**1**		**2**	**1**
CO4		**2**	**1**	**3**		**3**			**1**	**1**	**1**	**2**	**1**		**2**	**1**
CO5		**2**	**1**	**3**		**3**			**1**	**1**	**1**	**2**	**1**		**2**	**1**
CO6		**2**	**1**	**2**		**3**			**1**	**1**	**1**	**2**	**1**		**2**	**1**
	Avg CO's Calculation															

** For Entire Course, PO /PSO Mapping; 1 (Low); 2(Medium); 3(High) Contribution to PO/PSO*

PO1F	EngineeringFKnowledgeF	PO7F	EnvironmentF&FSustainabilityF	PSO1F	DomainFSkills1F
PO2F	ProblemFAnalysisF	PO8F	EthicsF	PSO2F	DomainFSkills2F
PO3F	DesignF&FDevelopmentF	PO9F	IndividualF&FTeamFWorkF	F	F
PO4F	InvestigationsF	PO10F	CommunicationFSkillsF	F	F
PO5F	ModernFToolsF	PO11F	ProjectFMgt.F&FFinanceF	F	F
PO6F	EnginerrF&FSocietyF	PO12F	LifeFLongFLearningF	F	F

F

JUSTIFICATION FOR MAPPING

SNO	**PO/PSO MAPPED**	**JUSTIFICATION**
CO1	**PO1,PO2,PO3,PSO1 &PSO2**	**CO1 is related with Engineering Knowledge, Problem Analysis, Design & Development, Investigations, Team Work, Communication and Mathematical Analysis.**
CO.2	**PO1,PO2,PO3,PSO1 &PSO2**	**CO2 is related with Engineering Knowledge, Problem Analysis, Design & Development, Investigations, Team Work, Communication and Mathematical Analysis.**
CO.3	**PO1,PO2,PO3,PSO1 &PSO2**	**CO3 is related with Engineering Knowledge, Problem Analysis, Design & Development, Investigations, Team Work, Communication and Mathematical Analysis.**
CO.4	**PO1,PO2,PO3,&PSO2**	**CO4is related with Engineering Knowledge, Problem Analysis, Design & Development, Investigations , Team Work, Communication & Fundamentals of Hardware & Software aspects of computer system.**
CO.5	**PO1,PO2,PO3 &PSO2**	**CO5 is related with Engineering Knowledge, Problem Analysis, Design & Development, Investigations, Team Work, Communication and Fundamentals of Hardware & Software aspects of computer system.**
CO.6	**PO1,PO2,PO3,PSO1 &PSO2**	**CO6 is related with Engineering Knowledge, Problem Analysis, Design & Development, Investigations, Team Work, Communication, Project Management , Mathematical Analysis & Fundamentals of Hardware & Software aspects of computer system**

GAPES IN THE SYLLABUS - TO MEET INDUSTRY/PROFESSION REQUIREMENTS, POs:

SNO	DESCRIPTION	PROPOSED ACTIONS
1		
2		

3		
4		
5		

PROPOSED ACTIONS: TOPICS BEYOND SYLLABUS/ASSIGNMENT/INDUSTRY VISIT/GUEST LECTURER/NPTEL ETC

TOPICS BEYOND SYLLABUS/ADVANCED TOPICS/DESIGN:

1	BOOTSTRAPPING
2	ANGULAR JS
3	JAVASCRIPTING
4	NODE JS
5	IONIC FRAMEWORK
6	ANY OTHER LASTEST WEB PROGRAMMING LANGUAGE

WEB SOURCE REFERENCES:

1	https://www.w3schools.com/html/
2	https://www.tutorialspoint.com/html/
3	https://www.tutorialspoint.com/html/html_basic_tags.htm
4	https://www.tutorialspoint.com/html/html_images.htm
5	https://www.tutorialspoint.com/html/html_tables.htm
6	https://www.tutorialspoint.com/html/html_frames.htm
7	https://www.tutorialspoint.com/html/html_forms.htm
8	https://www.w3schools.com/html/html_css.asp
9	https://www.tutorialspoint.com/css/
10	https://www.tutorialspoint.com/css/css_inclusion.htm
11	https://www.tutorialspoint.com/css/css_colors.htm
12	https://www.tutorialspoint.com/css/css_backgrounds.htm
13	https://www.tutorialspoint.com/css/css_padding.htm
14	https://www.tutorialspoint.com/css/css_margins.htm

DELIVERY/INSTRUCTIONAL METHODOLOGIES:

✓ CHALK & TALK	STUD. ASSIGNMENT	✓ WEB RESOURCES	✓ NPTEL/OTHERS
☐ LCD/SMART BOARDS	STUD. SEMINARS	☐ ADD-ON COURSES	☐ WEBNIARS

ASSESSMENT METHODOLOGIES-DIRECT

✓ ASSIGNMENTS	STUD. SEMINARS	✓ TESTS/MODEL EXAMS	✓ UNIV. EXAMINATION
✓ STUD. LAB PRACTICES	✓ STUD. VIVA	☐ MINI/MAJOR PROJECTS	☐ CERTIFICATIONS
☐ ADD-ON COURSES	☐ OTHERS		

ASSESSMENT METHODOLOGIES-INDIRECT

✓ ASSESSMENT OF COURSE OUTCOMES (BY FEEDBACK, ONCE)	✓ STUDENT FEEDBACK ON FACULTY (TWICE)
☐ ASSESSMENT OF MINI/MAJOR PROJECTS BY EXT. EXPERTS	☐ OTHERS

INNOVATIONS IN TEACHING/LEARNING/EVALUATION PROCESSES:

1.
2.
3.
4.
5.

Prepared by **Approved by**

(PROF. SYED REHAN) **(HOD)**

Lab Instructions:

- Make entry in the Log Book as soon as you enter the Laboratory.
- All the students should sit according to their Roll Numbers.
- All the students are supposed to enter the terminal number in the Log Book.
- Do not change the terminal on which you are working.
- Strictly observe the instructions given by the Faculty / Lab. Instructor.
- Take permission before entering in the lab and keep your belongings in the racks.
- NO FOOD, DRINK, IN ANY FORM is allowed in the lab.
- TURN OFF CELL PHONES! If you need to use it, please keep it in bags.
- Avoid all horseplay in the laboratory. Do not misbehave in the computer laboratory. Work quietly.
- Save often and keep your files organized.
- Don’t change settings and surf safely.
- Do not reboot, turn off, or move any workstation or PC.
- Do not load any software on any lab computer (without prior permission of Faculty and Technical Support Personnel). Only Lab Operators and Technical Support Personnel are authorized to carry out these tasks.
- Do not reconfigure the cabling/equipment without prior permission.
- Do not play games on systems.
- Turn off the machine once you are done using it.
- Violation of the above rules and etiquette guidelines will result in disciplinary action.

Continuous Assessment Practical

Exp No	NAME OF EXPERIMENT	Date	Sign	Remark
1	To Study HTML and Different HTML Tags.			
2	A) Design a web page to print Good Morning message in bold and italic font making colorful background. B) Create an HTML file which displays three images at the Left, Right and Center in the Browser. C) Design a web page file which contains a hyper Link.			
3	Design an HTML document with an example of Table to print your Bio Data.			
4	Develop a complete web Page using Frames and Frameset which gives information about a Hospital			
5	To Create a web page using different form tag.			
6	Create a Form to fill Student Information.			
7	To Study VB Script.			
8	A) Fibonacci Series program in VB Script. B) VB Script Program for Beginners to generate Date and Time in Different Format.			
9	To Study Java Script.			
10	Java Script example program for Fibonacci series. Java Script program to check odd or even numbers with example.			
11	To study Unix			
12	To Study Linux			

CONTENTS

Practical No.1

Aim :- To study HTML & different HTML tags

HTML stands for Hyper Text Markup Language, which is the most widely used language on Web to develop web pages. HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

HTML stands for **H**yper**t**e**x**t **M**arkup **L**anguage, and it is the most widely used language to write Web Pages.

- **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

HTML Tags

As told earlier, HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<Tag Name>**. Except few tags, most of the tags have their corresponding closing tags. For example, **<html>** has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag etc.

Above example of HTML document uses the following tags −

Sr.No	Tag & Description
1	**<!DOCTYPE...>** This tag defines the document type and HTML version.
2	**<html>** This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.
3	**<head>**

	This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
4	**<title>** The <title> tag is used inside the <head> tag to mention the document title.
5	**<body>** This tag represents the document's body which keeps other HTML tags like <h1>, <div>, <p> etc.
6	**<h1>** This tag represents the heading.
7	**<p>** This tag represents a paragraph.

To learn HTML, you will need to study various tags and understand how they behave, while formatting a textual document. Learning HTML is simple as users have to learn the usage of different tags in order to format the text or images to make a beautiful webpage.

World Wide Web Consortium (W3C) recommends to use lowercase tags starting from HTML 4.

HTML Document Structure

A typical HTML document will have the following structure –

```
<html>

   <head>
      Document header related tags
   </head>

   <body>
      Document body related tags
   </body>

</html>
```

We will study all the header and body tags in subsequent chapters, but for now let's see what is document declaration tag.

The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of the following declaration −

```
<!DOCTYPE html>
```

There are many other declaration types which can be used in HTML document depending on what version of HTML is being used. We will see more details on this while discussing <!DOCTYPE...> tag along with other HTML tags.

Heading Tags

Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements **<h1>, <h2>, <h3>, <h4>, <h5>,** and **<h6>**. While displaying any heading, browser adds one line before and one line after that heading.

Example

```
<!DOCTYPE html>

<html>

   <head>

      <title>Heading Example</title>

   </head>

   <body>

      <h1>This is heading 1</h1>

      <h2>This is heading 2</h2>

      <h3>This is heading 3</h3>

      <h4>This is heading 4</h4>

      <h5>This is heading 5</h5>

      <h6>This is heading 6</h6>

   </body>

         </html
```

This will produce the following result −

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

Paragraph Tag

The **<p>** tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening <p> and a closing </p> tag as shown below in the example –

Example

```
<!DOCTYPE html>

<html>


   <head>

      <title>Paragraph Example</title>

   </head>


   <body>

      <p>Here is a first paragraph of text.</p>

      <p>Here is a second paragraph of text.</p>

      <p>Here is a third paragraph of text.</p>

   </body>

</html>
```

This will produce the following result –

Here is a first paragraph of text.

Here is a second paragraph of text.

Here is a third paragraph of text

Line Break Tag

Whenever you use the **<br />** element, anything following it starts from the next line. This tag is an example of an **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

The <br /> tag has a space between the characters **br** and the forward slash. If you omit this space, older browsers will have trouble rendering the line break, while if you miss the forward slash character and just use <br> it is not valid in XHTML.

Example

```
<!DOCTYPE html>

<html>

 <head>

      <title>Line Break  Example</title>

   </head>


   <body>

      <p>Hello<br />

         You delivered your assignment ontime.<br />

         Thanks<br />

         Mahnaz</p>

   </body>

</html>
```

This will produce the following result –

Hello
You delivered your assignment on time.
Thanks
Mahnaz

Centering Content

You can use **<center>** tag to put any content in the center of the page or any table cell.

Example

```
<!DOCTYPE html>

<html>



   <head>

      <title>Centring Content Example</title>

   </head>



   <body>

      <p>This text is not in the center.</p>



      <center>

         <p>This text is in the center.</p>

      </center>

   </body>



</html>
```

This will produce following result −

This text is not in the center.

This text is in the center.

Horizontal Lines

Horizontal lines are used to visually break-up sections of a document. The **<hr>** tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

For example, you may want to give a line between two paragraphs as in the given example below −

Example

```
<!DOCTYPE html>

<html>

   <head>
```

```
      <title>Horizontal Line Example</title>
   </head>


   <body>
      <p>This is paragraph one and should be on top</p>
      <hr />
      <p>This is paragraph two and should be at bottom</p>
   </body>
</html>
```

This will produce the following result −

This is paragraph one and should be on top

This is paragraph two and should be at bottom

Again **<hr />** tag is an example of the **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

The **<hr />** element has a space between the characters **hr** and the forward slash. If you omit this space, older browsers will have trouble rendering the horizontal line, while if you miss the forward slash character and just use **<hr>** it is not valid in XHTML

Preserve Formatting

Sometimes, you want your text to follow the exact format of how it is written in the HTML document. In these cases, you can use the preformatted tag **<pre>**.

Any text between the opening **<pre>** tag and the closing **</pre>** tag will preserve the formatting of the source document.

Example

```
<!DOCTYPE html>
<html>


   <head>
      <title>Preserve Formatting Example</title>
   </head>
   <body>

      <pre>

         function testFunction( strText ){

            alert (strText)

         }

      </pre>

   </body>



</html>
```

This will produce the following result –

```
function testFunction( strText ){
   alert (strText)
}
```

Try using the same code without keeping it inside **<pre>...</pre>** tags

Non-breaking Spaces

Suppose you want to use the phrase "12 Angry Men." Here, you would not want a browser to split the "12, Angry" and "Men" across two lines –

```
An example of this technique appears in the movie "12 Angry Men."
```

In cases, where you do not want the client browser to break text, you should use a non-breaking space entity **&nbsp;** instead of a normal space. For example, when coding the "12 Angry Men" in a paragraph, you should use something similar to the following code –

Example

```
<!DOCTYPE html>

<html>

<head>

      <title>Nonbreaking Spaces Example</title>
```

```
   </head>



   <body>

      <p>An example of this technique appears in the movie
"12&nbsp;Angry&nbsp;Men."</p>

   </body>

</html>
```

This will produce the following result –

An example of this technique appears in the movie "12 Angry Men".

Conclusion: Thus we understood the basic tags of HTML which are used for making different web-pages we concluded how to use and when to use the tags and the basic format of HTML

Practical No.2

Aim :- Design a web page to print "good morning" in bold and italics point making colourful background.

Description:

<html>:-This tag encloses the complete HTML document and mainly comprises of document header which is represented by<head>---</head> &document's body which is represented by<body> ----- </body>tags

<head>:-This tag represent the document header which can keep other HTML tags like <title>.etc

<title>:-The <title>tag is used inside the<head> tag to mention the document title

<body>:-This tag represents the document body which keeps the other HTML tags<h1>.etc

<center>:This tag is used to display any content in the center of the page or any table cell.

<b>:-Anything that appears within <b> --- </b> element, is displayed in bold.

<i>:-Anything that appears within <i>---</i> element is displayed in italics

Html Background with Colors

The **bgcolor** attribute is used to control the background of an HTML element, specifically page body and table backgrounds.

Following is the syntax to use bgcolor attribute with any HTML tag.

```
<tagname bgcolor = "color_value"...>
```

Code:

```
<html>
<head>

<title>my first web page</title>

</head>

<body bgcolor="white"text="black">

<h1><center><b><i>GOOD MORNING</b></i></center></h1>

</body>
```

```
</html>
```

Output

Practical No.2 (B)

Aim: - Create a HTML file which displays three images at the left ,right &center in the browser.

Description:

<html>:-This tag encloses the complete HTML document and mainly comprises of document header which is represented by<head>---</head> &document's body which is represented by<body> ----- </body>tags

<head>:-This tag represent the document header which can keep other HTML tags like <title>.etc

<title>:-The <title>tag is used inside the<head> tag to mention the document title

<body>:-This tag represents the document body which keeps the other HTML tags<h1>.etc

<center>:This tag is used to display any content in the center of the page or any table cell.

Bgcolor: The **bgcolor** attribute is used to control the background of an HTML element, specifically page body and table backgrounds.

Following is the syntax to use bgcolor attribute with any HTML tag.

```
<tagname bgcolor = "color_value"...>
```

Insert Image

You can insert any image in your web page by using **<img>** tag. Following is the simple syntax to use this tag.

```
<img src = "Image URL" ... attributes-list/>
```

The <img> tag is an empty tag, which means that, it can contain only list of attributes and it has no closing tag.

Set Image Location

Usually we keep all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory **images** inside the home directory where we will keep our image test.png.

Set Image Width/Height

You can set image width and height based on your requirement using **width** and **height** attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

By default, image will align at the left side of the page, but you can use **align** attribute to set it in the center or right.

CODE :

```
<html>

<head>

<title>my second web page</title>

</head>

<body bgcolor="black"text="white"><p><h1><center>Three images aligned as
left,right and center</p></center></h1>

<img src="D:\The big folder of photos\HD Wallpapers\Full HD 1080p Wallpapers
(15).jpg"height="200"width="300"align="right">

<img src="D:\The big folder of photos\HD Wallpapers\Full HD 1080p Wallpapers
(15).jpg"height="200"width="300">

<center><img src="D:\The big folder of photos\HD Wallpapers\Full HD 1080p
Wallpapers (15).jpg"height="200"width="300"></center>

</body>

</html>
```

OUTPUT:-

Practical No.2 (C)

Aim:-Design a webpage which contains hyperlink

Description:

<html>:-This tag encloses the complete HTML document and mainly comprises of document header which is represented by<head>---</head> &document's body which is represented by<body> ----- </body>tags

<head>:-This tag represent the document header which can keep other HTML tags like <title>.etc

<title>:-The <title>tag is used inside the<head> tag to mention the document title

<body>:-This tag represents the document body which keeps the other HTML tags<h1>.etc

<center>:This tag is used to display any content in the center of the page or any table cell.

Bgcolor: The **bgcolor** attribute is used to control the background of an HTML element, specifically page body and table backgrounds.

Following is the syntax to use bgcolor attribute with any HTML tag.

```
<tagname bgcolor = "color_value"...>
```

Linking Documents

A link is specified using HTML tag <a>. This tag is called **anchor tag** and anything between the opening <a> tag and the closing </a> tag becomes part of the link and a user can click that part to reach to the linked document. Following is the simple syntax to use <a> tag.

```
<a href = "Document URL" ... attributes-list>Link Text</a>
```

Code:

```
<html>
<head>

<title>hyperlink</title>

</head>

<body bgcolor="orange"text="white">

<center><a href="http://www.google.com">GOOGLE</a></center>
```

```
</body>

</html>
```

OUTPUT:

Practical No 3

Aim: Design an HTML document with the example of table to print employee detail.

Description:

<html>:-This tag encloses the complete HTML document and mainly comprises of document header which is represented by<head>---</head> &document's body which is represented by<body> ----- </body>tags

<head>:-This tag represent the document header which can keep other HTML tags like <title>.etc

<title>:-The <title>tag is used inside the<head> tag to mention the document title

<body>:-This tag represents the document body which keeps the other HTML tags<h1>.etc

<center>:This tag is used to display any content in the center of the page or any table cell.

<b>:-Anything that appears within <b> --- </b> element, is displayed in bold.

<i>:-Anything that appears within <i>---</i> element is displayed in italics

Bgcolor: The **bgcolor** attribute is used to control the background of an HTML element, specifically page body and table backgrounds.

Following is the syntax to use bgcolor attribute with any HTML tag.

```
<tagname bgcolor = "color_value"...>
```

Table

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

Table Heading

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

Cellpadding and Cellspacing Attributes

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

Colspan and Rowspan Attributes

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

Code:

```
<html>

<head>

<title>table of emplyoee</title>

</head>

<body background="D:\The big folder of photos\HD Wallpapers\Full HD 1080p Wallpapers (25).jpg">

<b><h1><CENTER>THE TABLE OF EMPLYOEES OF DIFFERENT DEPARTMENTS</CENTER></H1></B>

<center><table border="1" cellpadding="5" cellspacing="5" BGCOLOR="ORANGE">

<tr bgcolor="red">

<th>SR.NO</TH>

<th>NAME</th>

<th>SALARY</th>

<th>DEPARTMENT</th>

<th>SIGNATURE</th>

</tr>

<tr>

<td>01</td>
```

```
<td>PUSHPAK</td>

<td>1000000</td>

<td>RESEARCH AND DEVELOPMENT</td>

<td>  </td>

</tr>

<tr>

<td rowspan="2">02</td>

<td>ROHISH</td>

<td>300000</td>

<td>MOBILES AND NETWORK</td>

<td>  </td>

</tr>

<tr>

<td>03</td>

<td>SMIT</td>

<td>4000000</td>

<td>FOOD AND TECHNOLOGY</td>

<td>  </td>

</tr>

<tr>

<td>04</td>

<td>TEJAS</td>

<td>700000</td>
```

```
<td>FOOD AND FLAVOURS</td>

<td>  </td>

</tr>

<tr>

<td>05</td>

<td>MANAL</td>

<td>89076</td>

<td>AUTOMOBILES</td>

<td>  </td>

</tr>

<tr>

<td>06</td>

<td>KALPAK</td>

<td>80976</td>

<td>ELECTRICAL</td>

<td>  </td>

</tr>

<tr>

<td>07</td>

<td>HARSHIT</td>

<td>567894</td>

<td>CARS AND DESIGN</td>

<td>  </td>
```

```
</tr>

<tr>

<td>08</td>

<td>SAMYAK</td>

<td>908753</td>

<td>ELECTRONIC DEVICES</td>

<td>  </td>

</tr>

<tr>

<td>09</td>

<td>DEEPAK</td>

<td>87654</td>

<td>BUILDINGS AND INFRASTRUCTURE</td>

<td>  </td>

</tr>

<tr>

<td>10</td>

<td>PIYUSH</td>

<td>98076</td>

<td>SOFTWARE DEVELOPMENT</td>

<td>  </td>

</tr>

<tr>
```

```
<td>11</td>

<td>IMDAD</td>

<td>98765432</td>

<td>ENGLISH LITERATURE</td>

<td>  </td>

</tr>

<tr>

<td>12</td>

<td>SOHAIL</td>

<td>897000</td>

<td>SCIENCE AND HUMANITIES</td>

<td>  </td>

</tr>

</table></center>

</body>

</html>
```

OUTPUT:

Practical No 4:

Aim: Develop a complete webpage using frames &frameset which gives information about employee

Description:

<html>:-This tag encloses the complete HTML document and mainly comprises of document header which is represented by<head>---</head> &document's body which is represented by<body> ----- </body>tags

<head>:-This tag represent the document header which can keep other HTML tags like <title>.etc

<title>:-The <title>tag is used inside the<head> tag to mention the document title

<body>:-This tag represents the document body which keeps the other HTML tags<h1>.etc

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back* button might not work as the user hopes.
- There are still few browsers that do not support frame technology.

Creating Frames

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

The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag –

Sr.No	Attribute & Description
1	**cols** Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways − Absolute values in pixels. For example, to create three vertical frames, use *cols = "100, 500, 100"*. A percentage of the browser window. For example, to create three vertical frames, use *cols = "10%, 80%, 10%"*. Using a wildcard symbol. For example, to create three vertical frames, use *cols = "10%, *, 10%"*. In this case wildcard takes remainder of the window. As relative widths of the browser window. For example, to create three vertical frames, use *cols = "3*, 2*, 1*"*. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
2	**rows** This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use *rows = "10%, 90%"*. You can specify the height of each row in the same way as explained above for columns.
3	**border** This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border.
4	**frameborder** This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.
5	**framespacing** This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing

	between each frames.

The <frame> Tag Attributes

Following are the important attributes of <frame> tag −

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

Code:

```
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset rows = "30%,40%,30%" border="5" framespacing="5">
<frame name = "top" src = "G:\_\New folder (3)\practical no
4\FRAME1.html">
<frame name = "main" scrolling="yes" src = "G:\_\New folder
(3)\practical no 4\FRAME2.html">
<frame name = "bottom" src = "G:\_\New folder (3)\practical no
4\FRAME3.html">
<noframes>
<body>Your browser does not support frames.</body>
</noframes>
</frameset>
</html>
```

OUTPUT:

Practical No. 5 & 6

Aim:-To create a Web page using different form tag create a form to fill students information.

Description:

<html>:-This tag encloses the complete HTML document and mainly comprises of document header which is represented by<head>---</head> &document's body which is represented by<body> ----- </body>tags

<head>:-This tag represent the document header which can keep other HTML tags like <title>.etc

<title>:-The <title>tag is used inside the<head> tag to mention the document title

<body>:-This tag represents the document body which keeps the other HTML tags<h1>.etc

<center>:This tag is used to display any content in the center of the page or any table cell.

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

```
<form action = "Script URL" method = "GET|POST">
   form elements like input, textarea etc.
</form>
```

Form Attributes

Apart from common attributes, following is a list of the most frequently used form attributes −

Sr.No	Attribute & Description
1	**action** Backend script ready to process your passed data.
2	**method** Method to be used to upload data. The most frequently used are GET and POST methods.
3	**target** Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
4	**enctype**

	You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are − **application/x-www-form-urlencoded** − This is the standard method most forms use in simple scenarios. **mutlipart/form-data** − This is used when you want to upload binary data in the form of files like image, word file etc.

Note − You can refer to Perl & CGI for a detail on how form data upload works.

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form −

- Text Input Controls
- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

Text Input Controls

There are three types of text input used on forms −

- **Single-line text input controls** − This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>** tag.
- **Password input controls** − This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.
- **Multi-line text input controls** − This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

Single-line text input controls

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.

Attributes

Following is the list of attributes for <input> tag for creating text field.

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

Password input controls

This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input>tag but type attribute is set to **password**.

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **checkbox.**.

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **radio**.

Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

Button Controls

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input>tag by setting its type attribute to **button**. The type attribute can take the following values –

Sr.No	Type & Description
1	**submit** This creates a button that automatically submits a form.

CODE:

```
<!doctype html>

<html>

<head>

<title>to diplay the forms</title>

</head>

<body background="G:\_\The big folder of photos\HD
Wallpapers\backgrounddefault.jpg"text="black">

<>

<pre>

<h1><center>APLICATION FORM</H1>

</CENTER>

<center><form bgcolor="yellow">FIRST NAME:<input
type="text"placeholder="FIRST NAME" name="">

<br>

LAST NAME:<input type="text" placeholder="LAST NAME"
name=""><br>

USER ID: <input type="text"  placeholder="USER ID"name=""><br>

PASSWORD:<input type="password" placeholder="PASSWORD"
name=""><br>

MALE:<input type="radio" name="GENDER" value="MALE">
FEMALE:<input type="radio" name="GENDER" value="FEMALE">

SUBJECT:<select name="dropdown">

<option value="Maths" selected>Maths</option>
```

```
<option value="Physics" selected>Physics</option>

<option value="Physics" selected>chemistery</option>

<option value="Physics" selected>"c" programming</option>

<option value="Physics" selected>HTML</option></SELECT>

ADDRESS:<input type="address"
placeholder="ADDRESS"name=""><br>

EMAIL :<input type="email" placeholder="E-
MAIL"name="email"><br>

COMMENT: <br>

<textarea rows = "10" cols = "100" bgcolor="green" name =
"descripton">

Enter your cooment............

</textarea>

<input type="submit" name="submit" value="SUBMIT"><br>

</form></center></pre>

</body>

</html>
```

OUTPUT:

Practical No. 7

AIM: To Study VB Script.

THEORY:

VBScript stands for **V**isual **B**asic Scripting that forms a subset of Visual Basic for Applications Microsoft VBScript (Visual Basic Script) is a general-purpose, lightweight and active scripting language developed by Microsoft that is modelled on Visual Basic. Nowadays, VBScript is the primary scripting language for Quick Test Professional (QTP), which is a test automation tool. This tutorial will teach you how to use VBScript in your day-to-day life of any Web-based or automation project development.

Features of VBScript

- VBScript is a lightweight scripting language, which has a lightning fast interpreter.
- VBScript, for the most part, is case insensitive. It has a very simple syntax, easy to learn and to implement.
- Unlike C++ or Java, VBScript is an object-based scripting language and NOT an Object- Oriented Programming language.
- VBScript is used by Active Server Pages **(ASP)**, a server side scripting environment for creating dynamic WebPages.
- VBScript is used for Client side scripting in Microsoft Internet Explorer.

Sample VBScript

Let us write a VBScript to print out "Hello World".

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

In the above example, we called a function document.write, which writes a string into the HTML document. This function can be used to write text, HTML, or both. So, the above code will display the following result:

```
Hello  World!
```

Whitespace and Line Breaks

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

Formatting

VBScript is based on Microsoft's Visual Basic. Unlike JavaScript, no statement terminators such as semicolon is used to terminate a particular statement.

Single Line Syntax

Colons are used when two or more lines of VBScript ought to be written in a single line. Hence, in VBScript, Colons act as a line separator.

```
<script language="vbscript" type="text/vbscript">
var1 = 10 :var2= 20
</script>
```

Multiple Line Syntax

When a statement in VBScript is lengthy and if user wishes to break it into multiple lines, then the user has to use underscore "_". This improves the readability of the code. The following example illustrates how to work with multiple lines.

```
<script language="vbscript" type="text/vbscript">
 var1 = 10
var2  = 20
Sum = var1 + var2
document.write("The  Sum  of  two  numbers"&_ "var1 and var2 is " & Sum)
</script>
```

Case Sensitivity

VBScript is a **case-insensitive language**. This means that language keywords, variables, function names and any other identifiers need NOT be typed with a consistent capitalization of letters. So identifiers int_counter, INT_Counter and INT_COUNTER have the same meaning within VBScript.

Comments in VBScript

Comments are used to document the program logic and the user information with which other programmers can seamlessly work on the same code in future.Comments are ignored by the interpreter while execution. Comments in VBScript are denoted by two methods.

Any statement that starts with a Single Quote (‘) is treated as comment. Following is the example:

```
<script language="vbscript" type="text/vbscript">
<!—
' This Script is invoked after successful login
' Written by : TutorialsPoint
' Return Value : True / False
```

```
//- >
</script>
```

Any statement that starts with the keyword “REM”. Following is the example:

```
<script language="vbscript" type="text/vbscript">
<!—
REM This Script is written to Validate the Entered Input
REM Modified by : Tutorials point/user2
//- > </script>
```

VBScript– Enabling in Browsers

Not all the modern browsers support VBScript. VBScript is supported just by Microsoft's Internet Explorer while other browsers (Firefox and Chrome) support just JavaScript. Hence, developers normally prefer JavaScript over VBScript.

Though Internet Explorer (IE) supports VBScript, you may need to enable or disable this feature manually. This tutorial will make you aware of the procedure of enabling and disabling VBScript support in Internet Explorer.

VBScript in Internet Explorer

Here are simple steps to turn on or turn off VBScript in your Internet Explorer:

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

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

VBScript Placement in HTML File

There is a flexibility given to include VBScript code anywhere in an HTML document. But the most preferred way to include VBScript in your HTML file is as follows:

- Script in <head>...</head> section.
- Script in <body>...</body> section
- Script in <body>...</body> sections.
- If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head

VBScript in <body>...</body> section

If you need a script to run as the page loads so that the script generates content in the page, the script goes in the <body> portion of the document. In this case, you would not have any function defined using VBScript:

```
<html>
<head>
</head>
<body>
<script  type="text/vbscript">
<!--
document.write("Hello  World")
//-->
</script>
<p>This is web page body </p>
</body>
</html>
```

It will produce the following result:

```
Hello  World
```

This is web page body

VBScript Variables

A variable is a named memory location used to hold a value that can be changed during the script execution. VBScript has only **ONE** fundamental data type, **Variant**.

Rules for Declaring Variables:

- Variable Name must begin with an alphabet.
- Variable names cannot exceed 255 characters.
- Variables Should NOT contain a period (.)
- Variable Names should be unique in the declared context.

Declaring Variables

Variables are declared using “dim” keyword. Since there is only ONE fundamental data type, all the declared variables are variant by default. Hence, a user **NEED NOT** mention the type of data during declaration.

Example 1: In this Example, IntValue can be used as a String, Integer or even arrays.

```
Dim Var
```

Example 2: Two or more declarations are separated by comma(,)

```
Dim  Variable1,Variable2
```

Assigning Values to the Variables

Values are assigned similar to an algebraic expression. The variable name on the left hand side followed by an equal to (=) symbol and then its value on the right hand side.

Rules

- The numeric values should be declared without double quotes.
- The String values should be enclosed within double quotes(")
- Date and Time variables should be enclosed within hash symbol(#)

Scope of the Variables

Variables can be declared using the following statements that determines the scope of the variable. The scope of the variable plays a crucial role when used within a procedure or classes.

- Dim
- Public
- Private

Dim

Variables declared using “Dim” keyword at a Procedure level are available only within the same procedure. Variables declared using “Dim” Keyword at script level are available to all the procedures within the same script.

Public

Variables declared using "Public" Keyword are available to all the procedures across all the associated scripts. When declaring a variable of type "public", Dim keyword is replaced by "Public".

Private

Variables that are declared as "Private" have scope only within that script in which they are declared. When declaring a variable of type "Private", Dim keyword is replaced by "Private".

VBScript– Constants

Constant is a named memory location used to hold a value that CANNOT be changed during the script execution. If a user tries to change a Constant Value, the Script execution ends up with an error. Constants are declared the same way the variables are declared.

Declaring Constants

Syntax

```
[Public  |  Private]  Const  Constant_Name  =  Value
```

The Constant can be of type Public or Private. The Use of Public or Private is Optional. The Public constants are available for all the scripts and procedures while the Private Constants are available within the procedure or Class. One can assign any value such as number, String or Date to the declared Constant.

VBScript– Operators

Let’s take an expression *4 + 5 is equal to 9*. Here, 4 and 5 are called **operands** and + is called the **operator**. VBScript language supports following types of operators:

- Arithmetic Operators

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

The Arithmetic Operators

VBScript supports the following arithmetic operators: Assume variable A holds 5 and variable B holds 10, then:

Operator	Description	Example
+	Adds two operands	A + B will give 15
-	Subtracts second operand from the first	A - B will give -5
*	Multiply both operands	A * B will give 50
/	Divide numerator by denominator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B MOD A will give 0
^	Exponentiation Operator	B^A will give 100000

The Comparison Operators

VBScript supports the following comparison operators: Assume variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is False.
<>	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A <> B) is True.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is False.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is True.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then conditionbecomes true.	(A >= B) is False.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is True.

The Logical Operators

VBScript supports the following logical operators: Assume variable A holds 10 and variable B holds 0, then:

Operator	Description	Example
AND	Called Logical AND operator. If both the conditions are True then Expression becomes true.	a<>0 AND b<>0 is False.
OR	Called Logical OR Operator. If any of the two conditions are True then condition becomes true.	a<>0 OR b<>0 is true.
NOT	Called Logical NOT Operator. Use to reverses thelogical state of its operand. If a condition is true then Logical NOT operator will make false.	NOT(a<>0 OR b<>0) is false.
XOR	Called Logical Exclusion. It is the combination of NOT and OR Operator. If one, and only one, of the expressions evaluates to True, result is True.	(a<>0 XOR b<>0) is false.

The Concatenation Operators

VBScript supports the following Concatenation operators: Assume variable A holds 5 and variable B holds 10 then:

Operator	Description	Example
+	Adds two Values as Variable Values are Numeric	A + B will give 15
&	Concatenates two Values	A & B will give 510

Concatenation can also be used for concatenating two strings. Assume variable A="Microsoft" and variable B="VBScript" then:

Operator	Description	Example
+	Concatenates two Values	A + B will give MicrosoftVBScript
&	Concatenates two Values	A & B will give MicrosoftVBScript

Result:

The essential aspect of vbscript has been studied

Practical No 8

AIM:

(a) : Write the script to generate Fibonacci Series in VBScript.

(b) : Write VB Script to generate Date and Time in Different Format.

Program: *script to generate Fibonacci Series*

```
<html>
<body>
<p> this is my vbscript program</p>
<center>
     <script language="vbscript" type="text/vbscript">
     dim a,b,c
     a=0
     b=1
     c=0
     document.write(a&"<br/>")
     document.write(b&"<br/>")
     do while c<=21
          c=a+b
          a=b
          b=c
          document.write(c&"<br/>")
     loop
     </script>
</center>
</body>
</html>
```

Output:

- *Script to generate Date and Time in Different Format*

```
<HTML>
<BODY>
<script LANGUAGE="VBSCRIPT" type="text/vbscript">
document.write("The current date is " & Date & "<br />")
document.write("The current month is " & Month(Date)& "<br />")
document.write("The current year is " & Year(Date)& "<br />")
document.write("The current Day is " & Day(Date)&  "<br />")
document.write("Date is IsDatefunction is " & IsDate(Date) & "<br />")
document.write("'January first eighteen hundred' a valid date format: " &
IsDate("January first eighteen hundred")& "<br />")
document.write("The time is now " & Time & "<br />")
document.write("The current minute is " & Minute(Time) & "<br />")
document.write("The current hour is " & Hour(Time) & "<br />")
document.write("The current second is " & Second(Time) & "<br />")
document.write("Number of seconds elapsed in the current day: " & Timer&
"<br />")
document.write("Date and time: " & Now& "<br />")
document.write("Current date: " & FormatDateTime(Date, 0) & "<br />")
document.write("Current date: " & FormatDateTime(Date, 1) & "<br />")
document.write("Current date: " & FormatDateTime(Date, 2) & "<br />")
document.write("Current time: " & FormatDateTime(Time, 3) & "<br />")
document.write("Current time: " & FormatDateTime(Time, 4))
```

```
</script>
<p>
0    display a date in the format mm/dd/yy (default)<br>
1    display a date in the format day, month and month day, year<br>
2    display a date in the format mm/dd/yy (default (same as 0))<br>
3    display time in the format hh:mm:ss: PM/AM (12 hour format)<br>
4    display time in the format hh:mm (24 hour format)
</p>
</BODY>
</HTML>
```

Output:

Practical No. 9

AIM: To Study JAVA Script.

THEORY:

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

JavaScript Basics

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

JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Features of java script

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform.

Client-Side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Advantages of JavaScript

The merits of using JavaScript are:

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

- Immediate feedback to the visitors: They don't have to wait for a page reload to see if they have forgotten to enter something.
- Increased interactivity: You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- Richer interfaces: You can use JavaScript to include such items as drag-and- drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript

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

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

JavaScript Development Tools

One of major strengths of JavaScript is that it does not require expensive development tools. You can start with a simple text editor such as Notepad. Since it is an interpreted language inside the context of a web browser, you don't even need to buy a compiler. To make our life simpler, various vendors have come up with very nice JavaScript editing tools. Some of them are listed here:

- Microsoft FrontPage: Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
- Macromedia Dreamweaver MX: Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.
- Macromedia HomeSite 5: HomeSite 5 is a well-liked HTML and JavaScript editor from Macromedia that can be used to manage personal websites effectively.

Java script – syntax

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

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

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

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

```
</script>
```

The script tag takes two important attributes:

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

So your JavaScript syntax will look as follows.

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

First JavaScript Code

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a `"//-->"`. Here `"//"` signifies a comment in JavaScript, so we add that to prevent a browser from reading

the end of the HTML comment as a piece of JavaScript code. Next, we call a function document.write which writes a string into our HTML document.

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

body>

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

This code will produce the following result

```
Hello  World!
```

Whitespace and Line Breaks

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

Semicolons are Optional

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

```
<script  language="javascript"  type="text/javascript">
<!--
var1  = 10
var2  = 20
//-->
</script>
```

But when formatted in a single line as follows, you must use semicolons:

```
<script  language="javascript"  type="text/javascript">
<!--
var1  =  10;  var2  =  20;
//-->
</script>
```

Note: It is a good programming practice to use semicolons

Case Sensitivity

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

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

NOTE: Care should be taken while writing variable and function names in JavaScript

Comments in JavaScript

JavaScript supports both C-style and C++-style comments. Thus:

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

JAVASCRIPT – ENABLING

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

JavaScript in Internet Explorer

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

- Follow Tools -> Internet Options from the menu.

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

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

Javascript – placement

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows:

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

If you need a script to run as the page loads so that the script generates content in the page, then the script goes in the <body> portion of the document. In this case, you would not have any function defined using JavaScript. Take a look at the following code.

```
<html>
<head>
</head>
<body>
<script  type="text/javascript">
<!--
document.write("Hello  World")
//-->
</script>
<p>This  is  web  page  body  </p>
</body>
</html>
```

This code will produce the following results:

```
Hello  World
```

JAVASCRIPT – VARIABLES

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container. Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword

JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- Global Variables: A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- Local Variables: A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable.

Javascript – operators

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

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Let's have a look at all the operators one by one.

Arithmetic Operators

JavaScript supports the following arithmetic operators: Assume variable A holds 10 and variable B holds 20, then:

S. No.	Operator and Description
1	+ (Addition) Adds two operands Ex: A + B will give 30
2	- (Subtraction) Subtracts the second operand from the first Ex: A - B will give -10
3	* (Multiplication) Multiply both operands Ex: A * B will give 200

4	/ (Division) Divide the numerator by the denominator Ex: B / A will give 2
5	% (Modulus) Outputs the remainder of an integer division Ex: B % A will give 0
6	++ (Increment) Increases an integer value by one Ex: A++ will give 11
7	-- (Decrement) Decreases an integer value by one Ex: A-- will give 9

Comparison Operators

JavaScript supports the following comparison operators: Assume variable A holds 10 and variable B holds 20, then:

S.No	Operator and Description
1	== (Equal) Checks if the value of two operands are equal or not, if yes, then the condition becomes true. Ex: (A == B) is not true.
2	!= (Not Equal) Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. Ex: (A != B) is true.

3	> (Greater than) Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. Ex: (A > B) is not true.
4	< (Less than) Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. Ex: (A < B) is true.
5	>= (Greater than or Equal to) Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A >= B) is not true.
6	<= (Less than or Equal to) Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A <= B) is true.

Logical Operators

JavaScript supports the following logical operators: Assume variable A holds 10 and variable B holds 20, then:

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

Bitwise Operators

JavaScript supports the following bitwise operators: Assume variable A holds 2 and variable B holds 3, then:

S.No	Operator and Description
1	& (Bitwise AND) It performs a Boolean AND operation on each bit of its integer arguments. Ex: (A & B) is 2.
2	| (BitWise OR) It performs a Boolean OR operation on each bit of its integer arguments. Ex: (A | B) is 3.
3	^ (Bitwise XOR)
	It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both. Ex: (A ^ B) is 1.
4	~ (Bitwise Not) It is a unary operator and operates by reversing all the bits in the operand. Ex: (~B) is -4.
5	<< (Left Shift) It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on. Ex: (A << 1) is 4.
6	>> (Right Shift) Binary Right Shift Operator. The left operand’s value is moved right by the number of bits specified by the right operand. Ex: (A >> 1) is 1.
7	>>> (Right shift with Zero) This operator is just like the >> operator, except that the bits shifted in on the left are always zero. Ex: (A >>> 1) is 1.

Assignment Operators

JavaScript supports the following assignment operators:

S.No	Operator and Description
1	= (Simple Assignment) Assigns values from the right side operand to the left side operand

Conditional statements

JavaScript supports the following forms of if..else statement –

- if statement
- if...else statement
- if...else if... statement.

if statement

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

The syntax for a basic if statement is as follows –

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

Here a JavaScript expression is evaluated. If the resulting value is true, the given statement(s) are executed. If the expression is false, then no statement would be executed.

if...else statement:

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

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

else{
   Statement(s) to be executed if expression is false
}
```

Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

if...else if... statement

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

The syntax of an if-else-if statement is as follows −

```
if (expression 1){
   Statement(s) to be executed if expression 1 is true
}
else if (expression 2){
   Statement(s) to be executed if expression 2 is true
}
else if (expression 3){
   Statement(s) to be executed if expression 3 is true
}
else{
   Statement(s) to be executed if no expression is true
}
```

There is nothing special about this code. It is just a series of if statements, where each if is a part of the else clause of the previous statement. Statement(s) are executed based on the true condition, if none of the conditions is true, then the else block is executed.

The while Loop

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

The syntax of while loop in JavaScript is as follows −

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

The do...while Loop

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

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

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

For Loop

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

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

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

The syntax of for loop is JavaScript is as follows −

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

Result:

The essential aspect of javascript has been studied

Practical No 10

AIM:

(a) : Java Script example program for Fibonacci series.

(b) : Java Script program to check odd or even numbers with example.

Program: *script to generate Fibonacci Series*

```
<html>
 <head><title>Fibonacci Series</title></head>
 <body>
  <script type="text/javascript">
  <!—
       var var1 = 0;
       var var2 = 1;
       var var3;
       var num = prompt("Enter the limit to generate fibonacci no");
       document.write(var1+"<br />");
       document.write(var2+"<br />");
       for(var i=3; i <= num;i++)
       {
            var3 = var1 + var2;
            var1 = var2;
            var2 = var3;
       document.write(var3+"<br />");
       }
  // -->
  </script>
 </body>
</html>
```

Program: *Script program to check odd or even numbers with example.*

```
<html>
   <body>
      <script type="text/javascript">
         <!--
            var i;
  for (i=1; i<=15; i++)
 {
               if( i % 2 == 0 ){
               document.write(i+ "<b> is Even Numver</b><br>");
            }
            else{
               document.write(i+ "<b> is Odd number</b><br>");
            }
 }
         //-->
      </script>
   </body>
</html>
```

Practical No 11

AIM: To study Unix

Theory:

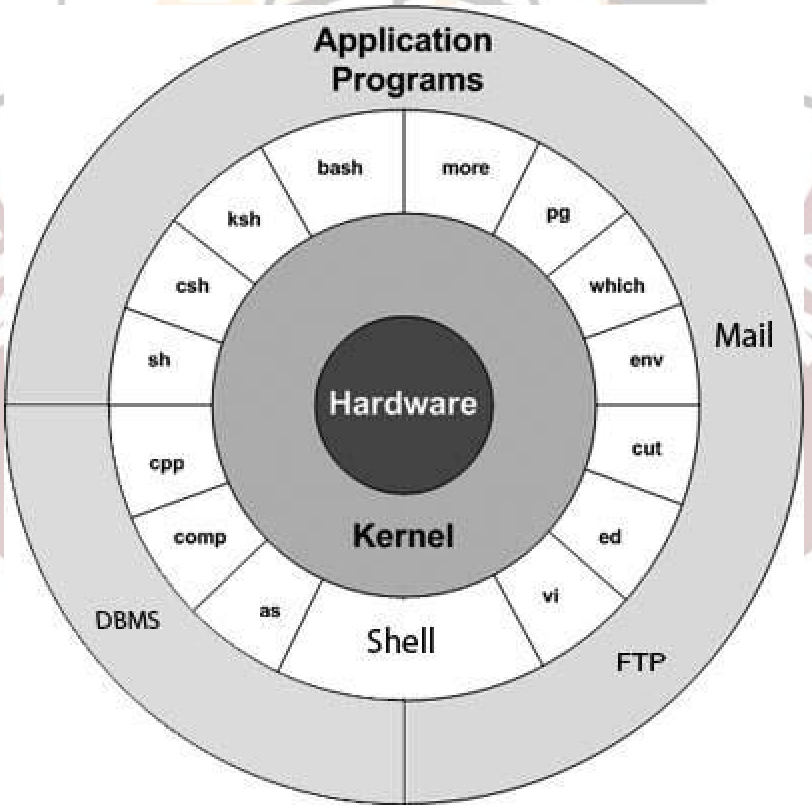
The Unix operating system is a set of programs that act as a link between the computer and the user.The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the operating system or the kernel.

Users communicate with the kernel through a program known as the shell. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

- Unix was originally developed in 1969 by a group of AT&T employees Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna at Bell Labs.
- There are various Unix variants available in the market. Solaris Unix, AIX, HP Unix and BSD are a few examples. Linux is also a flavor of Unix which is freely available.
- Several people can use a Unix computer at the same time; hence Unix is called a multiuser system.
- A user can also run multiple programs at the same time; hence Unix is a multitasking environment.

Unix Architecture

Here is a basic block diagram of a Unix system −



The main concept that unites all the versions of Unix is the following four basics −

- Kernel − The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.

- Shell − The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.
- A shell is a program whose primary purpose is to read commands and run other programs.
- The shell's main advantages are its high action-to-keystroke ratio, its support for automating repetitive tasks, and its capacity to access networked machines.
- The shell's main disadvantages are its primarily textual nature and how cryptic its commands and operation can be.
- Commands and Utilities − There are various commands and utilities which you can make use of in your day to day activities. cp, mv, catand grep, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.
- Files and Directories − All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the filesystem.

In Unix, there are three basic types of files −

- **Ordinary Files** − An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** − Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** − Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Listing Files

To list the files and directories stored in the current directory, use the following command −

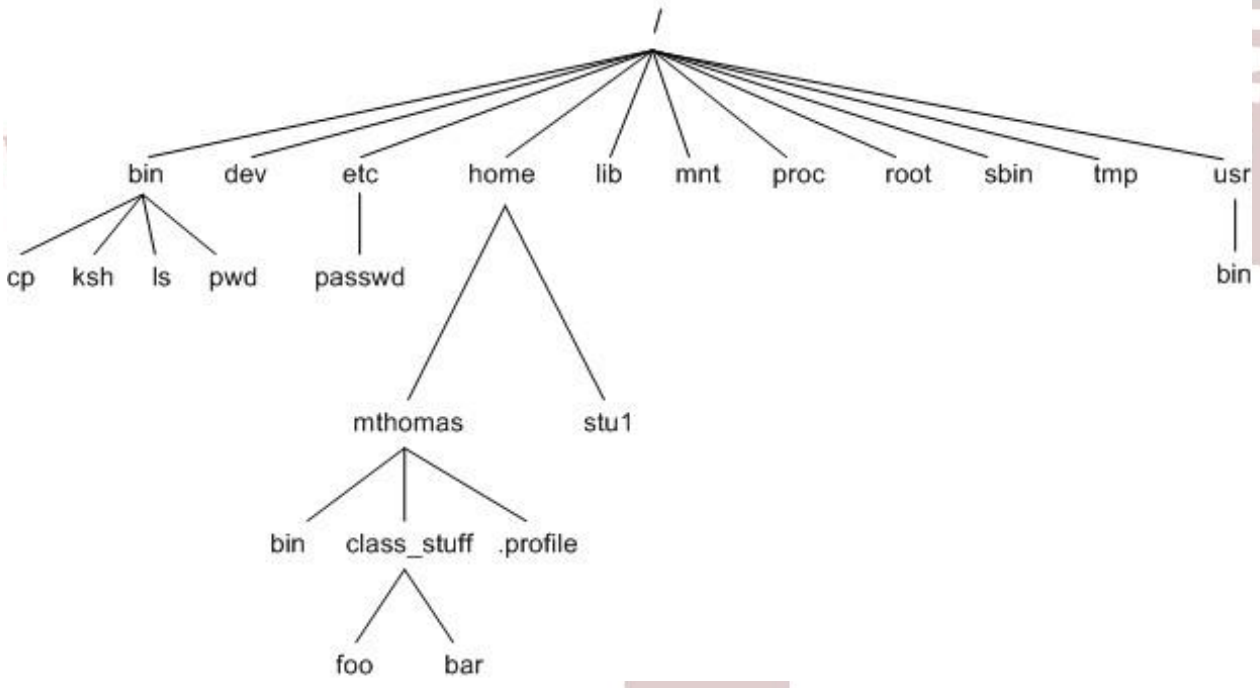
- The file system is responsible for managing information on the disk.
- Information is stored in files, which are stored in directories (folders).
- Directories can also store other directories, which forms a directory tree.
- `cd` path changes the current working directory.
- `ls` path prints a listing of a specific file or directory; ls on its own lists the current working directory.
- `pwd` prints the user's current working directory.
- `/` on its own is the root directory of the whole file system.
- A relative path specifies a location starting from the current location.

- An absolute path specifies a location from the root of the file system.
- Directory names in a path are separated with / on Unix, but \ on Windows.
- .. means ‘the directory above the current one’; . on its own means ‘the current directory’.
- Most files’ names are something .extension. The extension isn’t required, and doesn’t guarantee anything, but is normally used to indicate the type of data in the file.

Working With Files and Directories

- `cp old new` copies a file.
- `mkdir path` creates a new directory.
- `mv old new` moves (renames) a file or directory.
- `rm path` removes (deletes) a file.
- `*` matches zero or more characters in a filename, so *.txt matches all files ending in .txt.
- `?` matches any single character in a filename, so ?.txt matches a.txt but not any.txt.
- Use of the Control key may be described in many ways, including Ctrl-X, Control-X, and ^X.
- The shell does not have a trash bin: once something is deleted, it’s really gone.
- Depending on the type of work you do, you may need a more powerful text editor than Nano.

File Structure and Directory Structure



While this diagram is not all inclusive, the following system files (i.e. directories) are present in most Unix filesystems:

- bin - short for binaries, this is the directory where many commonly used executable commands reside
- dev - contains device specific files
- etc - contains system configuration files

- home - contains user directories and files
- lib - contains all library files
- mnt - contains device files related to mounted devices
- proc - contains files related to system processes
- root - the root users' home directory (note this is different than /)
- sbin - system binary files reside here. If there is no sbin directory on your system, these files most likely reside in etc
- tmp - storage for temporary files which are periodically removed from the filesystem
- usr - also contains executable commands

Result:

The essential aspect of Unix has been studied

Practical No 12

AIM: To study Linux

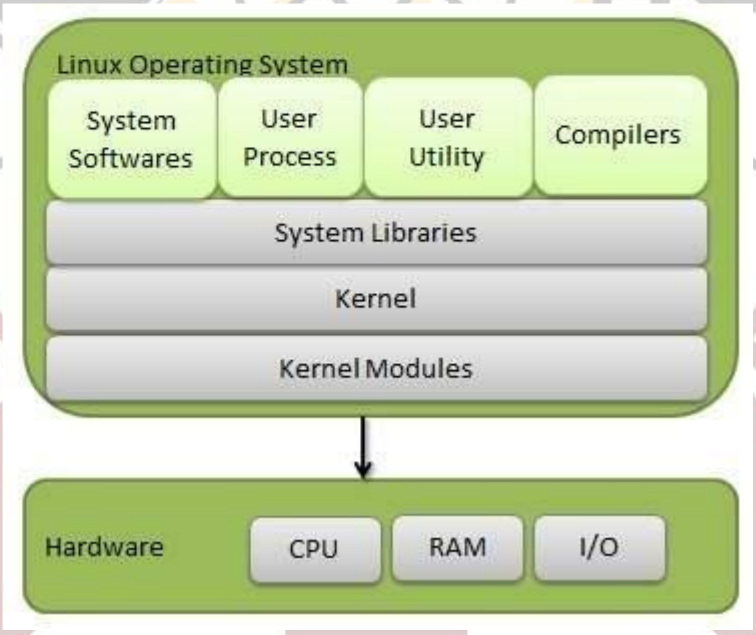
Theory:

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System

Linux Operating System has primarily three components

- **Kernel** − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Library** − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
- **System Utility** − System Utility programs are responsible to do specialized, individual level tasks.



Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

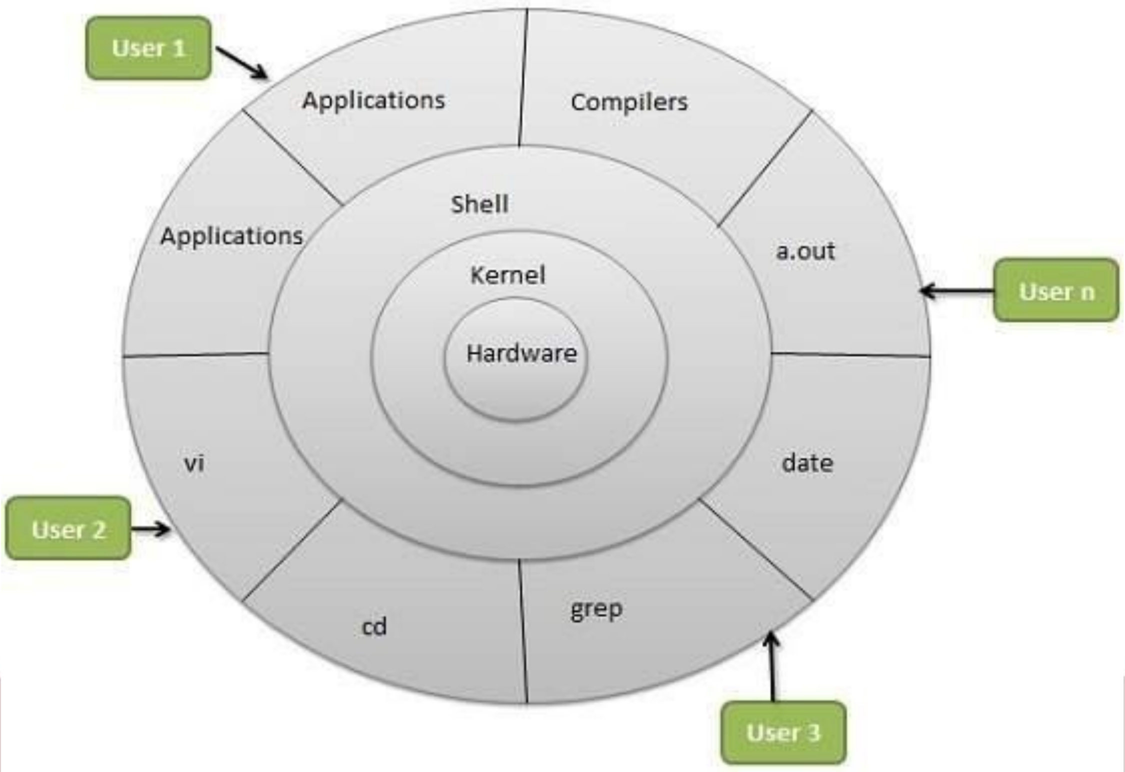
Basic Features

Following are some of the important features of Linux Operating System.

- **Portable** − Portability means software can works on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- **Open Source** − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** − Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** − Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** − Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
- **Security** − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Architecture

The following illustration shows the architecture of a Linux system −



The architecture of a Linux System consists of the following layers −

- **Hardware layer −** Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel −** It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell −** An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities −** Utility programs that provide the user most of the functionalities of an operating systems.

File Handling commands

- mkdir – make directories

  Usage: mkdir [OPTION] DIRECTORY... eg. mkdir prabhat
- ls – list directory contents

  Usage: ls [OPTION]... [FILE]... eg. ls, ls l, ls prabhat
- cd – changes directories

  Usage: cd [DIRECTORY] eg. cd prabhat
- pwd print name of current working directory Usage: pwd
- vim – Vi Improved, a programmers text editor

  Usage: vim [OPTION] [file]... eg. vim file1.txt
- cp – copy files and directories

  Usage: cp [OPTION]... SOURCE DEST eg. cp sample.txt sample_copy.txt

  cp sample_copy.txt target_dir

- mv – move (rename) files

  Usage: mv [OPTION]... SOURCE DEST eg. mv source.txt target_dir

  mv old.txt new.txt
- rm remove files or directories

  Usage: rm [OPTION]... FILE... eg. rm file1.txt , rm rf some_dir
- find – search for files in a directory hierarchy

  Usage: find [OPTION] [path] [pattern] eg. find file1.txt, find name file1.txt
- history – prints recently used commands Usage: history

File and Directory Structure

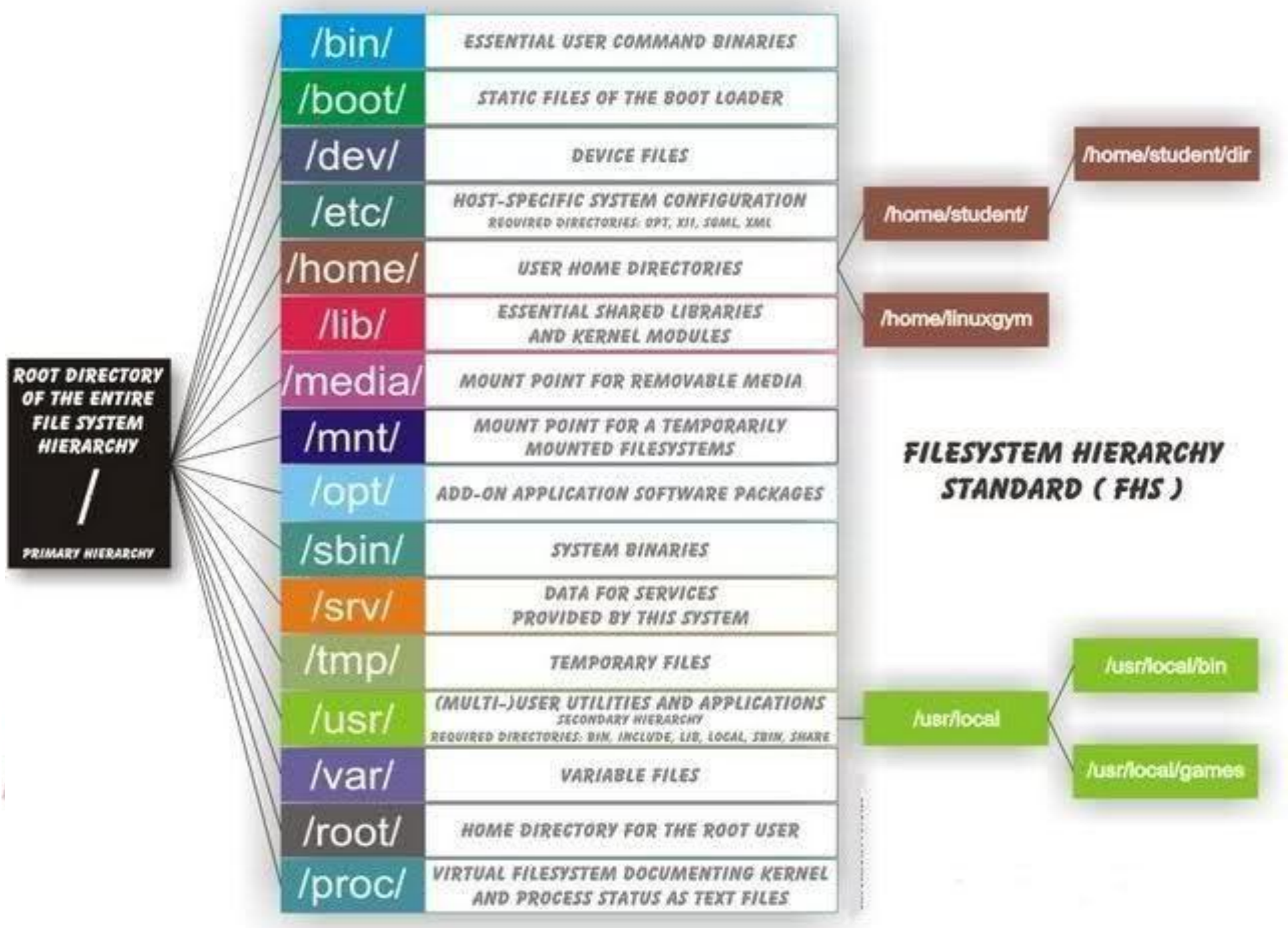
let us review the Linux filesystem structures and understand the meaning of individual high-level directories.

1. / – Root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /.

2. /bin – User Binaries

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.


ROOT DIRECTORY OF THE ENTIRE FILE SYSTEM HIERARCHY
/
PRIMARY HIERARCHY
/bin/
ESSENTIAL USER COMMAND BINARIES
/boot/
STATIC FILES OF THE BOOT LOADER
/dev/
DEVICE FILES
/etc/
HOST-SPECIFIC SYSTEM CONFIGURATION
REQUIRED DIRECTORIES: OPT, X11, SGML, XML
/home/
USER HOME DIRECTORIES
/lib/
ESSENTIAL SHARED LIBRARIES AND KERNEL MODULES
/media/
MOUNT POINT FOR REMOVABLE MEDIA
/mnt/
MOUNT POINT FOR A TEMPORARILY MOUNTED FILESYSTEMS
/opt/
ADD-ON APPLICATION SOFTWARE PACKAGES
/sbin/
SYSTEM BINARIES
/srv/
DATA FOR SERVICES PROVIDED BY THIS SYSTEM
/tmp/
TEMPORARY FILES
/usr/
(MULTI-)USER UTILITIES AND APPLICATIONS
SECONDARY HIERARCHY
REQUIRED DIRECTORIES: BIN, INCLUDE, LIB, LOCAL, SBIN, SHARE
/var/
VARIABLE FILES
/root/
HOME DIRECTORY FOR THE ROOT USER
/proc/
VIRTUAL FILESYSTEM DOCUMENTING KERNEL AND PROCESS STATUS AS TEXT FILES
/home/student/
/home/student/dir
/home/linuxgym
FILESYSTEM HIERARCHY STANDARD (FHS)
/usr/local
/usr/local/bin
/usr/local/games

3. /sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.
- But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid}

directory contains information about the process with that particular pid.

- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

7. /var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

9. /usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*
- For example: ld-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

14. /mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

16. /srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data.

Result:

The essential aspect of Linx has been studied