



LAB MANUAL

SECTION 1

Introduction to HTML

SECTION 2

Advanced HTML

SECTION 3

Introduction to JavaScript

SECTION 4

Dynamic Website Creation

PROGRAMME DESIGN COMMITTEE

Prof. Sanjeev K. Aggarwal, IIT, Kanpur
Prof. M. Balakrishnan, IIT, Delhi
Prof. Harish Karnick, IIT, Kanpur
Prof. C. Pandurangan, IIT, Madras
Dr. Om Vikas, Sr. Director, MIT
Prof. P. S. Grover, Sr. Consultant,
SOCIS, IGNOU
Prof. H.M. Gupta Dept. of Elect. Engg. IIT,
Delhi
Prof. M.N. Doja, Dept. of CE Jamia Millia,
New Delhi
Prof. C. Pandurangan Dept. of CSE, IIT,
Chennai
Prof. L. Ramesh Babu Dept. of CSE Acharya
Nagarjuna University Nagarjuna Nagar (AP)
Prof. N.S. Gill Dept. of CS, MDU, Rohtak
Prof. Arvind Kalia, Dept. of CS
HP University, Shimla

Prof. Anju Sahgal Gupta SOH, IGNOU, New Delhi
Prof. Sujatha Verma SOS, IGNOU, New Delhi
Prof. V. Sundarapandian IITMK, Trivandrum
Prof. Dharamendra Kumar
Dept. of CS, GJU, Hissar
Prof. Vikram Singh Dept. of CS, CDLU, Sirsa
Dr. P.K. Mishra, Associate Prof. Dept. of CS, BHU,
Varanasi
Prof. M.P. Mishra, SOCIS, IGNOU, New Delhi
Dr. Naveen Kumar, Reader SOCIS, IGNOU, New
Delhi
Dr. Subodh Kesharwani, Asst. Prof. SOMS, IGNOU,
New Delhi
Prof. P.V. Suresh, SOCIS, IGNOU
Prof. V.V. Subrahmanyam SOCIS, IGNOU
Prof. Akshay Kumar SOCIS, IGNOU
Shri Shashi Bhushan, SOCIS, IGNOU
Prof. Manohar Lal, SOCIS, IGNOU

COURSE DESIGN COMMITTEE

Shri Shashi Bhushan, SOCIS, IGNOU
Prof. Akshay Kumar SOCIS, IGNOU
Prof. R.S. Gupta, New Delhi
Prof. Sujatha Verma, SOS, IGNOU, New Delhi
Sh. Milind Mahajani, New Delhi
Dr. D.K. Lobiyal, SC & SS, JNU

Dr. Pravin Chandra IIC, DU, New Delhi
Prof. P.V. Suresh, SOCIS, IGNOU
Prof. V.V. Subrahmanyam SOCIS, IGNOU
Dr. Naveen Kumar, Reader SOCIS, IGNOU, New
Delhi
Prof. M.P. Mishra, SOCIS, IGNOU, New Delhi

FACULTY OF SOCIS

Prof. Sandeep Singh Rawat, Director, SOCIS,
IGNOU
Prof. P. Venkata Suresh, SOCIS, IGNOU
Prof. V.V. Subrahmanyam, SOCIS, IGNOU
Prof. Akshay Kumar, SOCIS, IGNOU

Prof. M.P. Mishra, SOCIS, IGNOU
Prof. Divakar Yadav, SOCIS, IGNOU
Dr. Sudhansh Sharma, Associate Professor, SOCIS,
IGNOU

PREPARATION TEAM

Adopted Section – 1 From MSCL-016 Block – 1 Unit – 2
Adopted Section – 2 From MSCL-016 Block – 1 Unit – 3
Adopted Section – 3 From MSCL-016 Block – 1 Unit – 4
Adopted Section – 4 From BCSL – 057 Lab Manual

Course Coordinator: Prof. Sandeep Singh Rawat, SOCIS, IGNOU, New Delhi

PRINT PREPARATION TEAM

Print Production

Registrar (Publication)

November, 2025

©Indira Gandhi National Open University, 2004 ISBN – 81-266-1253-3

All rights reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from the Indira Gandhi National Open University.

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110 068.

Printed and published on behalf on the Indira Gandhi National Open University, New Delhi by The Director, SOCIS.



ignou
THE PEOPLE'S
UNIVERSITY

COURSE INTRODUCTION

The **Web Programming Lab (BCSL-147)** is designed to provide learners with **hands-on experience** in developing both static and dynamic web pages using foundational web technologies such as **HTML**, **Cascading Style Sheets (CSS)**, and **JavaScript**. The course also introduces learners to modern web development tools and frameworks, providing a practical foundation for building interactive, responsive, and data-driven applications.

This lab course complements the theory course **BCS-053: Web Programming**, and emphasizes **skill-based learning** through structured lab sessions. Learners are encouraged to develop creativity, logical thinking, and problem-solving abilities while working on real-world web development tasks.

Each lab session in this course is of **three hours' duration** and focuses on progressively advanced aspects of web programming — from basic HTML structure and styling to the integration of scripting, interactivity, and simple database connectivity. The course encourages learners to use open-source tools and Integrated Development Environments (IDEs) such as **NetBeans**, **Eclipse**, or **Visual Studio Code** for implementation and testing.

Course Sections

The course comprises **four sections**, each focusing on key components of web programming:

- **Section 1:** This section introduces **HTML**, the foundational language for creating web pages. It explains how web servers deliver files to browsers in a structured format using HTML. Learners will explore a wide range of **HTML tags** and their attributes to create and format text, images, tables, and links.
- **Section 2:** This section describes how to design **interactive web pages** using features such as **frames, tables, and forms** for accepting user input. It enables learners to structure data and layout web content effectively for usability and engagement.
- **Section 3:** This section focuses on **JavaScript**, a scripting language primarily used for adding interactivity to web pages and validating user input in forms. Learners will study the object-based nature of JavaScript, learn to embed scripts within HTML documents, and create dynamic, interactive web pages.
- **Section 4:** This section integrates **JavaScript, XML, and JSP** to develop **dynamic, data-driven web applications**. Learners will gain practical exposure to creating simple websites, validating and processing data, and connecting applications to databases using tools like **NetBeans IDE**. The Section also introduces **JSTL** and techniques for database integration, enabling learners to build complete, functional web applications.

Learning Outcomes

By the end of the course, learners will be able to:

- Understand the structure and functionality of web technologies and web servers.
- Design and develop well-structured, interactive, and accessible web pages using **HTML5**, **CSS3**, and **JavaScript**.
- Apply client-side scripting for **form validation** and **dynamic content generation**.
- Integrate **basic database operations** into web applications using suitable server-side technologies and tools.
- Adopt **best practices** in web standards, usability, and responsive design.

Recommended Learning Resources

To keep pace with evolving technologies, learners are encouraged to explore the following reliable web resources and developer communities:

- <https://developer.mozilla.org>
- <https://www.w3schools.com>
- <https://www.netbeans.org>
- <https://www.oracle.com>
- <https://www.eclipse.org>
- <https://www.github.com>

This course serves as an **entry point for mastering modern web technologies**. Learners are encouraged to continue exploring beyond the scope of the lab to gain proficiency in advanced frameworks such as **React**, **Angular**, **Node.js**, or **Django** as they progress in their professional journey.

The **Web Programming Lab (BCSL-147)** thus aims to equip learners with the **practical knowledge, confidence, and creativity** required to design and develop robust, efficient, and innovative web solutions for the digital era.

SECTION 1 INTRODUCTION TO HTML

Structure	Page Nos.
1.0 Introduction	7
1.1 Objectives	8
1.2 What is HTML?	8
1.3 Basic Tags of HTML	9
1.3.1 HTML Tag	10
1.3.2 TITLE Tag	10
1.3.3 BODY Tag	11
1.4 Formatting of Text	13
1.4.1 Headers	13
1.4.2 Formatting Tags	13
1.4.3 PRE Tag	16
1.4.4 FONT Tag	17
1.4.5 Special Characters	18
1.5 Working with Images	21
1.6 META Tag	23
1.7 Summary	26
1.8 Solutions/ Answers	26
1.9 Exercises For Practice in Lab Session	32

1.0 INTRODUCTION

You would by now have been introduced to the Internet and the World Wide Web (often just called the Web) and how it has changed our lives. Today we have access to a wide variety of information through Web sites on the Internet. We can access a Web site if we have a connection to the Internet and a browser on our computer. Popular browsers are Microsoft Internet Explorer, Netscape Navigator and Opera. When you connect to a Web site, your browser is presented with a file in a special format by the Web server on the remote computer. The contents of the file are stored in a special format using Hyper Text Markup Language, often called HTML. This format is rendered, or interpreted, by the browser and you then see the page of the web site from your computer.

HTML is one language in a class of markup languages, the most general form of which is Standard Generalized Markup Language, or SGML. Since SGML is complex, HTML was invented as a simple way of creating web pages that could be easily accessed by browsers. HTML is a special case of SGML.

HTML consists of tags and data. The tags serve to define what kind of data follows them, thereby enabling the browser to render the data in the appropriate form for the user to see. There are many tags in HTML, of which the few most important ones are introduced in this Section. HTML files usually have the extension “.htm” or “.html”.

If you want to create Web pages, you need a tool to write the HTML code for the page. This can be a simple text editor if you are hand-coding HTML. You also have sophisticated HTML editors available that automate many (though not all) of the tasks of coding HTML. You also need a browser to be able to render your code so that you can see the results.

1.1 OBJECTIVES

After going through this Section you should be able to learn:

- basic concepts of HTML;
- basic tags of HTML;
- how to control text attributes such as the font;
- how to work with images in HTML; and
- significance of Meta Tag

The section covers only the simpler concepts of HTML and does not by any means deal with the subject comprehensively.

1.2 WHAT IS HTML?

As indicated earlier, HTML stands for HyperText Markup Language. HTML provides a way of displaying Web pages with text and images or multimedia content. HTML is not a programming language, but a markup language. An HTML file is a text file containing small markup tags. The markup tags tell the Web browser, such as Internet Explorer or Netscape Navigator, how to display the page. An HTML file must have an htm or html file extension. These files are stored on the web server. So if you want to see the web page of a company, you should enter the URL (Uniform Resource Locator), which is the web site address of the company in the address bar of the browser. This sends a request to the web server, which in turn responds by returning the desired web page. The browser then renders the web page and you see it on your computer.

HTML allows Web page publishers to create complex pages of text and images that can be viewed by anyone on the Web, regardless of what kind of computer or browser is being used. Despite what you might have heard, you don't need any special software to create an HTML page; all you need is a word processor (such as Microsoft Word) and a working knowledge of HTML. Fortunately, the basics of HTML are easy to master. However, you can greatly relieve tedium and improve your productivity by using a good tool. A simple tool is Microsoft FrontPage that reduces the need to remember and type in HTML tags. Still, there can always be situations where you are forced to handcode certain parts of the web page.

HTML is just a series of tags that are integrated into a document that can have text, images or multimedia content. HTML tags are usually English words (such as blockquote) or abbreviations (such as p for paragraph), but they are

distinguished from the regular text because they are placed in small angle brackets. So the paragraph tag is `<p>`, and the blockquote tag is `<blockquote>`. Some tags dictate how the page will be formatted (for instance, `<p>` begins a new paragraph), and others dictate how the words appear (`` makes text bold). Still others provide information - such as the title - that doesn't appear on the page itself. The first thing to remember about tags is that they travel in pairs. Most of the time that you use a tag - say `<blockquote>` - you must also close it with another tag - in this case, `</blockquote>`. Note the slash - / - before the word "blockquote"; that is what distinguishes a closing tag from an opening tag.

The basic HTML page begins with the tag `<html>` and ends with `</html>`. In between, the file has two sections - the header and the body.

The header - enclosed by the `<head>` and `</head>` tags - contains information about a page that will not appear on the page itself, such as the title. The body - enclosed by `<body>` and `</body>` - is where the action is. Everything that appears on the page is contained within these tags.

HTML pages are of two types:

- Static
- Dynamic

Static Pages

Static pages, as the name indicates, comprise static content (text or images). So you can only see the contents of a web page without being able to have any interaction with it.

Dynamic Pages

Dynamic pages are those where the content of the web page depend on user input. So interaction with the user is required in order to display the web page. For example, consider a web page which requires a number to be entered from the user in order to find out if it is even or odd. When the user enters the number and clicks on the appropriate button, the number is sent to the web server, which in turn returns the result to the user in an HTML page.

1.3 BASIC TAGS OF HTML

Let us now look at tags in more detail. A `<TAG>` tells the browser to do something. An `ATTRIBUTE` goes inside the `<TAG>` and tells the browser *how* to do it. A tag can have several attributes. Tags can also have default attributes. The *default value* is a value that the browser assumes if you have not told it otherwise. A good example is the font size. The default font size is 3. If you say nothing the size attribute of the font tag will be taken to have the value 3.

Consider the example shown in Fig. 1.1. Type the code specified in the figure in a text editor such as notepad and save it as "fig1.html". To render the file

and see your page you can choose one of two ways: 1) Find the icon of the html file you just made (fig1.htm) and double click on it. Or- 2) In Internet Explorer, click on File/Open File and point to the file (fig1.htm).

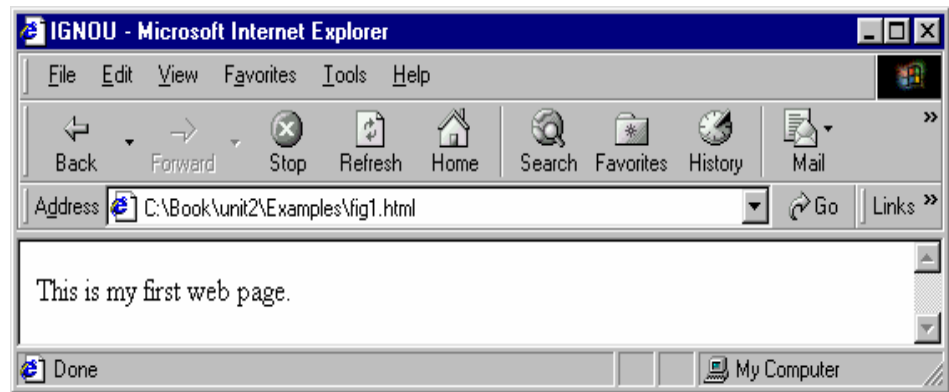


Figure 1.1: A Simple Web Page

```
<HTML>
<!-- This is a comment -->
<HEAD>
  <TITLE> IGNOU      </TITLE>
</HEAD>
<BODY>
  This is my first web page.
</BODY>
</HTML>
```

1.3.1 HTML Tag

As shown in Figure.1.1, <HTML> is a starting tag. To delimit the text inside, add a closing tag by adding a "/" to the starting tag. Most but not all tags have a closing tag. It is necessary to write the code for an HTML page between <HTML> and </HTML>.

Think of tags as talking to the browser or, better still, giving it instructions. What you have just told the browser is 'this is the start of an HTML document' (<HTML>) and 'this is the end of an HTML document' (</HTML>). Now you need to put some matter in between these two markers.

Every HTML document is segregated into a HEAD and BODY. The information about the document is kept within <HEAD> tag. The BODY contains the page content.

1.3.2 TITLE Tag

The only thing you have to concern yourselves with in the HEAD tag (for now) is the TITLE tag.

The bulk of the page will be within the BODY tag, as shown in Figure.1.1.

```
<HEAD>
  <TITLE> IGNOU </TITLE>
</HEAD>
```

Here the document has been given the title IGNOU. It is a good practice to give a title to the document created.

What you have made here is a skeleton HTML document. This is the minimum required information for a web document and all web documents should contain these basic components. Secondly, the document title is what appears at the very top of the browser window.

1.3.3 BODY Tag

If you have a head, you need a body. All the content to be displayed on the web page has to be written within the body tag. So whether text, headlines, textbox, checkbox or any other possible content, everything to be displayed must be kept within the BODY tag as shown in Figure 1.1.

Whenever you make a change to your document, just save it and hit the Reload/Refresh button on your browser. In some instances just hitting the Reload/Refresh button doesn't quite work. In that case hit Reload/Refresh while holding down the SHIFT key.

The BODY tag has following attributes:

- a. **BGCOLOUR:** It can be used for changing the background colour of the page. By default the background colour is white.
- b. **BACKGROUND:** It is used for specifying the image to be displayed in the background of the page.
- c. **LINK:** It indicates the colour of the hyperlinks, which have not been visited or clicked on.
- d. **ALINK:** It indicates the colour of the active hyperlink. An active link is the one on which the mouse button is pressed.
- e. **VLINK:** It indicates the colour of the hyperlinks after the mouse is clicked on it.
- f. **TEXT:** It is used for specifying the colour of the text displayed on the page.

Consider the following example:

```
<HTML>
<TITLE> IGNOU</TITLE>
<BODY    BGCOLOUR="#1234567"  TEXT = "#FF0000">
  Welcome to IGNOU
</BODY>
</HTML>
```

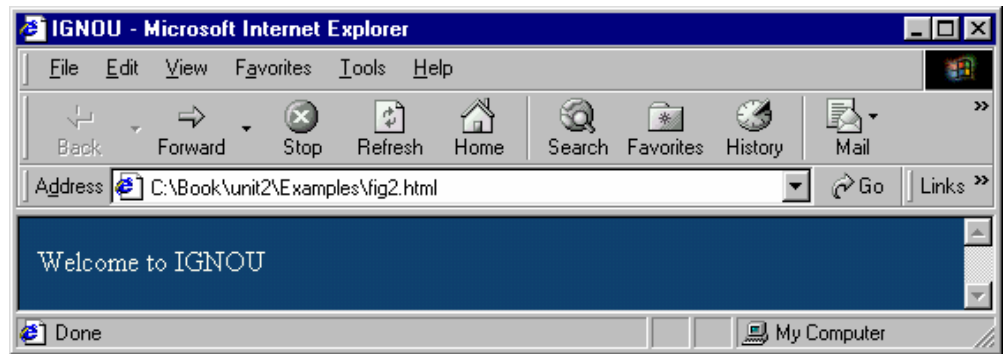


Figure 1.2: A Web Page with a Background Colour

The values specified for BGCOLOR and TEXT tags indicate the colour of the background of the page and that of the text respectively. These are specified in hexadecimal format. The range of allowable values in this format is from “#000000” to “#FFFFFF”. The “#” symbol has to precede the value of the colour so as to indicate to the browser that has to be interpreted as a hexadecimal value. In this six digit value, the first two digits specify the concentration of the colour red, the next two digits specify the concentration of the colour green and the last two digits specify the concentration of the colour blue. So the value is a combination of the primary colours red, green and blue and that is why it is called RGB colour. If we specify the value “#FF0000”, the colour appears to be red. “#000000” gives black and “#FFFFFF” gives the colour white. You also have the option of specifying the colour by giving its name, like:

<BODY TEXT = “WHITE”>.

You can also specify a background image instead. (Note that the image should be in the same folder as your HTML file. More on this below).

```
<HTML>
  <BODY BACKGROUND="swirlies.gif">
    Welcome to INDIA
  </BODY>
</HTML>
```

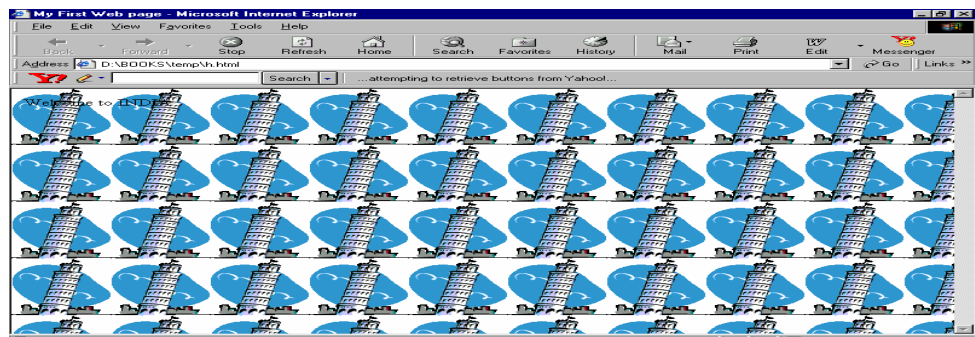


Figure 1.3: A Web Page with an Image in the Background

1.4 FORMATTING OF TEXT

Text formatting, in other words presenting the text on an HTML page in a desired manner, is an important part of creating a web page. Let us understand how we can lay out of text controls its appearance on a page.

1.4.1 Headers

Headers are used to specify the headings of sections or sub-sections in a document. Depending on the desired size of the text, any of six available levels (<H1> to <H6>) of headers can be used. Figure 1.4 shows the usage and varying size of the rendered text depending upon the tag used.

```
<HTML>
<HEAD>
  <TITLE> IGNOU</TITLE>
</HEAD>
<BODY>
  <CENTER>
    <H1> Header Level 1</H1>
    <H2> Header Level 2</H2>
    <H3> Header Level 3</H3>
    <H4> Header Level 4</H4>
    <H5> Header Level 5</H5>
    <H6> Header Level 6</H6>
  </CENTER>
</BODY>
</HTML>
```

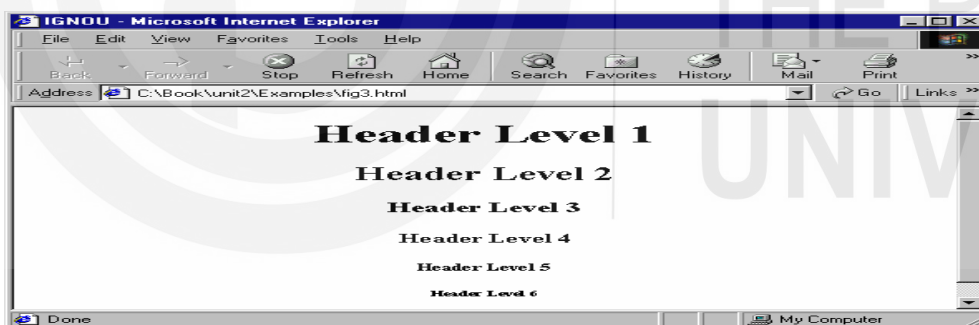


Figure 1.4: Rendering of the Six Header Levels

There is no predefined sequence for using the different levels of the header tags nor any restrictions on which one can be used. So the user has the option of using any level of header tag anywhere in the document. If you want to center text on a page, use the CENTER tag. The text written between <CENTER> and </CENTER> tag gets aligned in the center of the HTML page. As seen in Figure 1.4, the maximum size of the text is displayed using the <H1> tag. So the size goes in decreasing order with the increasing order of the level (i.e. From <H1> to <H6>).

1.4.2 Formatting Tags

Let us now look at some more tags that can be used to format text. These are all given in the example shown in Figure 1.5.

```
<HTML>
<HEAD>
<TITLE> IGNOU</TITLE>
</HEAD>
<BODY>
<B> IGNOU </B>
provides several <I> programmes </I>
in the <B><I>Computer </I></B> stream.
<P>
One of the <I> programmes </I> is <B><U> MCA</U></B>
</P>
<B>MCA </B> stands for <TT> Master Of Computer Applications </TT>
<BR>
<S>For MCA</S> <B> IGNOU </B> is considered to be one of the premier
universities.
<BR>
<STRONG>IGNOU</STRONG> believes in <STRONG><EM>
Quality</EM></STRONG> education
<BR>
<P>
According to <CITE> IGNOU, </CITE> <B> MCA<B> is one of its best
programmes offering convenient timings to the student so that s/he can pursue
the course while working at a job.
</P>
<BLOCKQUOTE>
For convenience all the courses offered by IGNOU can be seen on its website.
A student has also been provided the flexibility of seeing all the information
regarding admission to the next semester, examination result etc. on its
website.
</BLOCKQUOTE>
<HR NOSHADE>
<B> IGNOU contact details are :
<ADDRESS> IGNOU, <BR>
MAIDAN GARHI, <BR>
NEW DELHI – 110 068 <BR>
Website : www.ignou.ac.in
</ADDRESS>
</BODY>
</HTML>
```

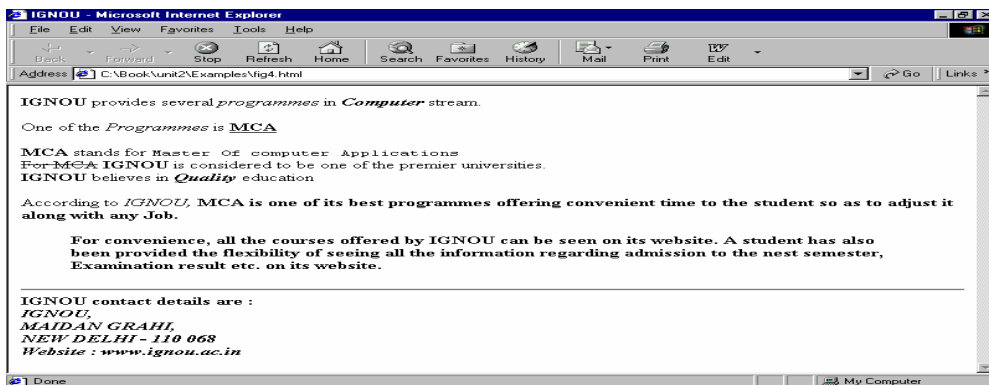


Figure 1.5: An Example showing Various Formatting Tags.

- a. **BOLD:** The text can be displayed in boldface by writing it in between the `` and `` tags.
- b. **ITALICS:** It starts with `<I>` and ends with `</I>` tag. It is used to display the text in italics.
- c. **UNDERLINE:** It is used for underlining the text to be displayed. The `<U>` tag is used for this purpose. These tags can be nested. So in order to see the text in boldface, in italics and underlined, it should be placed between the `<I><U>` and `</U></I>` tags. Note that the closing tags are written in reverse order, because any tag used within some other tag should be closed first.
- d. **PARAGRAPH:** If you want to display the text in the form of paragraphs, then the `<P>` tag should be used.
- e. **TT:** The `<TT>` tag is used for displaying text in a fixed width font similar to that of a typewriter.
- f. **STRIKE:** If you want the text to be marked with a strikethrough character, place it within the `<S>` and `</S>` tags.
- g. **STRONG:** There are certain text-based browsers that do not render the text as boldfaced. So you can use the `` tag instead of the `` tag. This displays the text to stand out in the most appropriate manner for the browser.
- h. **EM:** Just as the `` tag corresponds to the `` tag, the `` can be used instead of the `<I>` tag. The reason for using it is the same as for the `` tag. The `` tag is used for emphasizing the text in the manner most appropriate to the browser.
- i. **BR:** This tag is used for inserting a line break. The text written after the `
` tag is displayed from the beginning of the next line onwards. This tag does not need a corresponding terminating tag.
- j. **HR:** This tag puts a horizontal line on the page. It has the following attributes:

- **ALIGN:** It is used for aligning the line on the page. The possible values of this attribute are LEFT, RIGHT, and CENTER. It is useful when the width of the line is less than the width of the page.
 - **NOSHADE:** This attribute is used to prevent any shading effect.
 - **SIZE:** It is used for specifying the thickness of the line.
 - **WIDTH:** You can set the width of a line using this attribute. The value can be specified either in pixels or as a percentage of the width of the page, e.g., <HR WIDTH = “30%”>.
- k. **BLOCKQUOTE:** This tag indents the left margin of the text. It is used for displaying the text as quoted text as shown in Figure 1.5.
- l. **ADDRESS:** This tag, as shown in Figure 1.5, displays the text in italics.
- m. **CITE:** The text placed in between the <CITE> and </CITE> tags is rendered in italics by the browser.

1.4.3 PRE Tag

This tag is used to present the text exactly as written in the code, including whitespace characters. It is terminated by a </PRE> tag. Consider the example shown in Figure 1.6 to understand how this tag works.

```
<HTML>
<HEAD>
<TITLE>IGNOU</TITLE>
</HEAD>
<BODY>
<PRE>
```

IGNOU also offers a virtual campus. Studying through the virtual campus is a new concept in the field of education and this is the first such experiment in India.

While studying through the virtual campus mode, students have access to the following learning resources and experiences:

- Satellite based interactive tele-conferencing sessions.
- Viewing recorded video sessions.
- Computer based tutorials on CD-ROM.

Printed booklets for specific

```
</PRE>
</BODY>
</HTML>
```

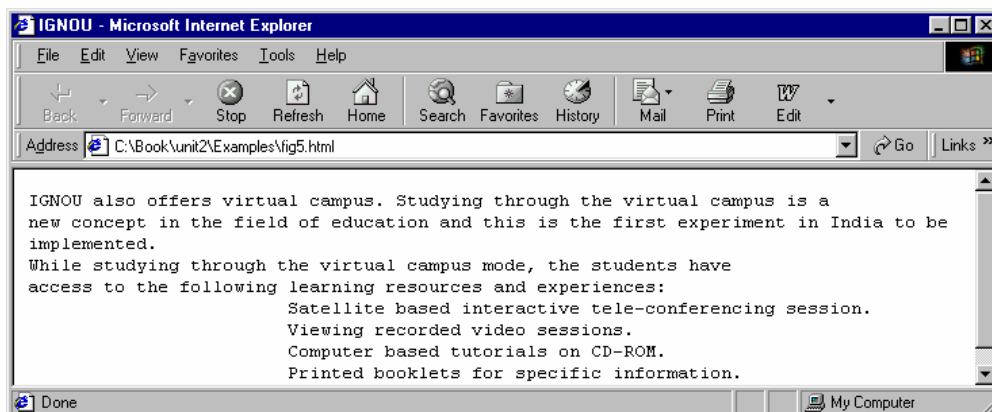


Figure 1.6: Presenting Preformatted Text

As shown in Figure 1.6, the format of the text presented in the browser remains the same as written in the code.

If we do not use the `<PRE>` tag, the browser condenses the white space when presenting the text on the web page.

1.4.4 FONT Tag

HTML provides the flexibility of changing the characteristics of the font such as size, colour etc. Every browser has a default font setting that governs the default font name, size and colour. Unless you have changed it, the default is Times New Roman 12pt with the colour being black. Now with IE 6.0 (Internet Explorer) you can specify font names other than the default, such as ARIAL and COMIC SANS. Consider the example shown in Figure 1.7.

```
<HTML>
<HEAD>
<TITLE> IGNOU </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<CENTER>
```

```
Welcome to <FONT SIZE=6>INDIA </FONT><BR>
Welcome to <FONT FACE = "ARIAL" SIZE=6>INDIA </FONT><BR>
Welcome to <FONT FACE = "ARIAL" SIZE=6 COLOUR = "BLUE">INDIA
</FONT><BR>
```

```
</CENTER>
</BODY>
</HTML>
```

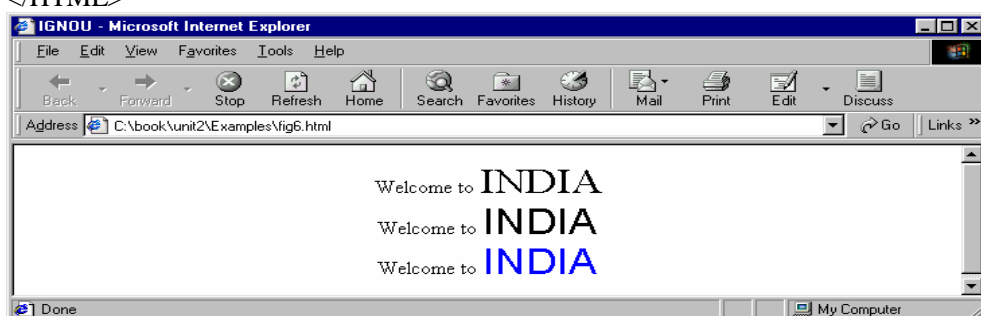


Figure 1.7: Using the FONT Tag

Let us look at the syntax of the tag with its different attributes.

The attributes are:

Tiny	Small	Regular	Extra Medium	Large	Real Big	Largest
1	2	3	4	5	6	7

c. **COLOUR:** With this attribute you can change the desired font colour. The values can be specified either by using the hexadecimal format as already described, i.e., #RGB, or by the name of the colour. The hex code of the colour has been explained earlier in this section. As shown in Figure 1.7, the value of the colour attribute in the third line has been specified as “Blue”. So the text present in the code between the and tags appears in blue. By default the colour of the text is black. If we specify the text colour in the tag then this value overrides the colour value, if any, specified in the <BODY> tag.

You have seen that there are certain characters that have special meaning in HTML code. For example, the “<” and “>” characters delimit tags. If you want to display such characters on the web page, you have to take care that the characters are not interpreted and are displayed literally. To display the “<” character, it can be specified as “<”. The “&” interprets the “lt” as the “<” symbol and displays it. But now what if you want to display the & symbol itself? Simply writing “&” in the code will not display it. But first, let us see how to display some special characters.

<HTML>

& is the ampersand symbol

" is the quotation mark symbol
 à is small a, grave
accent symbol
 À is capital a, grave accent symbol

ñ is small n, tilde symbol

Ñ is capital n, tilde symbol
 ü is the umlaut small u
symbol
 Ü is the umlaut

 is the symbol Delta
 ¼ is the quarter symbol

½ is the half symbol

</BODY>

</HTML>

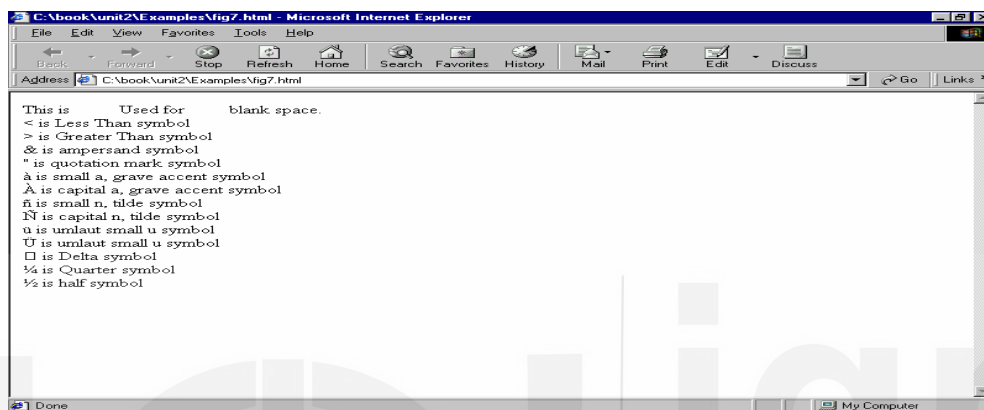


Figure 1.8: Entering Special Characters

The special characters shown in Figure 1.8 are some of the most frequently used characters displayed on web pages. Each of the special characters can be displayed by using its character sequence after the “&”. These can be seen in the following Table 1.1.

Table 1.1: Displaying Special Characters

Special Character	Character Symbol	Description
<	<	Less-than symbol
>	>	Greater-than symbol
&	&	Ampersand
“	"	Quotation Mark
	 	Blank space
à	à	small a, grave accent
À	À	capital A, grave accent
ñ	ñ	small n, tilde
Ñ	Ñ	capital N, tilde
ü	ü	umlaut small u
Ü	Ü	umlaut capital U
<		delta
¼	¼	One Fourth
½	½	Half

The browser will display your text in a steady stream unless you tell it to do so otherwise with line breaks. It will reduce any amount of white space to a single space. If you want more spaces, you must use the space character (). If you hit Return (or Enter) while you are typing, the browser will interpret that as a space unless there is already a space there.

Consider another example, which shows how to display multiple blank lines. Code a space character with a line break for each blank line you want.

```
<HTML>
<HEAD>
  <TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
  Welcome to <BR>
  &nbsp;<BR>
  &nbsp;<BR>
  &nbsp;<BR>
  &nbsp;<BR>
  &nbsp;<BR>INDIA
</BODY>
</HTML>
```

Check Your Progress 1 :

1. Describe yourself on a web page and experiment with colours in BGCOLOR, TEXT, LINK, VLINK, and ALINK. Try out different fonts and sizes and also the other tags you have studied so far, such as the <PRE> tag, as well.

Check Your Progress 2 :

1. Add the following information to the web page that you created above.

“Job: Software Engineer

The requirement for the job is that the person should be B.E./M.E/M.C.A. having an aggregate score of 70% or above. The job is project based, so it would be for ½ year only initially. ¼ of the salary would be deducted towards income tax, PF and other statutory deductions.”

1.5 WORKING WITH IMAGES

Let us now make our web pages more exciting by putting in images.



You specify an image with the (image) tag. Earlier in this section, displaying the images on a page was explained using the BACKGROUND attribute of the <BODY> tag, which displays the image as the background image. Background images make the page heavy and hence the page takes a considerable amount of time to load. But the tag can be used for displaying an image with the desired height and width. Let us look at an example (Figure 1.9).

```
<HTML>
<HEAD>
  <TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
  <IMG SRC="image.gif" WIDTH=130 HEIGHT=101 ALT = "IMAGE IS
  TURNED OFF" ALIGN = "BOTTOM" BORDER = 2> This text is placed
  at the middle of the image.
</BODY>
</HTML>
```

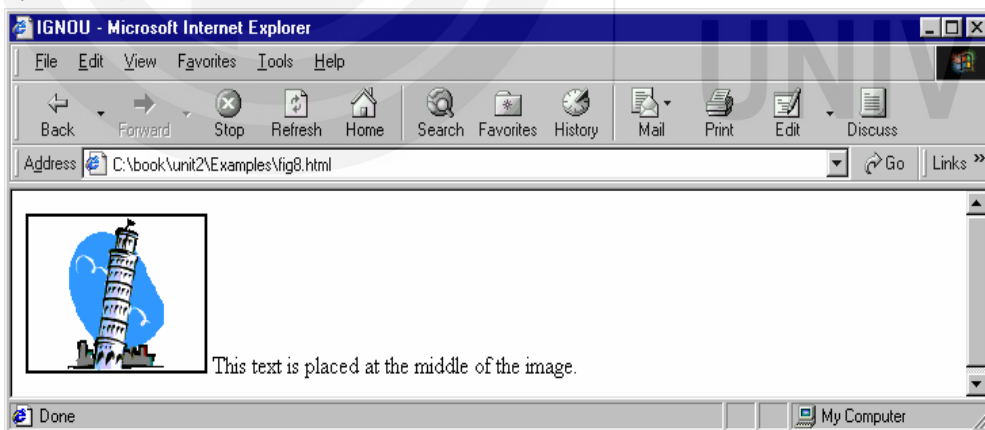


Figure 1.9: Displaying Images on a Web Page

Let us take a look at the syntax of the tag:

```
<IMG SRC = "FILENAME.GIF" WIDTH = "value" HEIGHT = "value"
ALT = "alternate text" BORDER = "value" ALIGN = "value">
```

- a. SRC: This attribute specifies the pathname to the source file that contains the image. The value in the above example, "image.gif", means that the browser will look for the image named image.gif in the same folder (or directory) as the html document itself.
- b. WIDTH: This is used for specifying the desired width of the image.
- c. HEIGHT: This is used for specifying the desired height of the image.
- d. BORDER: An important attribute of the IMG tag is BORDER. This attribute specifies the width of the border of the image. By default it is 0, i.e. there is no border. As shown in Figure 1.9 the image "image.gif" has been given a border 2 pixel wide. The following code gives a wider border to the above image.

```
<BODY BGCOLOR="#FFFFFF">  
<IMG SRC="image.gif" WIDTH=130 HEIGHT=101 BORDER=10>  
</BODY>
```

- e. ALT: Another IMG attribute that is important is ALT. ALT is sort of a substitute for the image that is displayed or used when the user is using a browser that does not display images. Someone may be using a text only browser, he may have image loading turned off for speed or he may be using a voice browser (a browser where the web page is read out). In those cases, that ALT attribute could be very important to your visitor as it could be used (given the proper text) to describe the image that is not on the screen.

Check Your Progress 3:

1. Create your own background with a paint program using the following steps:
 - Create a small graphic with the paint program.
 - Save it as a .jpg or .gif file in the same subdirectory (or folder) that you are keeping the html page that you have been creating.
 - Create a simple HTML file with a background, and put the name of your .jpg or .gif file after the BACKGROUND attribute. (Note: You can easily create the simple html file by copying the html tags above from this web page and pasting them into your new html file. Be sure to substitute the name of your .jpg or .gif file for "image.gif").
 - Save the simple html file that you have just created and open it with your web browser. What do you see? When you are finished, return to this page and continue.

1.6 META TAG

You might be aware of, and perhaps may have used, search engines such as Google to look for web pages on a topic of interest. The META Tag comes in useful if you want your web page to be easily locatable by search engines. When you enter a search string, the search engine shows web pages containing that string, provided the web page has used those in META tag appropriately. The search engine interacts with the META tag of the HTML page in order to find the required string. Information inside a Meta element should be such as to describe the document. Consider the following example (Figure 1.10).

```
<HTML>
<HEAD>
  <TITLE>IGNOU</TITLE>
  <META NAME="author" CONTENT="IGNOU">
  <META NAME="description" CONTENT="This website shows you the
different courses offered by
IGNOU">
  <META NAME="keywords" CONTENT=" Website, different courses
offered, IGNOU,mca,bca">
</HEAD>
<BODY>
<P>
The meta attributes of this document identify the author and courses
offered.
</P>
</BODY>
</HTML>
```

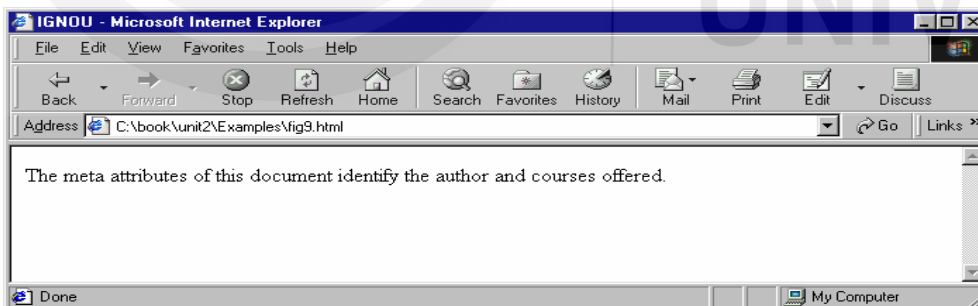


Figure 1.10: Using the META Tag

Meta tags have two attributes:

- a. **NAME:** This attribute is used for identifying the type of META tag you are including.
- b. **CONTENT:** This attribute is used to specify the keywords that the search engine catalogs.

Consider the following code of the example shown in Figure 1.10.

<META NAME= "description" CONTENT="This website shows you the different courses offered by IGNOU">

The CONTENT attribute provides the list of words in the form of a sentence to the search engine. So if someone searches for one of the keywords listed by you in the META tag, then your web site would also appear in the result of the search. It is useful to include META tags that include as many keywords as possible. This makes the web page more likely to show up in a search.

You can also specify keywords by separating them by commas as shown in the following code fragment of Figure 1.10.

<META NAME= "keywords" CONTENT= "Website, different courses offered, IGNOU,mca,bca">

You can use either of the methods of specifying the META tag as convenient.

Consider another example shown in Figure 1.11. This example demonstrates how to redirect a user if your site address has changed.

```
<HTML>
<HEAD>
<TITLE>IGNOU</TITLE>
<META HTTP-EQUIV="Refresh"
CONTENT="5;URL=http://www.ignou.ac.in">
</HEAD>
<BODY>
<P>
Sorry! We have moved! The new URL is: www.ignou.ac.in
</p>
<p>
You will be redirected to the new address in five seconds.
</p>
</BODY>
</HTML>
```

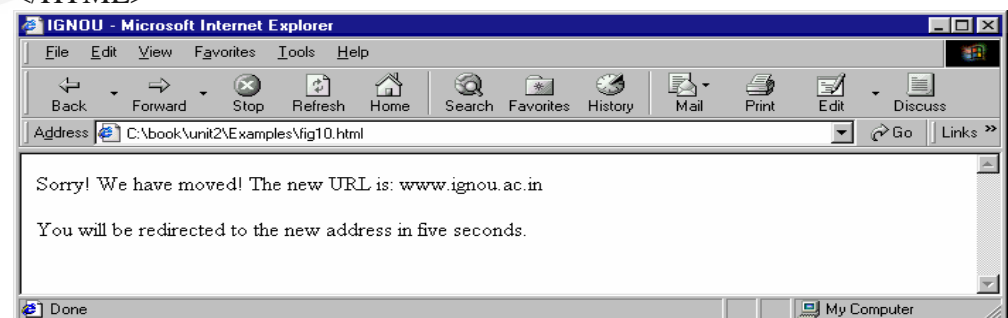


Figure 1.11: Redirecting a User if the Site has Moved

Consider the following code of Figure 1.11

```
<META HTTP-EQUIV= "Refresh"
CONTENT="5;URL=http://www.ignou.ac.in">
```

It indicates to the browser that the page has to be refreshed in 5 seconds with the new URL “http://www.ignou.ac.in”. So when the user sees this page by specifying its original URL, the browser is redirected to the webpage “www.ignou.ac.in” after five seconds. This type of redirection is useful when you want that a user accessing your old website should automatically be redirected to the new website address.

Now let us consider an example that makes use of almost all the tags explained so far.

Case Study: Design a single page web site for a store listing the products and services offered. The store sells computers and related products. The site should contain images explaining the products graphically.

```
<HTML>
<HEAD>
<TITLE> SOLVED CASE STUDY FOR HTML </TITLE>
</HEAD>
<BODY LINK="#0000ff" VLINK="#800080">

<P ALIGN="CENTER">&nbsp;</P>
<B><I><U><FONT SIZE=5><P ALIGN="CENTER">ABC Products</P>
</FONT>
</B></O></I><P>ABC store sells the latest in computers and computer
products. Besides, we also stock stationery.</P>
<P ALIGN="CENTER"><HR></P>
<B><U><P>Product 1.</P>
</U></B><P><IMG SRC="Image1.gif" WIDTH=127 HEIGHT=102></P>
<P>This is a notebook. It has 200 pages. Each page has three columns with a
heading for date, name and address. Its cost is Rs. 100 only.</P>
<P ALIGN="CENTER"><HR></P>
<B><U><P>Product 1.</P>
</U></B><P><IMG SRC="Image1.gif" WIDTH=127 HEIGHT=102></P>
<P>This is a computer. It has 512 MB RAM with a 1.3 GHz processor and an
80 GB HDD. Its cost is Rs. 30,000 only. It is pre-loaded with Windows 2003.
You can buy Microsoft Office software too from us.</P></BODY>
</HTML>
```

Check Your Progress 4:

1. Design a single page web site for a university containing a description of the courses offered. It should also contain some general information about the university such as its history, the campus, its unique features and so on. The site should be coloured and each section should have a different colour.

1.7 SUMMARY

the tag. The tag is used for inserting images in the document. We have also looked at the very useful <META> tag. This tag is used to redirect the users to other pages, and to provide information about the page.

References: INTERNET & WORLD WIDE WEB BY DEITEL, DEITEL & NIETO
HANDS ON HTML BY GREG ROBERTSON

1.8 SOLUTIONS / ANSWERS

🔍 Check Your Progress 1:

1

1.8 SOLUTIONS / ANSWERS

1.8 SOLUTIONS / ANSWERS

👉 Check Your Progress 1:

1.

[illegible]

<p> Father's Name: Mr. Shyam mehta Date of Birth: 23 Jan, 1976. Address: A2-81b, East Of kailash,New Delhi. Tel: 29090909,9220101010 Email: ram@hotmail.com Sex: Male Marital Status: Single Interests and activities: Troubleshooting hardware and software problems. </PRE> </BODY> </HTML></p>	
--	--

Check Your Progress 2:

```
1. <HTML>
<HEAD>
</HEAD>
<BODY LINK="#ffff00" VLINK="#ffff99" BGCOLOR="#ffffff">
<PRE>
<U><B><FONT FACE="Times New Roman, serif">Job: Software
Engineer
</FONT></B></U>
<FONT FACE="Times New Roman, serif">
```

The requirement for the job is that the person should be B.E./M.E/M.C.A having an aggregate score of 70% or above. The job is project based, so

[illegible]

1994
 &


```
</FONT>
</PRE>
</FONT>
</BODY>
</HTML>
```

Check Your Progress 3:

```
1. <HTML>
<HEAD>
<TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<IMG SRC="abc.jpg" WIDTH=130 HEIGHT=101 ALT = " IMAGE IS
TURNED OFF" ALIGN = "MIDDLE" BORDER = 2> This text is
placed at the middle of the image.
</BODY>
</HTML>
```

Check Your Progress 4:

```
1. <HTML>
<HEAD>
</HEAD>
<BODY LINK="#0000ff" VLINK="#800080"
BGCOLOR="#ff6600">
<P ALIGN="CENTER">&nbsp;</P>
<B><I><U><FONT SIZE=4><P ALIGN="CENTER">IGNOU</P>
</I></U></FONT><FONT SIZE=2><P><I></B>ndira <B>G</B>andhi
<B>N</B>ational <B>O</B>pen University is a very old and reputed
```

University. IGNOU offers various types of courses that are both academic and technical. </P>

<P> </P>

<P>Master in Computer Applications</P>

<P>This course has 6 semesters and the total duration is 3 years. The maximum duration allowed for completing the course is 7 years. The fee per semester is Rs 5000 </P>

<P>Bachelor in Computer Applications</P>

<P>This course has 6 semesters and the total duration is 3 years. The maximum duration allowed for completing the course is 6 years. The fee per semester is Rs 3000</P>

<P>Bachelor in Information Technology</P>

<P>This course has 6 semesters and the total duration is 3 years. The maximum duration allowed for completing the course is 5 years. The fee per semester is Rs 10,000.</P></BODY></HTML>

1.9 EXERCISES FOR PRACTICE IN LAB SESSION

Purpose:

This section aims to help learners apply the basic HTML concepts introduced in Section 1 through hands-on exercises. By performing these lab sessions, students will gain practical experience in creating simple web pages, experimenting with text formatting, colors, fonts, and images, and understanding how HTML tags work together to structure a webpage.

SESSION 1

Exercises

1. Write HTML code to develop a Web page having the background in red and title "My First Page" in any other colour.
2. Create an HTML document giving details of your name, age, telephone number, address, TLC code & enrolment number aligned in proper order.
3. Write an HTML code to design a page containing text, in form of paragraphs giving suitable heading style.
4. Create a page to show different attributes of Font tag.
5. Create a page to show different attributes: italics, bold, underline.
6. Design a page having background colour yellow, giving text colour red and using all the attributes of font tab.

SESSION 2

Exercises

1. Write an HTML code to create a Web page of blue color and display links in red colour.
2. Write an HTML code to create a Web page that contains an image at its center.

3. Create a Web page with appropriate content and insert an image towards the left hand side of the page. When user clicks on the image, it should open another Web page.
4. Create a Web page using *href* attribute of anchor tag & the attribute: *alink*, *vlink* etc.
5. Create a Web page, wherein when the user clicks on the link it should go to the bottom of the page.
6. Write HTML code to create a Web page of pink colour and display a moving message in red colour.

SESSION 3

Exercises

1. Create a Web page, showing an ordered list of the names of five of your friends.
2. Create an HTML document containing a nested list showing the content page of any book
3. Create a web page, showing an unordered list of names of five of your friends.
4. Create a Web page, which should contain a table having two rows and two columns.
5. Fill in some dummy data in the table created by you in question 1 of this session.
6. Create the following table in HTML with Dummy Data

Name of train	Place	Destination	Train No.	Time		Fair
				Arrival	Departure	

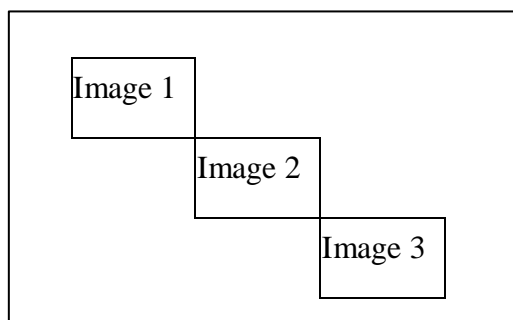
SESSION 4

Exercises

1. Create the following table

Colour (White)		
RED	GREEN	BLACK

2. Design an HTML Page having 3 images placed in the following format.



3. Write HTML code to generate the following output:

Weather	DELHI	MUMBAI
	40	35

4. What are HTML Physical style tags and Logical style tags?
5. Which HTML tag allows you to scroll text on the Web page?
6. What is the comment tag in HTML?
7. Design an HTML Page for the “Block Introduction” of this book: The page should allow scrolling and the code should contain a comment header with your name and enrolment number.

SESSION 5

Exercises

1. What difference does it make if we express the width of a table in percentage or in pixel value? And how do we set the width of a particular column or cell in a table?
2. Write HTML code to generate the following output:

1	2	3	4
5	Image		6
7			8
9	10	11	12

3. Create a Web page that should contain a table having seven rows and four columns, along with the attributes – colspan & rowspan.
4. What are the different versions of HTML?
5. List 5 different HTML Editors.
6. What is the different image formats?

SECTION 2 ADVANCED HTML

Structure	Page Nos.
2.0 Introduction	35
2.1 Objectives	36
2.2 Links	36
2.2.1 Anchor tag	36
2.3 Lists	38
2.3.1 Unordered Lists	38
2.3.2 Ordered Lists	39
2.3.3 Definition Lists	40
2.4 Tables	40
2.4.1 TABLE, TR and TD Tags	41
2.4.2 Cell Spacing and Cell Padding	44
2.4.3 Colspan and Rowspan	46
2.5 Frames	46
2.5.1 Frameset	47
2.5.2 FRAME Tag	47
2.5.3 NOFRAMES Tag	50
2.6 Forms	52
2.6.1 FORM and INPUT Tag	52
2.6.2 Text Box	54
2.6.3 Radio Button	55
2.6.4 Checkbox	55
2.6.5 SELECT Tag and Pull Down Lists	57
2.6.6 Hidden	57
2.6.7 Submit and Reset	58
2.7 Some Special Tags	59
2.7.1 COLGROUP	59
2.7.2 THEAD, TBODY, TFOOT	59
2.7.3 _blank, _self, _parent, _top	60
2.7.4 IFRAME	62
2.7.5 LABEL	62
2.7.6 Attribute for <SELECT>	63
2.7.7 TEXTAREA	64
2.8 Summary	67
2.9 Solutions/ Answers	68
2.10 Advanced HTML Exercises for Practice	78

2.0 INTRODUCTION

In the previous section you have learned the basics of HTML. After learning about how to make static web pages, let us now learn how to develop Interactive Web sites. A good web site should be interactive and easy to use and understand. Of course, very simple web sites that merely need to present some static information may not provide for user input. Interactive web sites are those that are capable of taking input from the user and presenting the output on the basis of the inputs given. To be able to do so, you will need more HTML features than have been covered so far. Features like Links, Lists, Tables, Input controls will allow you to create sophisticated web pages that

respond dynamically to user input. HTML provides all these features using different tags such as A, UL, OL, INPUT, FRAMESET and others, that you will study in this section. Besides the tags themselves you will also learn about their common attributes.

2.1 OBJECTIVES

This section will enable you to create sophisticated, interactive web pages. After going through this section you will be able to learn:

- links using ANCHOR tag;
- ordered, unordered and definition Lists;
- tables;
- frames to divide a web page into different parts; and
- forms for accepting user input.

2.2 LINKS

Hyperlinks, or links are one of the most important characteristics of web pages. A link moves us from the current page to a destination that is specified in the HTML page.

URL Stands for Universal Resource Locator. A URL is just an address that tells the browser precisely where on the Internet the resource is to be found. The process of parsing the URL and actually connecting to the resource can be somewhat complex and does not concern us here.

2.2.1 Anchor Tag

The Anchor tag is used to create links between different objects like HTML pages, files, web sites etc. It is introduced by the characters <A> and terminated by . HREF is the most common attribute of the ANCHOR tag. It defines the destination of the link.

```
<HTML>
<HEAD>
  <TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
  Go to <A HREF="http://www.ignou.ac.in/">IGNOU!</A>
</BODY>
</HTML>
```

As shown above, the text “IGNOU” present between the <A> and tags becomes the hyperlink. On clicking anywhere on this hyperlink, the browser would attempt to connect to the given URL and the website <http://www.ignou.ac.in> would open, if possible. An email link can be specified in the same way. We just have to specify the email address instead of a page

address as the value of HREF as shown in the following code. On clicking on the link, the default mail program on the user's computer opens up with a "To:" address as specified in the hyperlink. You can then type your message and send e-mail to that address.

```
<BODY BGCOLOR="#FFFFFF">  
Send me <A HREF="mailto:forrest@bubbagump.com">mail</A>  
</BODY>
```

It is also possible to make an image a link if desired. This is done using the tag. Consider the following example.

```
<HTML>  
<HEAD>  
  <TITLE>IGNOU</TITLE>  
</HEAD>  
<BODY BGCOLOR="#FFFFFF">  
  Go to <A HREF="http://ignou.ac.in/"><IMG SRC="image.gif"  
    WIDTH=130 HEIGHT=101></A>  
</BODY>  
</HTML>
```

So in the example shown above, the image becomes the link. When the user clicks on the image, the web site <http://www.ignou.ac.in> opens up, if possible.

☛ Check Your Progress 1:

1. Create a web page that provides links to five different web pages or to entirely different web sites.

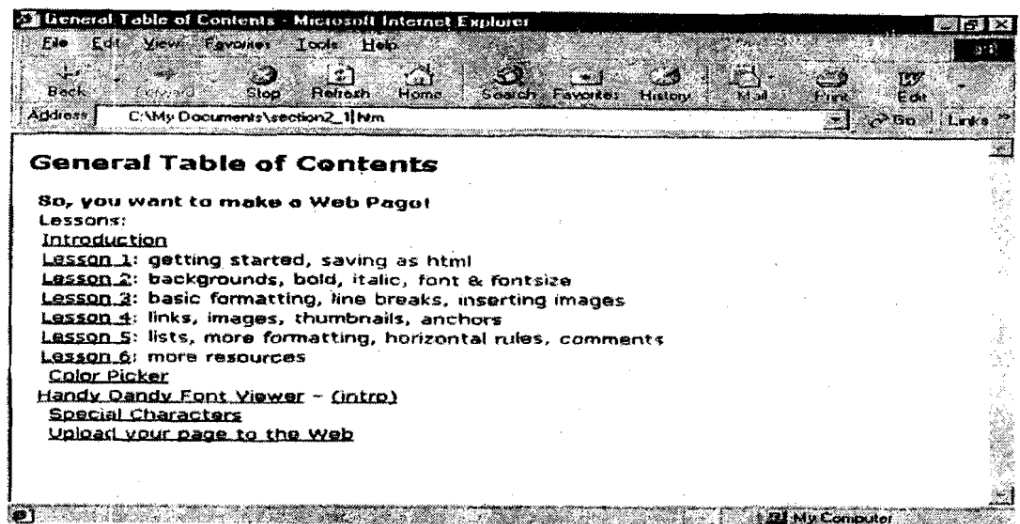
.....

.....

.....

.....

2. Write HTML code for the following page (the underlined text is linked to another file called link_text)



2.3 LISTS

Lists are used when the data are to be mentioned in the form of points like: causes of a particular issue, list of items etc. Lists break up the monotony of a long paragraph and direct the reader's attention to the essential parts.

Lists are segregated into three types, namely Ordered lists, Unordered lists and Definition lists.

2.3.1 Unordered Lists

First, we will build an unordered list. Sometimes, these lists are also called bulleted lists. These lists are characterized by list items that do not have numbers. They are used when the points in the list have no particular order. They are delimited by the and tags. Each point in the list is delimited by the and tags.

```
<HTML>
<HEAD>
<TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
What I want for Id
<UL>
<LI>a big red truck</LI>
<LI>an aeroplane that flies</LI>
<LI>a nice soft toy</LI>
<LI>a drum set</LI>
```

```

<LI>a Walkman</LI>
<LI> extra pocket money</LI>
</UL>
</BODY>
</HTML>

```

The syntax of the tag is:

```

<UL TYPE=""> where TYPE= "DISC", "SQUARE" or "CIRCLE".
<LI> </LI>
</UL>

```

The bullet appearance can be changed to be round (a dark circle), a disc or a circle depending on the value of the TYPE attribute. As shown in the Figure 2.3, the list of items are included within the and tags. Each list item must be preceded with a tag.

2.3.2 Ordered Lists

Lists having numbered items are known as ordered lists. They are used when the items in the list have a natural order. They can also be used when the number of items in the list needs to be known at a glance. To make an ordered list, simply change the tag to .

```

<HTML>
<HEAD>
<TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
  What I want for my birthday
  <OL>
    <LI>a big red truck</LI>
    <LI>a toy train</LI>
    <LI>a Barbie doll</LI>
    <LI>a drum set</LI>
    <LI>binoculars</LI>
    <LI>a month's extra pocket money</LI>
  </OL>
</BODY>
</HTML>

```

The syntax of the tag is:

```

<OL TYPE="" START=""> where TYPE= "1", "a", "A", "i", "I" and START
is the first item number.
<LI> </LI>
</OL>

```

The type of the numbering format desired should be specified as the value of the TYPE attribute. The possible types are shown above in the syntax. The numbering starts from the START attribute's value.

2.3.3 Definition Lists

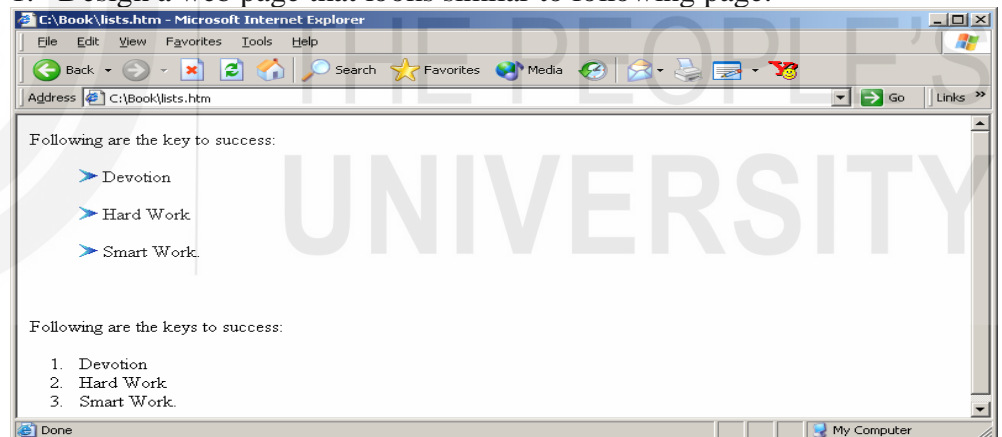
Another type of list is a definition list. Definition lists have a heading and the text appears below that.

```
<HTML>
<HEAD>
  <TITLE>IGNOU</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
  <DL>
    <DT>10th Amendment </DT>
    <DD>The powers not delegated to the United States by the Constitution, nor
    prohibited by it to the States, are reserved to the States respectively, or to the
    people.
  </DD>
  </DL>
</BODY>
</HTML>
```

A definition list is introduced by the <DL> tag and terminated by </DL>. The Definition heading should be specified between the <DT> and </DT> tags. The Definition should be specified between <DD> and </DD> tags.

☛ Check Your Progress 2:

1. Design a web page that looks similar to following page:



.....

.....

.....

.....

2.4 TABLES

In this section you will see how to put tables in your web documents. It is not that a table is simply a combination of rows and columns. If you have ever

seen any table in an attractive web page you might be interested to learn how they make good use of the **<TABLE>** and related tags.

2.4.1 Table, TR and TD Tags

Three tags form the essential ingredients for creating a table.

TABLE: This is the main tag. It tells the browser that a table follows. It has attributes like size and border width.

TR: A TableRow defines a horizontal row that consists of TableCell cells.

TD: This tag specifies an individual block or cell in a table row.

Thus a table is made up of rows, which in turn are made up of cells.

<--This-->	---is---	---a---	---Table---	---Row-->
			Cell	

You are now ready to create some tables! We must stress that if you want to learn how to make quality HTML documents, then you would be spending your time well if you teach yourself the tags. In our opinion the best HTML editors to use for beginners are text-based editors. These editors will force you to code HTML yourself. They don't attempt to do it for you. Once you are an expert, you can use other tools to improve your productivity so that you do not have to handcode your pages.

Now let us start creating tables. The following example shows some tables with different attributes.

```
<HTML>
<HEAD>
<TITLE> IGNOU</TITLE>
</HEAD>
<BODY>
<!-- - Table with border = 5 - - >
First: Table with border = 5
<TABLE BORDER=5>
<TR>
<TD>Ajay</TD>
</TR>
</TABLE>
<!-- - Table with width = 50%- - >
Second: Table with width = 50%
<TABLE BORDER=3 WIDTH=50%>
<TR>
<TD>Ajay</TD>
</TR>
</TABLE>
<!-- - Table with width = 50 - - >
Third: Table with width = 50
```

```

<TABLE BORDER=1 WIDTH=50>
<TR>
<TD>Ajay</TD>
</TR>
</TABLE>
<!-- Table with height = 75 and align = center and valign=top -->
Fourth: Table with height = 75 and align = center and valign=top -
<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
<TR>
<TD ALIGN=CENTER>Ajay</TD>
</TR>
</TABLE>
<!-- Table with an image -->
Fifth: Table with an image
<TABLE BORDER=3>
<TR>
<TD ALIGN=LEFT VALIGN=MIDDLE><IMG SRC="image1.gif"
WIDTH=32
HEIGHT=32></TD>
</TR>
</TABLE>
<!-- A complete Table with different sized columns -->
Sixth: A complete Table with different sized columns
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>
<TR>
<TD WIDTH=60%>Ajay</TD>
<TD WIDTH=20%>Ramesh</TD>
<TD WIDTH=20%>Vijay</TD>
</TR>
<TR>
<TD>Pankaj</TD>
<TD>Vikas</TD>
<TD>Rohan</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

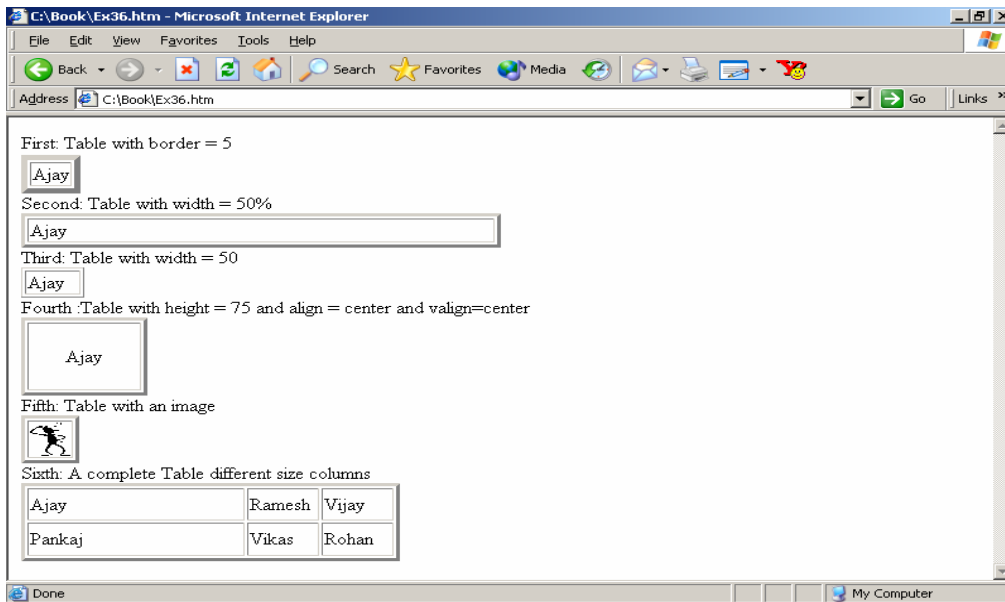


Figure 2.1: Tables with Different Attributes .

In the first table the BORDER attribute is given a value of 5. The default is no border i.e. border = 0.

When no sizes are specified, the table is only as big as it needs to be, as shown in the first and third table in Figure 2.1. Specifying a table size is easy. Let us reduce the table size to 50% of the browser window as has been done in the second example. See the output in the second table in Figure 2.6. As you can see in the example there are two ways to specify the table width. Each style has its uses.

You can also change the height of a table. The fourth table in Figure 2.6 shows the effect of the changed height.

You can control where in the cell the data will appear. For this purpose we use the ALIGN attribute. In the fourth table in Figure 2.6, we have used a center alignment. Similarly, right and left can be used for right and left alignment respectively. The default value is **ALIGN=left**. This is the value that the browser assumes if you have not told it otherwise.

You can also control where data will appear vertically in a cell. For this purpose you specify the VALIGN attribute. In the fourth table, we have used the value center. You can also use top or bottom. Images can also be placed and manipulated in a table data cell. In the folder that contains the document with the HTML code, substitute an **IMG** tag for text. This is shown in the fifth table. You can also include size attributes with all your image tags. We will not go into the details here, but doing so makes it easier for the browser to display the table and avoids any nasty little surprises while resizing the browser window, for instance.

Now, let us look at multiple rows. Suppose three more friends from across the street see what is going on and want to be in your table. We think we will give them their own row. Each row can be assigned a different width for its columns. In the sixth table we have used 60/20/20 as the relative percentage widths of the three columns. The last table shows how to create a table with

multiple rows and columns. The **WIDTH** attributes in the first row carry over to the second row.

2.4.2 Cell Spacing and Cell Padding

Next let us look at a couple of attributes called **CELLPADDING** and **CELLSPACING**. Both are part of the **<TABLE>** tag. **CELLPADDING** is the amount of space between the border of the cell and the contents of the cell. The default value for this attribute is 1. This is so that any text in the cells does not appear to be sticking to the border. However, you can specify a value of 0 if you wish.

The **CELLSPACING** attribute has a somewhat different meaning. It determines the spacing between adjacent cells. Its default value is 2. The following example shows these and some other important attributes of the **<TABLE>** tag.

```
<HTML>
<HEAD>
<TITLE> IGNOU</TITLE>
</HEAD>
<BODY>
First: Table with cellspacing and cellpadding
<TABLE BORDER=3 CELLSPACING=7 CELLPADDING=7>
<TR>
<TD>Ajay</TD>
<TD>Vijay</TD>
<TD>Rohan</TD>
</TR>
<TR>
<TD>Pankaj</TD>
<TD>Vikas</TD>
<TD>Sanjay</TD>
</TR>
</TABLE>
Second: Table with colspan
<TABLE BORDER=1>
<TR>
<TD COLSPAN=2>Ajay</TD>
<TD>Vijay</TD>
</TR>
<TR>
<TD>Vikas</TD>
<TD>Pankaj</TD>
<TD>Rohan</TD>
</TR>
</TABLE>
Third: Table with rowspan
<TABLE BORDER=1>
<TR>
<TD ROWSPAN=2>Ajay</TD>
```



```

<TD>Vijay</TD>
<TD>Rohan</TD>
</TR>
<TR>
<TD>Pankaj</TD>
<TD>Deepak</TD>
</TR>
</TABLE>

```

Fourth :Table with rowspan and colspan

```

<TABLE BORDER=1>
<TR>
<TD ROWSPAN=2>Ajay</TD>
<TD COLSPAN=2>Vijay</TD>
</TR>
<TR>
<TD>Pankaj</TD>
<TD>Rohan</TD>
</TR>
</TABLE>

```

Fifth: Table with strong tag and a link in cell's data

```

<TABLE BORDER=1>
<TR>
<TD COLSPAN=3
ALIGN=CENTER><STRONG><AHREF="http://IGNOU.org/">Ajay</STRO

```

```

NG>
</TD>
</TR>
<TR>
<TD>Vijay</TD>
<TD>Vikas</TD>
<TD>Pankaj</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

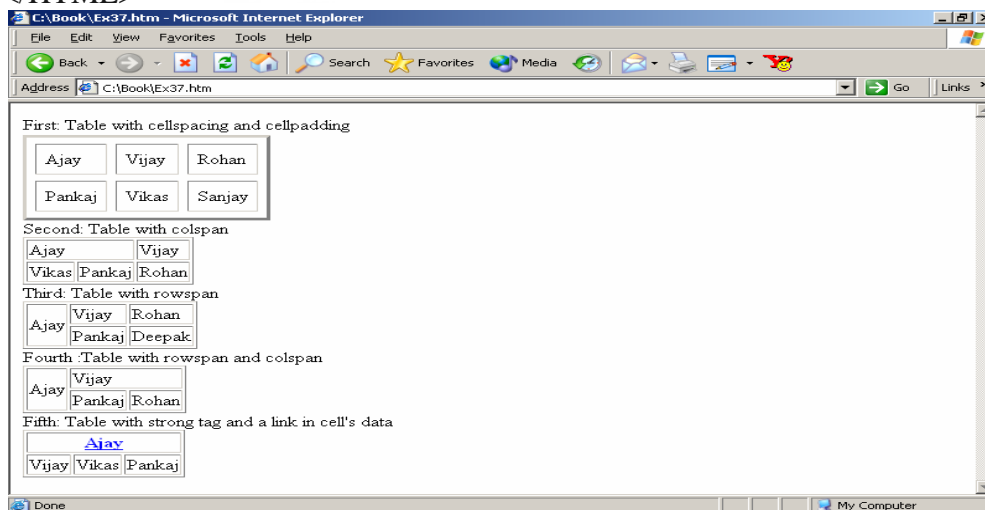


Figure 2.2: Some Important Attributes of the Table Tag Family

The first table in Figure 2.2 illustrates cellspacing and cellpadding.

2.4.3 Colspan and Rowspan

Now let us see how to work with **COLSPAN** (Column Span) and **ROWSPAN** (Row Span).

If we want the cell containing Ajay in Figure 2.7 to be extended to the next cell as well and make that area part of it own, we can use the **COLSPAN** attribute. This amounts to merging the two cells into one. Those of you who are familiar with spreadsheets such as Microsoft Excel will recognize this as similar to its “merge cells” feature. As you may have guessed, **<ROWSPAN>** is just like **<COLSPAN>**. The second table in Figure 2.7 shows COLSPAN, the third table shows ROWSPAN and the fourth table shows an example featuring both COLSPAN and ROWSPAN.

Check Your Progress 3:

1. Design a web page that has 5 equal columns. The table should look the same in all screen resolutions.

2. Make out a brief bio-data of yours and code it as an HTML Page. You can consider using tables to show your academic history.

2.5 FRAMES

Now let us understand how to make frames for web documents. The intelligent use of frames can give your pages a cleaner look and make them easier to navigate.

The basic concept goes like this: Each frame is a regular, complete HTML document. If you wanted to lay out your page into two frames placed side by side, then you would put one complete HTML document in the left frame and another complete HTML document in the right frame. In addition you need to write a third HTML document. This MASTER PAGE contains the **<FRAME>** tags that specify what goes where. Dividing a page into frames is actually quite simple.

There are only two major frame tags that you need to learn: **<FRAMESET>** and **<FRAME>**. The easiest way to explain them is to start making some frames. As an aspiring computer professional, you would be well advised to learn the HTML tags in detail. Do not use HTML editors unless you know HTML, just as you would not let a child use a calculator unless she already knows her arithmetic well. Once you have a thorough grasp of HTML, you can

then save labour by using different tools. The tools might not offer the flexibility that you can get through handcoding HTML. So they should be used judiciously. Of course, this advice is not meant for lay persons who want to make web pages.

We will now need a few HTML documents. We will give each document a name. Create a new folder somewhere and create the first HTML file as One.htm. Now make another html document. Save this in the same folder as Two.htm. Now do the same for Three, Four, Five, and Six. Save them just like the others. You should now have a folder that contains 6 complete standalone HTML documents.

2.5.1 Frameset

Now make a master page in which you write the following code .

```
<HTML>
<HEAD>
<TITLE>My Frame Page -- The Master Page</TITLE>
</HEAD>
<FRAMESET>
</FRAMESET>
</HTML>
```

Now, save it in your folder (with all the other files) as **index.htm**. If you try to open it with your browser now it will be blank. All you have done so far is make a title.

Now let us start defining just how things are going to look. Tell the browser to split the main window into 2 columns, each occupying 50% of the window.

```
<FRAMESET COLS="50%,50%">
</FRAMESET>
```

This will still give a blank output as you have not specified what goes into the windows. We have one more thing to do before our code displays some output.

2.5.2 Frame Tag

As you can guess, this tag is used for placing an HTML file in the frame created. We must now tell the browser what to put in each frame.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50%,50%">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
</FRAMESET>
</HTML>
```

You also need to note here that **<FRAMESET>** is a container tag, and **<FRAME>** is not. A container tag has an opening **<TAG>** and a closing **</TAG>**. So notice that the **<FRAME>** tag has no delimiter to terminate it. Everything is in its attributes.

The **<FRAMESET>** tag does all the dividing of the page into different windows. It also has attributes that specify how to divide them up. Can we divide the page into more than 2 pieces? Yes, just make sure that you specify a page to occupy each section or the browser will get confused. Look at the code and the output in Figure 2.3.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="20%,20%,20%,20%,20%">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
<FRAME SRC="Five.htm">
</FRAMESET>
</HTML>
```



Figure 2.3: A Web Page with Five Frames

It is only a small step to making the frames all of different sizes. Just make sure your arithmetic is correct and that the percentages you specify add up to 100, or the browser will come up with its own interpretation.

If we divide the page into ROWS instead of COLS we get a different layout.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET ROWS="10%,20%,30%,15%,25%">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
<FRAME SRC="Five.htm">
</FRAMESET>
</HTML>
```

Let us now take another example with only 2 frames. We can specify 50 to indicate that number of pixels instead of 50%. We can also use * instead of a number. The * means whatever is left over.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50,*">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
</FRAMESET>
</HTML>
```

When you use frames you have to be very careful to code properly to ensure that all viewers are able to look at reasonably consistent views. Let us suppose that you make a frame 100 pixels wide on the left and 100 pixels wide on the right. If some users are running an 800 × 600 screen they see the middle area as 600 pixels wide. Other users may have a screen set at 640 × 480. What do they see? The middle area for them is only 440 pixels wide. So if you use any absolute dimensions in your **<FRAMESET>** tags you should have at least one * that will produce an elastic frame. That way everything will look at least reasonably good. If you do not do that, your page might need to scroll on one resolution and not on another. As far as possible you might want to avoid absolute values in your frames and work on relative numbers so that things get taken care of automatically by the browser.

We can have more than one leftover frame and specify a size relationship between them. Try it yourself and see the result.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50,*,2*">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAME SRC="Three.htm">
</FRAMESET>
</HTML>
```

The above code means: Make 3 frames. Make the first 50 pixels wide. Divide the rest between frames 2 and 2. But make frame 3 twice as big as frame 2. Put One.html/ in the first frame, Two.html/ in the second and Three.html/ in the third.

It is important to note that everything is done in order. The first **<FRAME>** is displayed according to the first size attribute in the **<FRAMESET>** tag (**50/One**), the second frame with the second (***/Two**) and the third frame with the third attribute set (**2*/Three**).

Frames inside other frames

Here we will discuss how to divide frames into different frames i.e. how to put horizontal frames in a vertical one and vice-versa. Here we are going to divide a frame in half horizontally.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50,*,2*">
<FRAME SRC="One.htm">
<FRAME SRC="Two.htm">
<FRAMESET ROWS="50%,50%">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
</FRAMESET>
</FRAMESET>
```



Figure 2.4: A web page with frame inside other frames

Here the frame three is at the top and Four at the bottom. As shown in Figure 2.4. This has been done just to illustrate how to achieve the effect and is not meant to suggest that you actually divide up your pages like this. Too many frames on a page do not look good. A good rule of thumb is not to have more than 3 frames on your page. If you can, avoid frames altogether.

2.5.3 Noframes Tag

The `<NOFRAMES>` tag can be used for those browsers that are not able to interpret `<FRAME>` tags. Although most, if not all, of your visitors will be able to see frames, there is still a small number of such browsers and there still are many users around who do not have the latest in equipment. To address as wide an audience as possible, you could write a no-frames version of your page.

```
<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="50,*,2*">
<FRAMESET ROWS="50,*,*">
<FRAME SRC="One.htm">
```

```

<FRAME SRC="Five.htm">
<FRAME SRC="Six.htm">
</FRAMESET>
<FRAME SRC="Two.htm">
<FRAMESET ROWS="50%,50%">
<FRAME SRC="Three.htm">
<FRAME SRC="Four.htm">
</FRAMESET>
</FRAMESET>
<BODY><NOFRAMES>   your   browser   does   not   handle   frames!
</NOFRAMES>
</BODY>
</HTML>

```

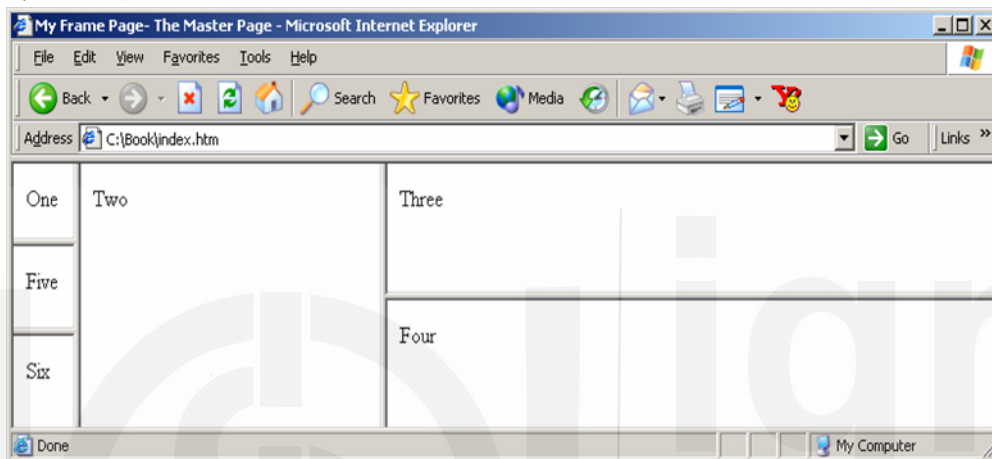


Figure 2.5: Putting in a NOFRAMES Version

Put your non-frames page down between the `<NOFRAMES>` tags. Output is shown in Figure 2.5. If someone is using an old browser, it will skip everything above and come straight down here. Frames-capable browsers will ignore what is between the `<NOFRAMES>` tags.

We can also put images in the frames. Let us see how to put in “world.gif” as an example.

```

<FRAME SRC="world.gif" WIDTH=146 HEIGHT=162>

```

The scrollbars that you see can be specified as **YES**, **NO** or **AUTO**. **YES** means the window gets scrollbars - whether they are needed or not. **NO** means there will be no scrollbars. **AUTO** is the default. If scrollbars are needed, they appear while if they are not needed they stay conveniently out of the way. So let us see how to get rid of our scrollbars.

```

<FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO>

```

You would notice a problem with this code. The image is not in the right frame. The next two attributes deal with margins. The browser automatically gives each frame some empty space around its contents. This is normally useful for aesthetic purposes.

You can control the size of these margins by using **MARGINWIDTH** and **MARGINHEIGHT**. They control the left and right and top and bottom margins respectively. We will set them both to 1. (1 is the minimum)

```

<HTML>
<HEAD>
<TITLE>My Frame Page- The Master Page</TITLE>
</HEAD>
<FRAMESET COLS="146,*">
<FRAMESET ROWS="162,*">
<FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO
MARGINWIDTH=1 MARGINHEIGHT=1>
<FRAME SRC="One.htm">
</FRAMESET>
<FRAME SRC="Two.htm">
</FRAMESET>

```

2.6 FORMS

Now let us get a grip on how to add interactivity to your web documents by way of the **<FORM>** tag. With this tag you can add to your web pages a guestbook, order forms, surveys, get feedback or put in any other form that you wish.

2.6.1 FORM and INPUT Tag

A good way to learn about forms is to use your notepad editor and create a new HTML document. Save it as **form1.htm** in some folder somewhere. You might want to create a separate folder for learning this tag. Start up your browser. Use it to open **form1.html** and run Notepad and the browser side by side. This way you can create your pages and almost instantaneously see the results of your work. Remember to hit the refresh button on your browser.

Next we must tell the browser where to send the data we gather and how to send it. There are two ways you can do this.

- 1) You can send the data to a cgi script or use some other mechanism for processing, or
- 2) You can have the data emailed to you. The first option is outside the scope of discussion of this section.

The second, or mailto, form should have the following attributes in the **<FORM>** tag.

Note: Microsoft's Internet Explorer 2.0 does not support mailto forms. When you try to submit the information, the new mail message window pops up. It does support forms sent to a CGI script.

```

<HTML>
<HEAD>
<TITLE> Welcome to IGNOU </TITLE>
</HEAD>
<BODY>
<FORM METHOD=POST ACTION=mailto:xxx@xxx.xxx
ENCTYPE="application/x-www-form-urlencoded">
</FORM>
</BODY>

```


</HTML>

The line containing the *mailto* keyword is very important. The only thing you have to do is specify your email address after *mailto*: The rest must be written exactly as shown. The words FORM, METHOD, POST and ACTION do not have to be capitalized but there must be a space between every two attributes, between FORM and METHOD, between POST and ACTION, and between the e-mail address and ENCTYPE.

Some mail programs are capable of converting the data without needing a separate program. You may want to try this method first. Just remove the instruction

**ENCTYPE="application/x-www-form-urlencoded" and in its place use
ENCTYPE="text/plain".**

The following example shows a general form that includes some of the commonly used controls.

```
<HTML>
<HEAD>
<TITLE> Welcome to IGNOU </TITLE>
</HEAD>
<BODY>
<FORM METHOD=POST>
```

Text box with name, value and size attributes

```
<INPUT TYPE=TEXT NAME="ADDRESS" VALUE="44 XYZ" SIZE=10>
```


Text box with password and maxlength

```
<INPUT TYPE=PASSWORD MAXLENGTH=10>
```


Radio button with default checked
 Who is your best friend?


```
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="Ajay"
CHECKED>
```

Ajay


```
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="Rohan">
Rohan <BR>
```

```
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="Tarun">
Tarun<P>
```


Checkbox


```
<INPUT TYPE=CHECKBOX NAME="AJAY" VALUE="YES"> Ajay
<BR>
```

```
<INPUT TYPE=CHECKBOX NAME="Rohan" VALUE="YES"> Rohan
<BR>
```

```
<INPUT TYPE=CHECKBOX NAME="BU" VALUE="YES"> Bunt<P>
```


Drop down list


```
<SELECT NAME="BEST FRIEND" SIZE=4>
```

```
<OPTION VALUE="Ajay">Ajay
```

```
<OPTION VALUE="Rohan">Rohan
```

```
<OPTION VALUE="Tarun">Tarun
```

```

<OPTION VALUE="Gagan">Gagan
<OPTION VALUE="Harish">Harish
<OPTION VALUE="Manjit">Manjit
</SELECT>
</FORM>
</HTML>

```

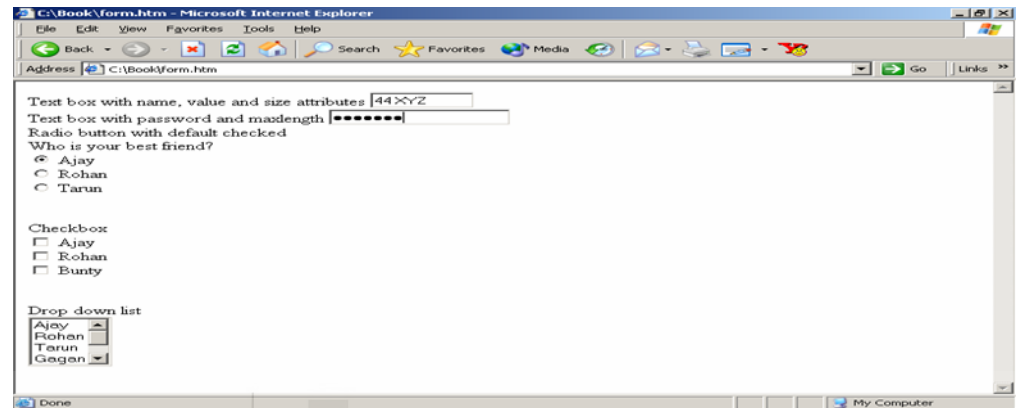


Figure 2.6: A Web Page with a Form

2.6.2 Text Box

The `<INPUT>` tag is used to indicate where user input is expected. It has different attributes, of which the `TYPE` attribute is used to specify the kind of input that is to be provided. The most common value of this attribute of the `<INPUT>` tag is `TEXT`. As shown in Figure 2.6, every `INPUT` needs a `NAME`. When the user types in his address (for example 1234 ABC), it will become the input's value and be paired with `NAME` so the end result after running it through the Mailto Formatter will be `ADDRESS=1234 ABC`.

We can, if we want, type in a `VALUE`.

```
<INPUT TYPE=TEXT NAME="ADDRESS" VALUE="44 XYZ">
```

This will automatically pair the value 44 XYZ with the name ADDRESS, unless the user changes it. Take care to use quotes as specified in the example.

We can specify the size of the text input box.

```
<INPUT TYPE=TEXT NAME="ADDRESS" VALUE="44 XYZ" SIZE=10>
```

The default value is 20. You already know that the default value is the value that the browser assumes if you have not told it otherwise.

Go ahead and remove `VALUE="44 XYZ"`.

If we want, we can specify how many characters a user can input.

Experiment with this and try to input more than 10 characters! The `MAXLENGTH` attribute is used to restrict the number of characters to be entered in the textbox.

```
<INPUT TYPE=TEXT NAME="ADDRESS" SIZE=20 MAXLENGTH=10>
```

Very similar to the TYPE=TEXT is the TYPE=PASSWORD. It is exactly the same, except that for security it displays *** instead of the actual input. The text entered as password would not be echoed on the page. So you can use this whenever you want to accept a password or some other sensitive information from the user.

<INPUT TYPE=PASSWORD>

Remember that each <INPUT> must have a NAME, **that gives the name of the field.**

<INPUT TYPE=PASSWORD NAME="USER PASSWORD">

The SIZE, VALUE, and MAXLENGTH attributes work here also just as they do with TEXT

2.6.3 Radio Button

Radio buttons are used when only one out of the group of options is to be chosen. In the example code we have put a line break after each button.

Each of the Radio Buttons must be assigned a VALUE and you must label each button. The code given in Figure 2.18 illustrates this. You can also modify these labels with other HTML tags if you wish.

This takes care of the basics of your radio buttons. You can tidy things up by adding a statement above the buttons, and if you want choose a default selection (optional). To choose a default selection you need to add "CHECKED" with the appropriate radio button. The user of course can only choose one option. The choice will be returned to you as the name/value pair BEST FRIEND=Ajay (or whatever the user picks).

2.6.4 Checkbox

Checkboxes are used when one or more out of the group of options is to be chosen. Building Check boxes is very similar to radio buttons. Figure 2.7 illustrates the use of Checkbox.

The user can choose any number such as 1, 2, none or all of the options. The choices will be returned to you as the name/value pairs

Ajay=YES

Tarun=YES

(or whatever the user chooses. If nothing, nothing will be returned to you)

The following is code for making a table containing different options for 3 different questions. Try it yourself and see the result.

<CENTER>

<TABLE WIDTH=600 BORDER=1 CELSPACING=1><TR>

<TD WIDTH=199>

Which of these guys are your friends?

<INPUT TYPE=CHECKBOX NAME="Friend?..Ajay" VALUE="YES">

Ajay


```

<INPUT TYPE=CHECKBOX NAME="Friend?..Rohan" VALUE="YES">
Rohan
<BR>
<INPUT TYPE=CHECKBOX NAME="Friend?..Tarun" VALUE="YES">
Tarun<BR>
<INPUT TYPE=CHECKBOX NAME="Friend?..BU" VALUE="YES">
Bunty<P>
</TD>
<TD WIDTH=200>
Which of these guys would you lend money to?<BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...Ajay"
VALUE="YES">
Ajay <BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...Rohan"
VALUE="YES">
Rohan <BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...Tarun"
VALUE="YES">
Tarun<BR>
<INPUT TYPE=CHECKBOX NAME="Lend money?...BU"
VALUE="YES">
Bunty<P>
</TD>
<TD WIDTH=199>
Which of these guys would you trust with your sister?<BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...Ajay"
VALUE="YES"> Ajay
<BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...Rohan"
VALUE="YES">
Rohan <BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...Tarun"
VALUE="YES">
Tarun<BR>
<INPUT TYPE=CHECKBOX NAME="Date sister?...BU" VALUE="YES">
Bunty<P>
</TD>
</TR></TABLE>
</CENTER>

```

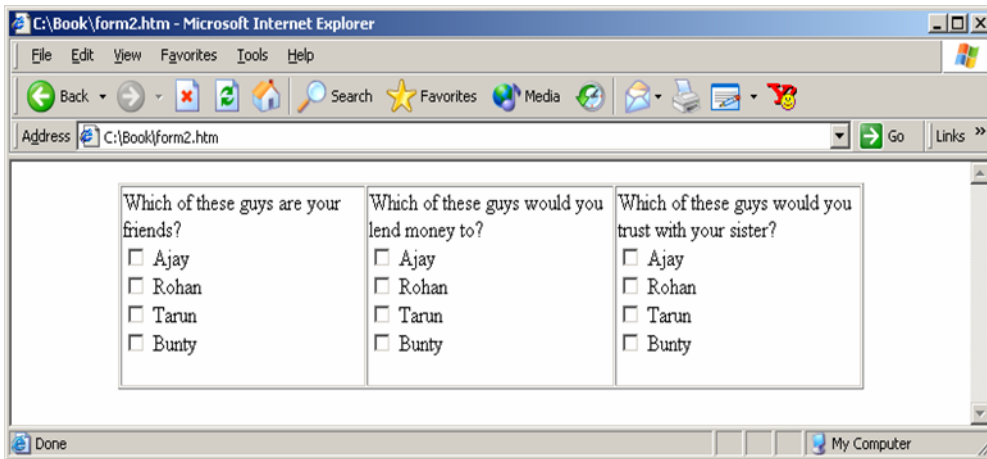


Figure 2.7: A Form on a Web Page with Checkboxes

2.6.5 SELECT Tag and Pull Down Lists

The next type of input is a Pull Down List. With this type you have to use `<SELECT>` instead of `<INPUT>` and it also has a closing tag. This control is used when we have a long list of items to be displayed. The user can also choose any item. All we have to do to turn it into a Scrolling List is to add a `SIZE` attribute to the `<SELECT>` tag. The `SIZE` is simply how many options show in the window.

2.6.6 Hidden

Yet another type of input is HIDDEN input.

`<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Friend Form 1">`

A HIDDEN input is a name/value pair that is returned to you but does not show up anywhere on the web page. The hidden input above is needed for use with the *mailto* Formatter (MTF). It is how MTF recognizes the forms it has to parse.

Suppose you were a company trying to generate leads for a new product. You have a standard form for gathering information that has name, company, phone, products interested in, etc. The only problem is there are 6 slightly different versions of the form in 6 different places. You need to know what is coming from where. What to do?

You could add a HIDDEN input to your forms like this:

```
<HTML>
<BODY>
<FORM METHOD=POST>
<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Version 1">
...for the first version

<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Version 2">
...for the second version
```

```

<INPUT TYPE=HIDDEN NAME="FORMNAME" VALUE="Version 3">
...for the
third version
</FORM>
</HTML>

```

By the way, it doesn't matter what the name/value pair in the hidden input is (or any input for that matter).

<INPUT TYPE=HIDDEN NAME="E" VALUE="Mc^2"> HIDDEN inputs are also useful for cgi scripts. For example, many Internet Service Providers have a script you can have your forms sent to. It then sends the form back to you properly formatted. The hidden input can be used to tell the cgi script who you are, where to send the parsed data, and so on.

2.6.7 Submit and Reset

Submit and Reset are special types of input buttons. Submit is used to send the data to the server and Reset clears/resets the form.

```

<INPUT TYPE=SUBMIT>
<INPUT TYPE=RESET>

```

We can easily change what the buttons say.

```

<INPUT TYPE=SUBMIT VALUE="Send it off!"><BR>
<INPUT TYPE=RESET VALUE="Clear the form!"><P>

```

If necessary, the SUBMIT button can also have a NAME. You would need this if, for whatever reason, you had more than one SUBMIT button.

☛ Check Your Progress 4:

Create a Web page similar to the following:

PERSONAL INFORMATION

First Name:

Middle Name:

Last Name:

Date of Birth: date month year

Marital Status: [Select one]

Gender: [Select one]

Pincode:

Country: India Indonesia Iran

Education: [Select One]

Annual Income: [Select one]

City: [Select One]

If others please pecify:

State:

Occupation: [Select one]

Industry: [Select one]

Accounts not accessed for a consecutive period of 120 days or more may be deactivated without any prior intimation. All data in such mailboxes will be lost. Indiatimes will have no liability for the same.

2.7 SOME SPECIAL TAGS

We now look at some new tags that have been added to HTML in the latest versions.

2.7.1 COLGROUP

<COLGROUP> defines a group of columns in the table and allows you to set the properties of those columns. <COLGROUP> goes immediately after the <TABLE> tag and before any <TR>. <COLGROUP> works very much like <COL>, but you should note that <COLGROUP> requires both an opening and a closing tag.

```
<TABLE BORDER=1 CELLPADDDING=4 RULES=GROUPS
FRAME=BOX>
<COLGROUP></COLGROUP>
<COLGROUP SPAN=3></COLGROUP>
```

2.7.2 THEAD, TBODY, TFOOT

<THEAD>, <TBODY>, and <TFOOT> form groups of rows. <THEAD> indicates that a group of rows are the header rows at the top of the table. <TBODY> indicates that a group of rows are body rows. <TFOOT> indicates that a group of rows are the footer rows at the bottom of the table.

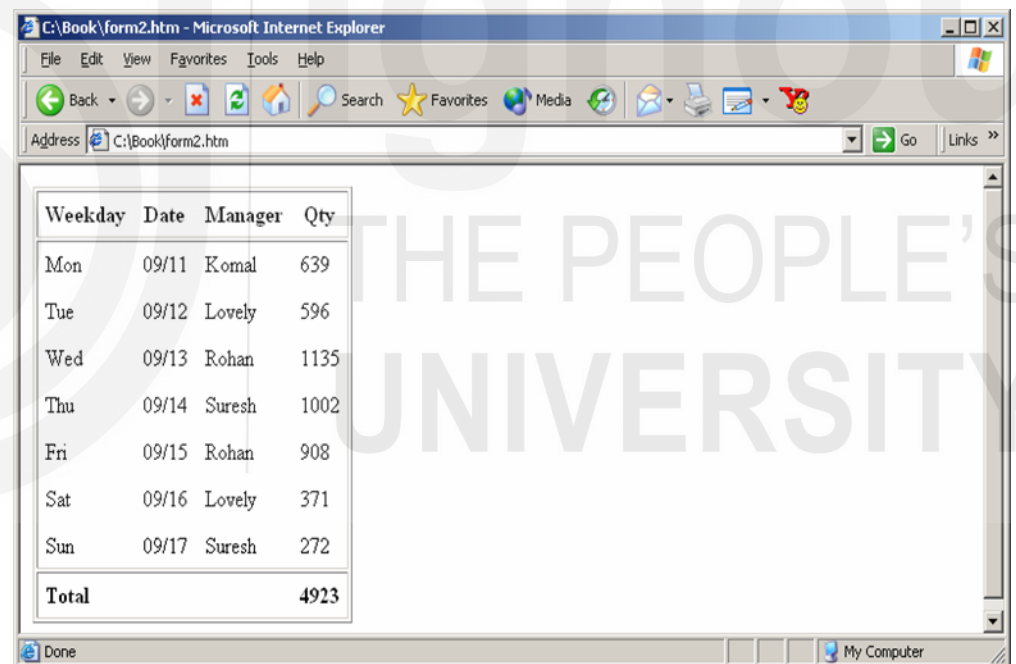
The most popular use for these three tags, which are currently only recognized by MSIE 4 and up, is to put borders between groups of rows instead of between every two rows. For example, suppose you have a table in which you want borders around the top row, the bottom row, and around the entire block of rows in between. You could do that with the following code. Note that in addition to <THEAD>, <TBODY>, and <TFOOT> you also must use <TABLE RULES=GROUPS>. The following example shows the use of these tags and output is shown in Figure 2.8.

```
<HTML>
<BODY>
<TABLE CELLPADDDING=6 RULES=GROUPS FRAME=BOX>
<THEAD>
<TR> <TH>Weekday</TH> <TH>Date</TH> <TH>Manager</TH>
<TH>Qty</TH> </TR>
</THEAD>
<TBODY>
<TR> <TD>Mon</TD> <TD>09/11</TD> <TD>Komal</TD>
<TD>639</TD>
</TR>
<TR> <TD>Tue</TD> <TD>09/12</TD> <TD>Lovely</TD>
<TD>596</TD>
</TR>
```

```

<TR> <TD>Wed</TD> <TD>09/13</TD> <TD>Rohan</TD>
<TD>1135</TD>
</TR>
<TR> <TD>Thu</TD> <TD>09/14</TD> <TD>Suresh</TD>
<TD>1002</TD>
</TR>
<TR> <TD>Fri</TD> <TD>09/15</TD> <TD>Rohan</TD>
<TD>908</TD>
</TR>
<TR> <TD>Sat</TD> <TD>09/16</TD> <TD>Lovely</TD>
<TD>371</TD>
</TR>
<TR> <TD>Sun</TD> <TD>09/17</TD> <TD>Suresh</TD>
<TD>272</TD>
</TR>
</TBODY>
<TFOOT><TR> <TH ALIGN=LEFT COLSPAN=3>Total</TH>
<TH>4923</TH>
</TR></TFOOT>
</TABLE>
</HTML>

```



Weekday	Date	Manager	Qty
Mon	09/11	Komal	639
Tue	09/12	Lovely	596
Wed	09/13	Rohan	1135
Thu	09/14	Suresh	1002
Fri	09/15	Rohan	908
Sat	09/16	Lovely	371
Sun	09/17	Suresh	272
Total			4923

Figure 2.8: Using the THEAD, TBODY and TFOOT Tags

2.7.3 _blank, _self, _parent, _top

These all are attributes of the <A> tag. The following example explains each of these attributes.

```

<HTML>
<BODY>
<UL>
<LI>TARGET = "_blank"</LI>
<A HREF="newwindow.html" TARGET="_blank">a new window</A>
<BR>

```


 TARGET = "_self"

go to next page

TARGET = "_top"

top

</HTML>

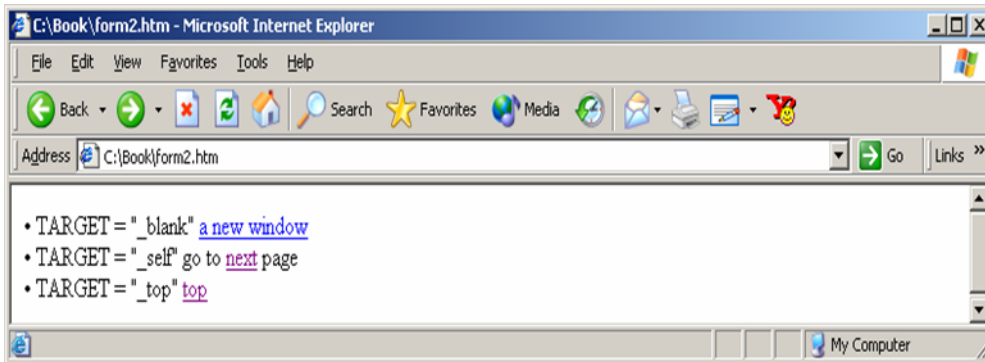


Figure 2.9: Attributes of the <A> Tag

- **TARGET = "_blank"**

"_blank" opens the new document in a new window. Run the code given in Figure 2.9 and check how it works. This value does not require the use of any frames. "_blank" is popular in web pages which are devoted to links to "other resources on the net". By opening a new window for each resource, the user has a sense of a "main" page (the list of resources) and "secondary" pages (each individual resource). Also, the web site providing the link does not risk being supplanted by a link that it has provided.

Note: Some versions of MSIE do not support "_BLANK"

- **TARGET = "_self"**

"_self" puts the new document in the same window and frame as the current document. "_self" works the same as if you had not used TARGET at all. For an example see Figure 2.9.

- **TARGET = "_parent"**

"_parent" is used in a situation where a frameset file is nested inside another frameset file. A link in one of the inner frameset documents which uses "_parent" will load the new document where the inner frameset file had been.

If the current document's frameset file does not have any "parent", then "_parent" works exactly like "_top": the new document is loaded in the full window. Note that "_parent" does not work in a frameset, which is merely nested inside another frameset in the same frameset file.

- **TARGET = "_top"**

"_top" loads the linked document in the topmost frame, that is, the new page fills the entire window.

Refer to Figure 2.9.

2.7.4 IFRAME

<IFRAME> is an HTML 4.0 addition to the frames toolbox. Currently only MSIE supports <IFRAME>. Unlike frames created using <FRAMESET> and <FRAME>,

<IFRAME> creates a frame that sits in the middle of a regular non-framed web page.

<IFRAME> works like , only instead of putting a picture on the page, it puts another web page.

For example, suppose within the same directory as this page there is a file called "hello.html". This code puts hello.html into an inline frame:

```
<IFRAME SRC="hello.html" WIDTH=450 HEIGHT=100>
```

If you can see this, your browser does not understand IFRAME. However, we'll still link link you to the file.

</IFRAME> which gives us this inline frame:

Here's what the code means:

IFRAME

The name of the <IFRAME> tag SRC="hello.html"

The URL of the document to show in the inline frame. WIDTH=450
HEIGHT=100

The dimensions of the inline frame.

If you can see this, your browser doesn't understand IFRAME. However, we'll still link you to the file.

Code between <IFRAME> and </IFRAME> is not displayed by browsers that understand <IFRAME>. Browsers that do not understand <IFRAME> will display this code (because they don't know how to ignore it).

You can do most of the things with <IFRAME> that you can do with regular frames, including setting the frame border, internal margins, and setting information on scroll bars. You can use the attribute so that you can set links to the target frame.

2.7.5 <LABEL>

<LABEL>, an HTML 4.0 element supported by MSIE and Netscape 6, defines a set of text that is associated with a particular form element. For example, the code below indicates that the phrase "send more information" is associated with the "moreinfo" checkbox because the checkbox is within the <LABEL> element:

```
<HTML>
```

```
<BODY>
```

```
<LABEL FOR="moreinfo"> send more information
```

```
<INPUT NAME="moreinfo" TYPE="checkbox" ID="moreinfo">
```

```
</LABEL>
```

```
</HTML>
```

The FOR attribute is required in the above example. The value of FOR should be the same as the value of ID in the form field that the label applies to.

You can also associate a <LABEL> with a field that is not within its contents using the FOR attribute.

2.7.6 Attribute for <SELECT>

TABINDEX = *integer*

TABINDEX is supported by MSIE 4.x and higher and Netscape 6.

Normally, when the user tabs from field to field in a form (in a browser that allows tabbing, not all browsers do) the tab order is the order in which the fields appear in the HTML code.

However, sometimes you want the tab order to flow a little differently. In that case, you can number the fields using TABINDEX. The tabs then flow in order from the one with the lowest TABINDEX to the highest.

The code below illustrates this and output is shown in Figure 2.10.

```
<HTML>
<BODY>
<TABLE BORDER CELLPADDING=3 CELLSPACING=5
BGCOLOR="#FFFFCC">
<TR>
<TD>name: <INPUT NAME="realname" TABINDEX=1></TD>
<TD ROWSPAN=3>comments<BR>

<TEXTAREA COLS=25 ROWS=5
TABINDEX=4></TEXTAREA></TD></TR>
<TR> <TD>email: <INPUT NAME="email" TABINDEX=2></TD></TR>
<TR> <TD>department: <SELECT NAME="dep" TABINDEX=3>
<OPTION VALUE="">...
<OPTION VALUE="mkt">Marketing
<OPTION VALUE="fin">Finance
<OPTION VALUE="dev">Development
<OPTION VALUE="prd">Production</SELECT></TD></TR>
</TABLE>
</HTML>
```

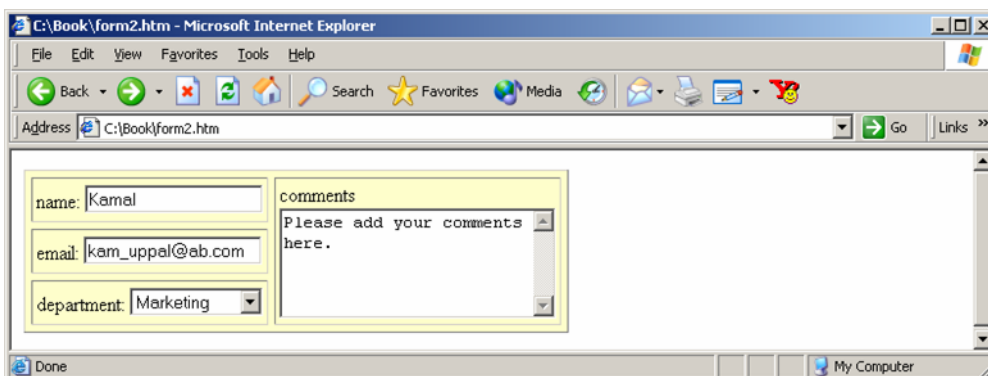


Figure 2.10 : Using the LABEL Tag

TABINDEX can also be used with <A>, <INPUT>, <TEXTAREA>, and <BUTTON>.

2.7.7 TEXTAREA

<TEXTAREA> indicates a form field where the user can enter large amounts of text. In most respects, <TEXTAREA> works like an <INPUT> field. It can have a name and a default value. It has to be noticed that <TEXTAREA> is a container tag, so it has a start tag and an ending tag.

In its simplest form, <TEXTAREA> requires the NAME, COLS and ROWS attributes, and nothing between <TEXTAREA ...> and </TEXTAREA>. Look at the code fragment shown below

```
<FORM ACTION="../cgi-bin/mycgi.pl" METHOD=POST> your  
comments:<BR>  
<TEXTAREA NAME="comments" COLS=40 ROWS=6></TEXTAREA>
```

```
<P><INPUT TYPE=SUBMIT VALUE="submit">  
</FORM>
```

gives us this form:
your comments:

The matter between <TEXTAREA ...> and </TEXTAREA> are used as the default value

```
<FORM ACTION="../cgi-bin/mycgi.pl"> your response:<BR>  
<TEXTAREA NAME="comments" COLS=40 ROWS=6>
```

```
Kamal said  
: I think it's a great idea  
: but it needs more thought  
</TEXTAREA>
```

```
<P><INPUT TYPE=SUBMIT VALUE="submit">  
</FORM>
```

gives us your response: (shown in Figure 2.11)

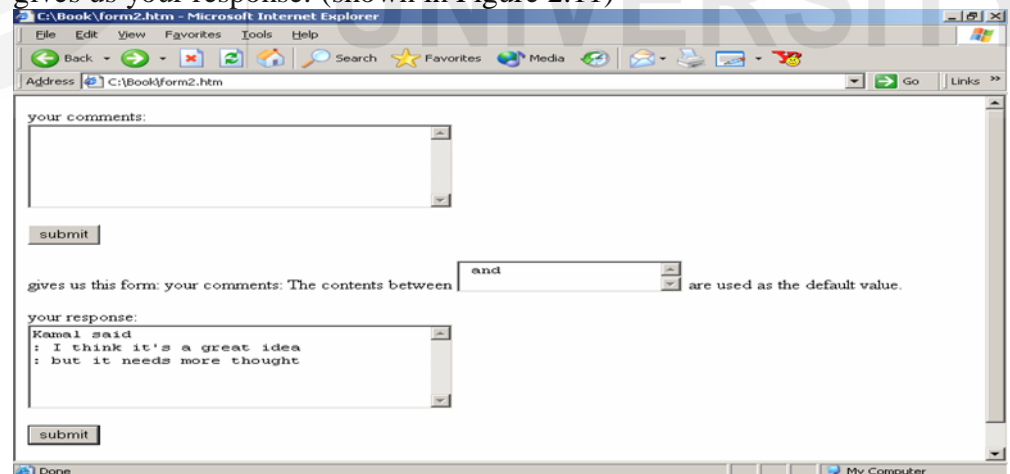


Figure 2.11: Using the TEXTAREA Tag

The contents are interpreted as text only; HTML markup is ignored. Theoretically the user can type unlimited amounts of text into the textarea field. In practice the browser sets a limit that is usually no more than 32 K. If you want users to send in their latest novel, consider using some file upload mechanism.

Case Study: Suppose your boss has asked you to create a Web page from which customers can order computer equipment. You need to collect the customer's name, address, phone number, age, credit card information, and what the customer wants to order.

```
<HTML>
<HEAD>
<TITLE>ComputoRama Order Form</TITLE>
</HEAD>
<BODY>
<FORM ACTION="mailto:mail@abc.com" METHOD=POST>
<TABLE BORDER="2" CELLPADDING="1">
<TR>
<TD ROWSPAN="2">Who Are You?</TD>
<TD><INPUT TYPE="text" NAME="FirstName" SIZE=20></TD>
<TD><INPUT TYPE="text" NAME="MiddleInitial" SIZE=1></TD>
<TD><INPUT TYPE="text" NAME="LastName" SIZE=20></TD>
<TD><INPUT TYPE="text" NAME="Age" SIZE=3></TD>
</TR>
<TR>
<TD><FONT SIZE="-2">First Name</FONT></TD>
<TD><FONT SIZE="-2">MI</FONT></TD>
<TD><FONT SIZE="-2">Last Name</FONT></TD>
<TD><FONT SIZE="-2">Age</FONT></TD>
</TR>
<TR>
<TD ROWSPAN="3">How Do We Contact You?</TD>
<TD COLSPAN="4" VALIGN="TOP">Street Address: <TEXTAREA
name="StreetAddress" rows=2 cols=30></TEXTAREA></TD>
</TR>
<TR>
<TD COLSPAN="2">City: <INPUT TYPE="text" NAME="City"
SIZE=20></TD>
<TD COLSPAN="2">State: <INPUT TYPE="text" NAME="State"
SIZE=2></TD>
</TR>
<TR>
<TD COLSPAN="2">ZIP Code: <INPUT TYPE="text" NAME="ZIPCode"
SIZE=10></TD>
<TD COLSPAN="2">Daytime Phone
(<INPUT TYPE="text" NAME="Phone1" SIZE=3>
<INPUT TYPE="text" NAME="Phone2" SIZE=3>-
<INPUT TYPE="text" NAME="Phone3" SIZE=4></TD>
</TR>
<TR>
<TD>Credit Card
<INPUT TYPE="radio" NAME="CreditCardType" VALUE="Visa"
CHECKED>Visa
<INPUT TYPE="radio" NAME="CreditCardType"
VALUE="MasterCard">M/C</TD>
<TD COLSPAN="2" ALIGN="CENTER">
```

```

<INPUT TYPE="text" NAME="CreditCardNumber1" SIZE=4>
<INPUT TYPE="text" NAME="CreditCardNumber2" SIZE=4>
<INPUT TYPE="text" NAME="CreditCardNumber3" SIZE=4>
<INPUT TYPE="text" NAME="CreditCardNumber4" SIZE=4></TD>
<TD COLSPAN="2">Expiration Date:
<INPUT TYPE="text" NAME="ExpirationMonth" SIZE=2>/
<INPUT TYPE="text" NAME="ExpirationYear" SIZE=2></TD>
</TR>
<TR>
<TD>Merchandise</TD>
<TD COLSPAN=4><SELECT MULTIPLE NAME="Merchandise"

```

```

SIZE=1>

```

```

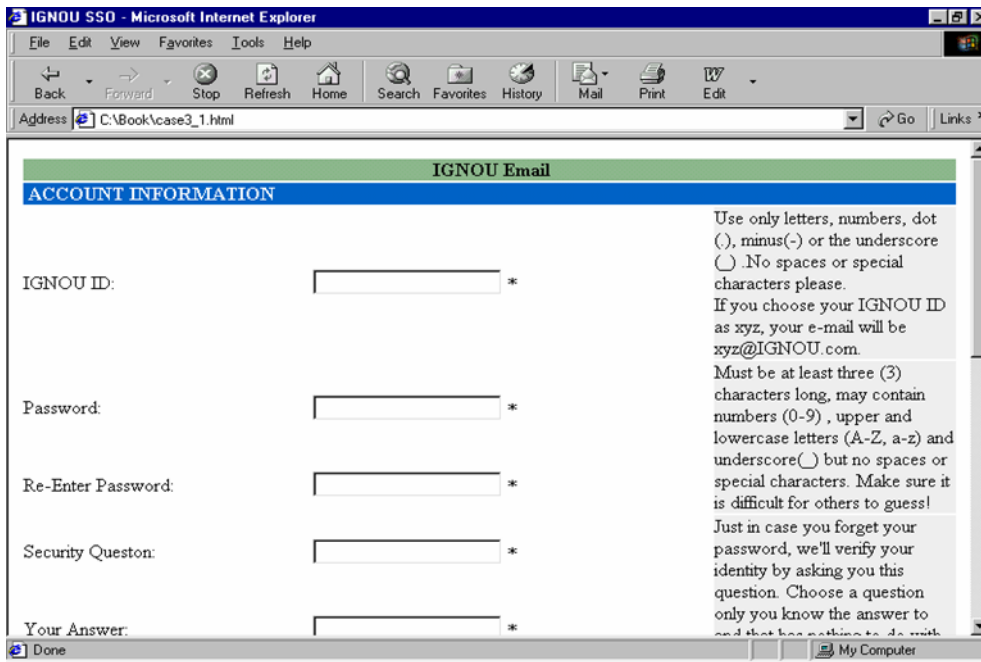
<OPTION SELECTED> HAL-470 <OPTION> Banana90000
<OPTION> High Res Monitor <OPTION> Low Res Monitor
<OPTION> Deluxe Keyboard <OPTION> Regular Keyboard
<OPTION> Laser Printer <OPTION> Inkjet Printer <OPTION> Dot Matrix
Printer
<OPTION> Mouse <OPTION> Trackball
<OPTION> Scanner
</SELECT></TD>
</TR>
<TR>
<TD ALIGN=CENTER COLSPAN="5">
<H1>Thank You For Your Order!</H1>
</TD>
</TR>

</TABLE>
<CENTER>
<INPUT TYPE="submit" VALUE="Ship It!"> <INPUT TYPE="reset"
VALUE="Clear Entries">
</CENTER>
</FORM>
</BODY>
</HTML>

```

Check Your Progress 4:

1. Suppose you are asked to design a registration form for members of a web site. Registration information will include hobbies, interests, assets owned by the member etc. Create the following Web page.(Case study)



2.8 SUMMARY

In this section we have learned some important and advanced topics of HTML. You should now be able to develop interactive Web pages also. We have discussed ways of linking to different locations in a Web site. We have learned about lists that allow us to develop Web pages with elementary data. Lists can be of three types i.e. Ordered Lists, Unordered Lists and Definition Lists. We all know how important a table is in any document -- from someone's bio data to a Chief Executive's report a table is the simplest way to understand the information. If we display the data in paragraph form then it would look unorganized and would be difficult to comprehend. On the other hand, a table presents all the data at a glance. We have also seen how frames are used to divide a page into multiple sections. For example if we want to create a Web page that contains information on three different topics, then we could use frames.

One of the most important points in Web pages is the interaction with the end user. For creating interactive Web pages we use Forms. In a form we can have different controls like Text box, Radio buttons, check box, drop down lists etc. All these controls allow us to interact with the user accept inputs from the user. In the final section we have learned about some of the recently introduced tags that are available in the latest version of HTML.

2.9 SOLUTIONS / ANSWERS

☛ Check Your Progress 1:

1. Following is the code to design a Web page that provides links to five different Web sites.

```
<HTML>
<TITLE> Link to five different Web sites </title>
<BODY>
```

```
Web site 1 <A HREF="http://www.hotmail.com/">Hotmail</A> Web site 2
<A HREF="http://www.google.com/">google</A> Web site 3 <A
HREF="http://www.astalavista.com/">altavista</A>
Web site 4 <A HREF="http://www.education.com/">education</A> Web site
5 <A HREF="http://www.computer.com/">computer</A>
</BODY>
</HTML>
```

2. Following is the code for the given page

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<B><FONT SIZE=5><P>GENERAL TABLE OF CONTENTS</P>
</FONT><P>So, you want to make a Web page</P></B> LESSONS:
<P><A HREF=" ../.. /makapage/introduction.htm">Introduction</A></P>
<P><A HREF=" ../.. /makapage/lesson01.html">Lesson 1</A>: getting started,
saving as html <BR>
<A HREF=" ../.. /makapage/lesson02.html">Lesson 2</A>: backgrounds, bold,
italic, font & fontsize <BR>
<A HREF=" ../.. /makapage/lesson02.html">Lesson 3</A>: basic formatting,
line breaks, inserting images <BR>
<A HREF=" ../.. /makapage/lesson04.html">Lesson 4</A>: links, images,
thumbnails, anchors <BR>
<A HREF=" ../.. /makapage/lesson05.html">Lesson 5</A>: lists, more
formatting, horizontal rules, comments <BR>
<A HREF=" ../.. /makapage/lesson06.html">Lesson 6</A>: more resources
<BR>
<A HREF=" ../.. /makapage/picker/index.html">Color Picker</A> <BR>
<A HREF=" ../.. /makapage/dafonts/master.html">Handy Dandy Font
Viewer</A> -
<A HREF=" ../.. /makapage/dafonts/index.html">(intro)</A> <BR>
<A HREF=" ../.. /makapage/special.html">Special Characters</A> <BR>
<A HREF=" ../.. /makapage/upload.html">Upload your page to the Web</A>
</P>
<P>&nbsp;</P>
<FONT SIZE=2><P> </P></FONT></BODY>
</HTML>
```


Check Your Progress 2:

Code of the given page

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<P> Following are the key to success: </P>
<UL>
<LI><img src = "arrow.gif"> Devotion </LI>
<LI><img src = "arrow .gif"> Hardwork </LI>
<LI><img src ="arrow.gif"> Smart work </LI>
</UL>
<P> Following are the key to success: </P>
<OL>
<LI> Devotion </LI>
<LI> Hardwork </LI>
<LI> Smartwork </LI>
</OL>
</BODY>
</HTML>
```

Check Your Progress 3:

1. Following is the code for a Web page that has 5 equal columns.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<TABLE BORDER CELLSPACING=3 BORDERCOLOR="#000000">
<TR><TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
<TD WIDTH="20%">
<P>Column 1</TD>
</TR>
</TABLE>
<P> </P></BODY>
</HTML>
```

2. Design your Bio-Data in HTML Page .

```
<HTML>
<HEAD>
</HEAD>
<BODY>

<B><U><P>Qualifications</P></B></U>
```

<P ALIGN="JUSTIFY">X & XII from CBSE Board.</P>
 <P ALIGN="JUSTIFY">BCOM (PASS) from Delhi University in 1999.</P>
 <P ALIGN="JUSTIFY">O Level from DOEACC Society India in 2000.</P>
 <P ALIGN="JUSTIFY">MCP (Microsoft Certified Professional) in 2001 in Analyzing Requirements and Defining Solutions Architecture with <U>93.9%</U>.</P>
 <P ALIGN="JUSTIFY">CIC (Certificate In Computing) from Indira Gandhi National Open University with <U>Ist Division</U>. </P>
 <P ALIGN="JUSTIFY">PGDCA (Post Graduate Diploma in Computer Applications) from IGNOU with <U>Ist Division</U>.</P>
 <P ALIGN="JUSTIFY">ADCA (Advance Diploma in Computer Applications) from IGNOU with <U>Ist Division</U>.</P>
 <P ALIGN="JUSTIFY">MCA from Indira Gandhi National Open University with <U>Ist Division</U>. </P>
 <U><P>Languages Known</P>
 <P ALIGN="JUSTIFY"></U>C including data structures, C++ including data structure, MS COBOL 4.5, MS FORTRAN 77, Assembly using MASM, Unix Shell Programming, HTML, Corman Common LISP, Visual Basic 6.0.</P>
 <U><P>Personal Details</P>
 </U><P>Name ABC</P>

<P>Father's Name	XYZ	</P>
 <P>Date of Birth	23rd Jan,1979.	</P>
 <P>Address	A-2, East Of Kailash, New Delhi.	</P>
 <P>Tel.	29090909</P>
 <P>Email	ram@hotmail.com</P>
 <P>Sex	Male	</P>
 <P>Marital Status	Single</P>
 <U><P>Interests and activities</P>
 </U><P>Troubleshooting hardware and software problems.	</P>
 <U><P>Additional Assets</P>
 </U><P>Hard work, devotion and curiosity to learn and adapt to new environments.	</P>
 <P> </P>
 <P>Date:
 (RAM)</P>
 <P>Place:</P></BODY>
 </HTML>

Check Your Progress 4:

```

<head>
<title>IGNOU SSO</title>
<META HTTP-EQUIV="expires" CONTENT="Sat, 15 Mar 2003 23:59:59 GMT">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="Content-Style-Type" content="text/css">
</head>
  
```

```

<body>
<form name= "theForm" action= "/suggest/emailRegistrationServlet"
method=POST onsubmit="return validate()">
<input type="Hidden" name="successURL"
value="http://email.IGNOU.com/success.html">
<input type="Hidden" name="pageTitle" value="IGNOU Email">
<input type="Hidden" name="sourceChannel" value="Email">
<table border=0 cellpadding=0 width="100%">
<tbody>
<tr bgcolor=#8fbc8f>
<td colspan=3 align=center><b class="tdw">&nbsp;IGNOU Email</b></td>
</tr>
<tr bgcolor=#0066cc>
<td colspan=3><b>&nbsp;<font color="#FFFFFF"><span>ACCOUNT
INFORMATION</span></font></b></td>
</tr>
<tr>
<td valign=center width="31%" class="td"> IGNOU ID: </td>
<td valign=center width="43%"> <input name=username> <span>*</span>
</td>
<td bgcolor=#f0f0f0 width="26%" class="td1">Use only letters, numbers,
dot(.), minus(-) or the underscore (_) .No spaces or special characters
please.<br> <span>If you choose your IGNOU ID as xyz,
your e-mail will be xyz@IGNOU.com.</span> </td>
</tr>
<tr>
<td height=42 width="31%" class="td">Password:</td>
<td height=42 width="43%"> <input name=password type=password>
<span>*</span></td>
<td bgcolor=#f0f0f0 rowspan=2 width="26%" class="td1">Must be at least
three (3) characters long, may contain numbers (0-9) , upper and lowercase
letters (A-Z, a-z) and underscore(_) but no spaces or special characters.
Make sure it is difficult for others to guess! </td>
</tr>
<tr>
<td width="31%" class="td">Re-Enter Password:</td>
<td width="43%"> <input name=confirmPassword type=password> <span
>*</span></td>
</tr>
<tr>
<td width="31%" class="td">Security Qestion:</td>
<td width="43%"> <input name=secretQus> <span>*</span></td>
<td bgcolor=#f0f0f0 rowspan=2 width="26%" class="td1">Just in case you
forget your password, we'll verify your identity by asking you this question.
Choose a question only you know the answer to and that has nothing to do
with your password whatsoever.</td>
</tr>
<tr>
<td height=2 width="31%" class="td">Your Answer:</td>
<td height=2 width="43%"> <input name=secretAns> <span>*</span>
</td>
</tr>

```

```
<span >* </span></td>
<td width="26%">&nbsp;</td>
</tr>
<tr>
<td width="31%" class="td">Middle Name:</td>
<td width="43%"> <input name=MiddleName> </td>
<td width="26%">&nbsp;</td>
</tr>
<tr>
<td width="31%" class="td">Last Name:</td>
<td width="43%"> <input name=lastName> <span >*</span> </td>
<td width="26%">&nbsp;</td>
</tr>
<tr>
<td width="31%" class="td">Date of Birth</td>
<td width="43%"> <font face="MS Sans Serif" size="2">
```

```
<span >* </span></td>
<td width="26%">&nbsp;</td>
</tr>
<tr>
<td width="31%" class="td">Middle Name:</td>
<td width="43%"> <input name=MiddleName> </td>
<td width="26%">&nbsp;</td>
</tr>
<tr>
<td width="31%" class="td">Last Name:</td>
<td width="43%"> <input name=lastName> <span >*</span> </td>
<td width="26%">&nbsp;</td>
</tr>
<tr>
<td width="31%" class="td">Date of Birth</td>
<td width="43%"> <font face="MS Sans Serif" size="2">
```

[illegible]

```

<option value="31"><font face="Arial, Helvetica, sans-serif">31</font></option>
</select>
<select name="month">
<option selected><font face="Arial, Helvetica, sans-serif">month</font></option>
<option value="01"><font face="Arial, Helvetica, sans-serif">01</font></option>
<option value="02"><font face="Arial, Helvetica, sans-serif">02</font></option>
<option value="03"><font face="Arial, Helvetica, sans-serif">03</font></option>
<option value="04"><font face="Arial, Helvetica, sans-serif">04</font></option>
<option value="05"><font face="Arial, Helvetica, sans-serif">05</font></option>
<option value="06"><font face="Arial, Helvetica, sans-serif">06</font></option>
<option value="07"><font face="Arial, Helvetica, sans-serif">07</font></option>
<option value="08"><font face="Arial, Helvetica, sans-serif">08</font></option>
<option value="09"><font face="Arial, Helvetica, sans-serif">09</font></option>
<option value="10"><font face="Arial, Helvetica, sans-serif">10</font></option>
<option value="11"><font face="Arial, Helvetica, sans-serif">11</font></option>
<option value="12"><font face="Arial, Helvetica, sans-serif">12</font></option>
</select>
<select
size=1 name=year>
<option selected>year</option>
<option value="1970">1970</option>
<option value="1971">1971</option>
<option value="1972">1972</option>
<option value="1973">1973</option>
<option value="1974">1974</option>
<option value="1975">1975</option>
<option value="1976">1976</option>
<option value="1977">1977</option>
<option value="1978">1978</option>
<option value="1979">1979</option>
<option value="1980">1980</option>
<option value="1981">1981</option>
<option value="1982">1982</option>
<option value="1983">1983</option>
<option value="1984">1984</option>
<option value="1985">1985</option>
<option value="1986">1986</option>
<option value="1987">1987</option>

```

```

<option value="1988">1988</option>
<option value="1989">1989</option>

```

```

<option value="1990">1990</option>
<option value="1991">1991</option>
<option value="1992">1992</option>
<option value="1993">1993</option>
<option value="1994">1994</option>
<option value="1995">1995</option>
<option value="1996">1996</option>
<option value="1997">1997</option>
<option value="1998">1998</option>
<option value="1999">1999</option>
<option value="2000">2000</option>
<option value="2001">2001</option>
<option value="2002">2002</option>
</select>

```

```

</font><span>*</span></td>

```

```

<td rowspan=5 align="center"> <font face=arial size=2
color=red><b>Accounts not accessed for a <br>
consecutive period of 120 days or<br> more may be deactivated without<br>
any prior intimation. All data in such<br> mailboxes will be lost. IGNOU
will<br>
have no liability for the same.</b> </font> </td>

```

```

</tr>

```

```

<tr>

```

```

<td height=25 width="31%" class="td">Marital Status </td>

```

```

<td height=25 width="43%"> <select name=maritalStatus size=1 id="select">

```

```

<option selected value="">[Select one]</option>

```

```

<option value="S">Single</option>

```

```

<option value="M">Married</option>

```

```

</select> <span>*</span></td>

```

```

</tr>

```

```

<tr>

```

```

<td width="31%" class="td">Gender</td>

```

```

<td width="43%"> <font face="Arial, Helvetica, sans-serif" size=2>

```

```

<select size=1 name=gender>

```

```

<option value="" selected>[Select one]</option>

```

```

<option value=M>Male</option>

```

```

<option value=F>Female</option>

```

```

</select>

```

```

</font> <span>*</span></td>

```

```

</tr>

```

```

<tr>

```

```

<td height=35 width="31%" class="td">Pincode </td>

```

```

<td height=35 width="43%"> <span class=td1r>

```

```

<input name=pin size=10 maxLength=15 onKeyPress="if (event.keyCode < 45
|| event.keyCode > 57) event.returnValue = false;">

```

```

</span> <span>*</span></td>

```

```

</tr>

```

```

<tr>

```

```

<td height=24 width="31%" class="td">Country:</td>

```

```

<td height=24 width="43%"> <select name="country" size=3>
<option value="in">Select One
<option value="us">United States of America
<option value="bs">Bahamas
<option value="bh">Bahrain
<option value="bd">Bangladesh
<option value="bb">Barbados
<option value="by">Belarus
<option value="be">Belgium
<option value="bz">Belize
<option value="bj">Benin
<option value="bm">Bermuda
<option value="bt">Bhutan
<option value="cv">Cape Verde
<option value="zm">Zambia
<option value="zw">Zimbabwe </select> <span >* </span></td>
</tr>
<tr>
<td height=27 width="31%" class="td"> Education </td>
<td height=27 width="43%"> <select name=education id="select2">
<option selected value="">[Select One]</option>
<option value=Grad_Post_Prof>Graduate/Post Graduate-
Professional</option>
<option value=Grad_Post_Gen>Graduate/Post Graduate-General</option>
<option value=SomeColl>SomeCollege but not Graduate</option>
<option value=SSC_HSC>SSC/HSC</option>
<option value=5to9>School 5-9 Years</option>
<option value=lessthan4>School upto 4 years</option>
</select> <span >* </span></td>
<td height=27 width="26%">&nbsp;</td>
</tr>
<tr>
<td height=2 width="31%" class="td">Annual Income </td>
<td height=2 width="43%"> <select name=incomeGroup size=1 id="select3"
style="HEIGHT: 22px; WIDTH: 190px">
<option selected value="">[Select one]</option>
<option value=0to1>Rs 1 Lakh & below</option>
<option value=1to3>Rs 1 Lakh -2.99 Lakhs</option>
<option value=3to5>Rs 3 - 4.99 Lakhs</option>
<option value=5to10>Rs 5 - 9.99 Lakhs</option>
<option value=mysterious>a mysterious sum of money</option>
<option value=morethan10>above Rs 10 Lakhs</option>
</select> <span > * </span></td>
<td height=2 width="26%">&nbsp;</td>
</tr>
<TR>
<TD width="31%"><FONT face="Arial, Helvetica, sans-serif"
size=2>City:</FONT></TD>

<TD colSpan=2> <FONT face="Arial, Helvetica, sans-serif" size=2>
<SELECT name=city>
<option value="" selected>[Select One] </option>

```



```

<option value="Ahmedabad">Ahmedabad </option>
<option value="Bangalore">Bangalore </option>
<option value="Patna">Patna </option>
<option value="Noida">NOIDA </option>
<option value="Pune">Pune </option>
<option value="Surat">Surat </option>
<option value="Varanasi">Varanasi </option>
<option value="Vishakapatnam">Vishakapatnam</option>
</SELECT>
</FONT> <FONT color=#ff0000 face=Arial size=1>*</FONT> </TD>
</TR>

```

```

<TR>
<TD height=35><FONT face="Arial, Helvetica, sans-serif"
size=2>State:</FONT></TD>
<TD height=35>
<INPUT maxLength=100 name=state size=35>
<FONT color=#ff0000 face=Arial size=1>*</FONT>
</TD>
<td height=2>&nbsp;</td>
</TR>
<TR>
<TD height=25><FONT face="Arial, Helvetica, sans-serif"
size=2>Occupation</FONT></TD>
<TD height=25>
<SELECT name=occupation size=1 style="HEIGHT: 22px; WIDTH: 190px">
<option value="" selected>[Select one]</option>
<option value="Artist">Artist</option>
<option value="Social worker">Social worker</option>
<option value="Student">Student</option>
<option value="Teacher/Administrator">Teacher / Administrator</option>
<option value="Unemployed/Between_jobs">Unemployed / Between
jobs</option>
<option value="No_Work">What ? me work</option>
<option value="Others">others</option>
</SELECT> <font color=#ff0000 face=Arial size=1>*</font> </TD>
<TD height=25>&nbsp;</TD>
</TR>
<TR>
<TD height=29><FONT face="Arial, Helvetica, sans-serif"
size=2>Industry</FONT></TD>
<TD height=29>
<SELECT name=industry>
<OPTION selected value="">[Select one]</OPTION>
<OPTION value=Sales>Sales</OPTION>
<OPTION value=Marketing>Marketing</OPTION>
<OPTION value=Production>Production</OPTION>
<OPTION value="Self_Employed">Self Employed</OPTION>
<OPTION value=Student>Student</OPTION>
<OPTION value=Others>Others</OPTION>
</SELECT> <font color=#ff0000 face=Arial size=1>*</font> </TD>
<TD height=29>&nbsp;</TD>

```

```
</TR>
</table>
</form>
</body>
</html>
```

2.10 ADVANCED HTML EXERCISES FOR PRACTICE

Purpose:

The purpose of this section is to reinforce the concepts covered in Advanced HTML. These exercises guide learners in developing interactive and well-structured web pages using links, lists, tables, frames, and forms. Through practical implementation, students will enhance their understanding of advanced HTML features and their real-world applications in webpage design.

SESSION 1

Exercises

1. How do you handle the situation when the browser doesn't support frames?
2. What are inline frames?
3. Which tag is used to define frames in HTML?
4. Write an HTML code to develop a Web page having two frames that divide the Web page into two equal rows.
5. Write an HTML code to develop a Web page having two frames that divide the Web page into two equal rows and then divide the second row into two equal columns.
6. Write an HTML code to develop a Web page having frames as described in the above question and then fill each frame with a different background colour.

SESSION 2

Exercises

1. What are the tags used to display information for browsers that do not support frames?
2. Write the various attributes of frameset tag and frame tag.
3. Write a code in HTML to design a page with two frames. The frame should remain proportionate even when page is resized.
4. Write the code to develop a Web page, as shown below, using frames:

5. Write the code to make the background colour of each frame in the above question different.
6. Create a Web page divided into two equal frames.

--	--

SESSION 3

Exercises

1. Create a Web page having two frames, one containing links and the containing content of the links. When link is clicked, appropriate contents should be displayed in Frame 2.
2. Create a home page for a TLC in following format:

TLC Information	
Links	Appropriate Information

3. Create a Web page using all the attributes of the frame and other tags.
4. Design a page with a text box called 'name' and a button with label 'Enter'. When you click on the button another page should open, with the message "Welcome <name>", where name should be equal to the name entered in the first page.
5. What are the values of method attribute of the form tag?
6. Set default value of 'name' text box of question 1 of this session to Victoria. Add another button called Reset on click of this button name 'text box' should be set to 100 default value.

SESSION 4

Exercises

1. Design a form using all input types.

2. Create a simple form accepting:

Name

Enrolment Number

And Submit button

1. Design a Web Page, which is like 'compose' page of e-mail

To

Copy

Message:

Send

1. Which element is used to accept large text inputs from user?

2. Write a code to create a Web page having radio buttons labeled as name of colours. Clicking on each radio button should change the colour of the Web page.

SESSION 5

Exercises

1. What is the purpose of hidden field?

2. Create a form accepting the values: Name

Address

Marks in 10 + 2, Graduation & Post Graduation

3. Which element is used to display a drop down list box?

4. Design a series of three HTML Pages for ABC. COM each called from the previous one. Accept Name on the first page. When the user clicks on the enter button, second page should open. The second page should not display the name but a 'Welcome screen with some information

about ABC.COM. When the user will click on the 'next' button it should display the name accepted in page 1 on page 3.

(Hint: you may use hidden fields)

5. Create a Web page; divide that page into two frames. In one frame create two links that will display different HTML forms in the other frame.



SECTION 3 INTRODUCTION TO JAVASCRIPT

Structure	Page Nos.
3.0 Introduction	82
3.1 Objectives	83
3.2 JavaScript Variables and Data Types	83
3.2.1 Declaring Variables	83
3.2.2 Data Types	84
3.3 Statements and Operators	85
3.4 Control Structures	86
3.4.1 Conditional Statements	87
3.4.2 Loop Statements	88
3.5 Object-Based Programming	90
3.5.1 Functions	90
3.5.2 Executing Deferred Scripts	93
3.5.3 Objects	94
3.6 MessageBox in Javascript	101
3.6.1 Dialog Boxes	101
3.6.2 Alert Boxes	101
3.6.3 Confirm Boxes	102
3.6.4 Prompt Boxes	103
3.7 Javascript with HTML	103
3.7.1 Events	103
3.7.2 Event Handlers	103
3.8 Forms	107
3.8.1 Forms Array	107
3.9 Summary	116
3.10 Solutions/ Answers	116
3.11 JavaScript Lab Exercises	125
3.12 Further Readings	127

3.0 INTRODUCTION

JavaScript is the programming language of the Web. It is used mainly for validating forms. JavaScript and Java can be related to each other. There exist many other differences between the two. The client interprets JavaScript, whereas in Java, one can execute a Java file only after compiling it. JavaScript is based on an object model. In this section you will learn how to write JavaScript code and insert them into your HTML documents, and how to make your pages more dynamic and interactive.

Besides basic programming constructs and concepts, you will also learn about Object- based programming in JavaScript. We will also discuss some commonly used objects, message boxes and forms. One of the most important parts of JavaScript is Event handling that will allow you to write Event-driven code.

3.1 OBJECTIVES

After going through this section you would be able to learn and use the following features of the JavaScript:

- Operators;
- Loop constructs;
- Functions;
- Objects such as Math object, Date object;
- Input and output boxes;
- Event handlers;
- Form object; and
- Form array.

3.2 JAVASCRIPT VARIABLES AND DATATYPES

Let us first see the skeleton of a JavaScript file.

```
<HTML>
<HEAD>
  <TITLE>IGNOU </TITLE>
  <SCRIPT LANGUAGE = "JavaScript">
    </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

JavaScript code should be written between the `<SCRIPT>` and `</SCRIPT>` tags. The value `LANGUAGE = "JavaScript"` indicates to the browser that Javascript code has been used in the HTML document. It is a good programming practice to include the Javascript code within the `<HEAD>` and `</HEAD>` tags.

Now let us start with variables. Variables store and retrieve data, also known as "values". A variable can refer to a value, which changes or is changed. Variables are referred to by name, although the name you give them must conform to certain rules. A JavaScript identifier, or name, must start with a letter or underscore ("`_`"); subsequent characters can also be digits (0-9). Because JavaScript is case sensitive, letters include the characters "A" through "Z" (uppercase) and the characters "a" through "z" (lowercase). Typically, variable names are chosen to be meaningful and related to the value they hold. For example, a good variable name for containing the total price of goods orders would be *total_price*.

3.2.1 Declaring Variables

You can declare a variable with the `var` statement:

```
var strname = some value
```

You can also declare a variable by simply assigning a value to the variable. But if you do not assign a value and simply use the variable then it leads to an error.

Strname = some value

You assign a value to a variable like this:

```
var strname = "Hello"
```

Or like this:

```
strname = "Hello"
```

The variable name is on the left hand side of the expression and the value you want to assign to the variable is on to the right side. Thus the variable "strname" shown above gets the value "Hello" assigned to it.

Life span of variables

When you declare a variable within a function, the variable can be accessed only within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

3.2.2 Data Types

A value, the data assigned to a variable, may consist of any sort of data. However, JavaScript considers data to fall into several possible types. Depending on the type of data, certain operations may or may not be allowed on the values. For example, you cannot arithmetically multiply two string values. Variables can be of these types:

Data Types	Description
Number	<p>3 or 7.987 are the examples of Integer and floating-point numbers.</p> <p>Integers can be positive, 0, or negative; Integers can be expressed in decimal (base 10), hexadecimal (base 16), and octal (base 8). A decimal integer literal consists of a sequence of digits without a leading 0 (zero). A leading 0 (zero) on an integer literal indicates it is in octal; a leading 0x (or 0X) indicates hexadecimal. Hexadecimal integers can include digits (0-9) and the letters a-f and A-F. Octal integers can include only the digits 0-7.</p> <p>A floating-point number can contain either a decimal fraction, an "e" (uppercase or lowercase), that is used to represent "ten to the power of" in scientific notation, or both. The exponent part is an "e" or "E" followed by an integer, which can be signed (preceded by "+" or "-"). A floating-point literal must have at least one digit and either a decimal point or "e" (or "E").</p>
Boolean	<p>True or False. The possible Boolean values are true and false. These are special values, and are not usable as 1 and 0. In a comparison, any expression that evaluates to 0 is taken to be false, and any statement that evaluates to a number other than 0 is taken to be true.</p>
String	<p>"Hello World!" Strings are delineated by single or double quotation marks. (Use single quotes to type strings that contain quotation marks.)</p>
Object	<p>MyObj = new Object();</p>

Null	Not the same as zero – no value at all. A null value is one that has no value and means nothing.
Undefined	A value that is undefined is a value held by a variable after it has been created, but before a value has been assigned to it.

3.3 STATEMENTS AND OPERATORS

Assignment Operators		
Operator	Functionality	Example/Explanation
=	Sets one value equal to another	counter=0 Sets the counter to equal the number 0
+=	Shortcut for adding to the current value.	clicks += 2 Sets the variable named counter to equal the current value plus two.
-=	Shortcut for subtracting from the current value.	clicks -= 2 Sets the variable named counter to equal the current value minus two.
*=	Shortcut for multiplying the current value.	clicks *= 2 Sets the variable named counter to equal the current value multiplied by two.
/=	Shortcut for dividing the current value.	clicks /= 2 Sets the variable named counter to equal the current value divided by two.

Comparison Operators		
Description: The comparison operators compare two items and return a value of "true" if the condition evaluates to true, else they return false.		
Operator	Functionality	Example/Explanation
==	Returns a true value if the items are the same	Counter == 10 Returns the value "true" if the counter's value is currently equal to the number 10
!=	Returns a true value if the items are not the same	Counter != 10 Returns the value "true" if the counter's value is any value except the number 10
>	Returns a true value if the item on the left is greater than the item on the right	counter>10 Returns the value "true" if the counter's value is larger than the number 10
>=	Returns a true value if the item on the left is equal to or greater than the item on the right	counter>=10 Returns the value "true" if the counter's value is equal to or larger than the number 10
<	Returns a true value if the item on the left is less than the item on the right	counter<10 Returns the value "true" if the counter's value is smaller than the number 10
<=	Returns a true value if the item on the left is equal to or less than the item on the right	counter<=10 Returns the value "true" if the counter's value is equal to or less than the number 10
Computational Operators		
Description: The computational operators perform a mathematical function on a value or values, and return a single value.		
Operator	Functionality	Example/Explanation

+	Adds two values together	counter+2 Returns the sum of the counter plus 2
-	Subtracts one value from another	counter-2 Returns the sum of the counter minus 2
*	Multiplies two values	counter*10 Returns the result of the variable times 10
/	Divides the value on the left by the one on the right and returns the result	counter/2 Divides the current value of the counter by 2 and returns the result
++X	Increments the value, and then returns the result	++counter Looks at the current value of the counter, increments it by one, and then returns the result. If the counter has a value of 3, this expression returns the value of 4.
X++	Returns the value, and then increments the value	counter++ Returns the value of the counter, then increments the counter. If the counter has a value of 3, this expression returns the value of 3, then sets the counter value to 4.
--X	Decreases the value, and then returns the result	--counter Looks at the current value of the counter, decreases it by one, and then returns the result. If the counter has a value of 7, this expression returns the value of 6.
X--	Returns the value, and then decreases the value	counter-- Returns the value of the counter, then decreases the counter value. If the counter has a value of 7, this expression returns the value of 7, then sets the counter value to 6.

Logical Operators		
Description: The logical operators evaluate expressions and then return a true or false value based on the result.		
Operator	Functionality	Example/ Explanation
&&	Looks at two expressions and returns a value of "true" if the expressions on the left and right of the operator are both true	If day = 'friday' && date=13 then alert("Are You Superstitious?") Compares the value of the day and the value of the date. If it is true that today is a Friday and if it is also true that the date is the 13th, then an alert box pops up with the message "Are You Superstitious?"
	Looks at two expressions and returns a value of "true" if either one -- but not both -- of the expressions are true.	if day='friday'&&date=13 then alert("Are You Superstitious?") else if day='friday' date=13 then alert("Aren't you glad it isn't Friday the 13th?") Compares the value of the day and the value of the date. If it is true that today is a Friday and if it is also true that the date is the 13th, then an alert box pops up with the message "Are You Superstitious?" If both are not true, the script moves onto the next line of code... Which compares the value of the day and the value of the date. If either one -- but not both -- is true, then an alert box pops up with the message "Aren't you glad it isn't Friday the 13th?"

3.4 CONTROL STRUCTURES

JavaScript supports the usual control structures:

- The conditionals if, if...else, and switch;
- The iterations for, while, do...while, break, and continue;

3.4.1 Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have three conditional statements:

- if statement - use this statement if you want to execute a set of code when a condition is true
- if...else statement - use this statement if you want to select one of two sets of code to execute
- switch statement - use this statement if you want to select one of many sets of code to execute

```
if( myVariable == 2 ) {  
  myVariable = 1;  
} else {  
  myVariable = 0;  
}
```

If the value of myVariable in Figure 3.1 is 2 then the first condition evaluates to true and the value of myVariable is set to 1. If it is anything other than 2 then the else part gets executed.

Now let us see an example of a nested if statement in Figure 3.2.

```
if ( myVariable == 2 ) {  
  myVariable = 1;  
  
} else {  
  
  If (myVariable == 5 ) {  
    myVariable = 3;  
  } else {  
    myVariable = 4;  
  }  
}
```

Switch Statement

If there exist multiple conditions, the switch statement is recommended. This is because only one expression gets evaluated based on which control directly jumps to the respective case.

```
switch(myVar) {  
  case 1:  
    //if myVar is 1 this is executed  
  case 'sample':  
    //if myVar is 'sample' (or 1, see the next paragraph)  
    //this is executed  
  case false:  
    //if myVar is false (or 1 or 'sample', see the next paragraph)
```

```
//this is executed
default:
//if myVar does not satisfy any case, (or if it is
//1 or 'sample' or false, see the next paragraph)
//this is executed
}
```

As shown in Figure 3.3, depending on the value of “myvar”, the statement of the respective case gets executed. If a case is satisfied, the code beyond that case will also be executed unless the break statement is used. In the above example, if myVar is 1, the code for case 'sample', case ‘false’ and ‘default’ will all be executed as well.

3.4.2 Loop Statements

A loop is a set of commands that executes repeatedly until a specified condition is met. JavaScript supports two loop statements: for and while. In addition, you can use the break and continue statements within loop statements. Another statement, for...in, executes statements repeatedly but is used for object manipulation.

- **For Statement**

A for loop repeats until a specified condition evaluates to false. The JavaScript for loop is similar to the Java and C for loops. A for statement looks as follows:

```
for ([initial-expression]; [condition]; [increment-expression]) { Statements
}
```

When a for loop executes, the following sequence of operations occur:

1. The initializing expression initial-expression, if any, is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.
2. The condition expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the loop terminates.
3. The update expression increment-expression executes.
4. The statements get executed, and control returns to step 2. Actually the syntax provides for a single statement; when enclosed in braces ‘{’ and ‘}’, any number of statements are treated as a single statement.

The following function contains a for loop that counts the number of selected options in a scrolling list (a select object that allows multiple selections). The for loop declares the variable i and initializes it to zero. It checks that i is less than the number of options in the select object, performs the succeeding if statement, and increments i by one after each pass through the loop.

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

```

<TITLE>IGNOU </TITLE>
<SCRIPT LANGUAGE = "JavaScript">
function howMany(selectObject) {
var numberSelected=0
for (var i=0; i < selectObject.options.length; i++) {
if (selectObject.options[i].selected==true)
numberSelected++
}
return numberSelected

</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="selectForm">
<P><B>Choose some music types, then click the button below:</B>
<BR><SELECT NAME="musicTypes" MULTIPLE>
<OPTION SELECTED> R&B
<OPTION> Jazz
<OPTION> Blues
<OPTION> New Age
<OPTION> Classical
<OPTION> Opera
</SELECT>
<P><INPUT TYPE="button" VALUE="How many are selected?"
onClick="alert ('Number of options selected: ' +
howMany(document.selectForm.musicTypes))">
</FORM>
</BODY>
</HTML>

```

- **While Statement**

The while statement defines a loop that iterates as long as a condition remains true. In the following example the control waits until the value of a text field becomes "go": while (Document.Form1.Text1.Value != "go") {Statements }

In a while loop the condition is evaluated first before executing the statements.

- **For In Statement**

This is a different type of loop, used to iterate through the properties of an object or the elements of an array. For example consider the following statement that loops through the properties of the Scores object, using the variable x to hold each property in turn:

```
For (x in Scores) {Statements}
```

- **Break Statement**

The break statement is used for terminating the current While or For loop and then transferring program control to the statement just after the terminated

loop. The following function has a break statement that terminates the while loop when *i* becomes equal to 3, and then returns the value $3 * x$.

```
function testBreak(x) {  
  var i = 0  
  while (i < 6) {  
    if (i == 3)  
      i++  
    break  
  }  
  return i*x  
}
```

- **Continue Statement**

A continue statement terminates execution of the block of statements in a while or for loop and continues execution of the loop with the next iteration. In contrast to the break statement, continue does not terminate the execution of the loop entirely.

Instead,

- In a while loop, it jumps back to the condition.
- In a for loop, it jumps back to the increment-expression.

The following example shows a While loop with a continue statement that executes when the value of *i* becomes equal to three. Thus, *n* takes on the values one, three, seven, and twelve.

```
i = 0  
n = 0  
while (i < 5) {  
  i++  
  if (i == 3) continue  
  n += i  
}
```

3.5 OBJECT-BASED PROGRAMMING

JavaScript is a very powerful object-based (or prototype-based) language. JavaScript is not a full-blown OOP (Object-Oriented Programming) language, such as Java, but it is an object-based language. Objects not only help you better understand how JavaScript works, but in large scripts, you can create self-contained JavaScript objects, rather than the procedural code you may be using now. This also allows you to reuse code more often.

3.5.1 Functions

Functions are the central working units of JavaScript. Almost all the scripting code uses one or more functions to get the desired result. If you want your page to provide certain a user-defined functionality, then functions are a convenient way of doing so. Therefore it is important that you understand what a function is and how it works.

First let us understand the basic syntax of a function; then we look at how to call it. After that you must know how to pass arguments and why you need to do this.

Finally, you have to know how to return a value from a function. The following code shows the implementation of a function.

```
function example(a,b)
{
    number += a;
    alert('You have chosen: ' + b);
}
```

The function made above can be called using the following syntax.
Example(1,'house')

In fact, when you define the function Example, you create a new JavaScript command that you can call from anywhere on the page. Whenever you call it, the JavaScript code inside the curly brackets {} is executed.

Calling the Function

You can call the function from any other place in your JavaScript code. After the function is executed, the control goes back to the other script that called it.

```
alert('Example 1: the House');
example(1,'house');
(write more code)
```

So this script first generates an alert box, then calls the function and after the function is finished it continues to execute the rest of the instructions in the calling code.

Arguments

You can pass arguments to a function. These are variables, either numbers or strings, which are used inside the function. Of course the output of the function depends on the arguments you give it.

In the following example we pass two arguments, the number 1 and the string 'house':
example(1,'house');

When these arguments arrive at the function, they are stored in two variables, a and b. You have to declare these in the function header, as you can see below.

```
function example(a,b)
```

```
<HTML>
<HEAD>
<TITLE>IGNOU </TITLE>
<SCRIPT Language = "JavaScript"> function example(a, b)
{
    var number; number += a ;
    alert('You have chosen: ' + b);
}
</SCRIPT>
```

```

</HEAD>
<BODY>
<FORM NAME="selectForm">
<P><B>Click the button below:</B>
<BR>
<P><INPUT TYPE="button" VALUE="Click"
onClick="example(1,'house')">
</FORM>
</BODY>
</HTML>

```



Figure 3.1: Using Functions

It adds 1 to number and displays “You have chosen: house”. Of course, if you call the function like example (5,'palace'), then it adds 5 to number and displays “You have chosen: palace”. The output of the function depends on the arguments you give it. Figure 3.1 illustrates the use of function.

Returning a value

One more thing a function can do is to return a value. Suppose we have the following function:

```

<HTML>
<HEAD>
<TITLE>IGNOU </TITLE>
<SCRIPT Language = "JavaScript"> function calculate(a,b,c)
{
d = (a+b) * c; return d;
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language = "JavaScript"> var x = calculate(4,5,9);
var y = calculate((x/3),3,5);
alert('calculate(4,5,9) = ' + x + ' and ' + ' calculate((x/3),3,5) = ' + y);
</SCRIPT>
</BODY>
</HTML>

```

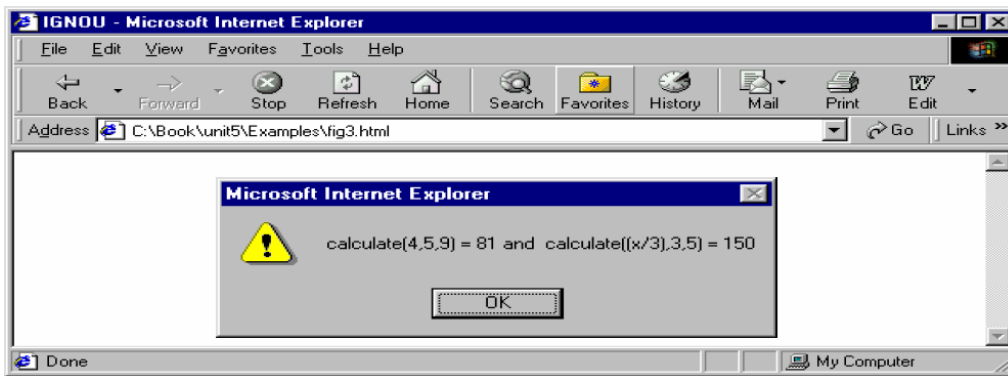



Figure 3.2: Using a Function that Returns a Value

The function shown in Figure 3.2 calculates a number from the numbers you pass to it. When it is done it returns the result of the calculation. The function passes the result back to the function that called it. When the function executes the return statement, control goes back to the calling program without executing any more code in the function, if there is any.

The calling of the function is done using the following two statements in the figure:

```
var x = calculate(4,5,9);
var y = calculate((x/3),3,5);
```

It means that you have declared a variable *x* and are telling JavaScript to execute `calculate()` with the arguments 4, 5 and 9 and to put the returned value (81) in *x*. Then you declare a variable *y* and execute `calculate()` again. The first argument is *x*/3, which means $81/3 = 27$, so *y* becomes 150. Of course you can also return strings or even Boolean values (true or false). When using JavaScript in forms, you can write a function that returns either true or false and thus tells the browser whether to submit a form or not.

3.5.2 Executing Deferred Scripts

Deferred scripts do not do anything immediately. In order to use deferred commands, you must call them from outside the deferred script. There are three ways to call deferred scripts:

- From immediate scripts, using the function mechanism
- By user-initiated events, using event handlers
- By clicking on links or image-map zones that are associated with the script

Calling Deferred Code from a Script

A function is a deferred script because it does not do anything until an event, a function, a JavaScript link, or an immediate script calls it. You have probably noticed that you can call a function from within a script. Sometimes you are interested in calling a function from the same script, and in other cases you might want to call it from another script. Both of these are possible.

Calling a function from the same script is very simple. You just need to specify the name of the function, as demonstrated below.

```
<HTML>
<HEAD>
<TITLE>Calling deferred code from its own script</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
makeLine(30)
function makeLine(lineWidth) { document.write("<HR SIZE=" + lineWidth +
">")
}
makeLine(10)
// -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

3.5.3 Objects

A JavaScript object is an instance of datatype. Object is given a unique name and the collection of properties of the corresponding object may be accessed using the dot syntax. As a quick introduction to the concept of JavaScript objects, here is the code that creates an instance of an object called myObj:

```
var myObj = new Object()
myObj.business = "Voice and Data Networking"; myObj.CEO = "IGNOU";
myObj.ticker = "CSCO";
```

After having run that code (in a scope situation that allowed myObj to be accessed as required), you could run this document.write("My Object ticker symbol is " + myObj.ticker + "."); and get a complete sentence in your HTML document.

- **Document Object**

The document object is a property of the window object. This object is the container for all HTML HEAD and BODY objects associated within the HTML tags of an HTML document

Document Object Properties

Property	Description
AlinkColor	The color of active links
BgColor	The background color of the web page. It is set in the <BODY> tag. The following code sets the background color to white. document.bgColor = "#FFFFFF"
Cookie	Used to identify the value of a cookie
Domain	The domain name of the document server
Embeds	An array containing all the plugins in a document
FgColor	The text color attribute set in the <body> tag
FileCreatedDate	Use this value to show when the loaded HTML file was created

fileModifiedDate	Use this value to show the last change date of the HTML file currently loaded
lastModified	The date the file was modified last
Layers	An array containing all the layers in a document
LinkColor	The color of HTML links in the document. It is specified in the <BODY> tag.
Title	The name of the current document as described between the header <TITLE> tags.
URL	The location of the current document
VlinkColor	The color of visited links as specified in the <BODY> tag

Document Object Methods

Method	Description
Clear	This is depreciated
Close	Closes an output stream that was used to create a document object
contextual	It can be used to specify style of specific tags. The following example specified that text in blockquotes is to be blue: document.contextual(document.tags.blockquote).color = "blue"; Multiple styles may be specified in the contextual method to set the value of text in a <H3> tag that is underlined to the color blue, for example. document.contextual(document.tags.H3, document.tags.U).color = "blue";
elementFromPoint(x, y)	Returns the object at point x, y in the HTML document.
getSelection	Get the selected text (if any is selected)
open([mimeType])	Opens a new document object with the optional mime type.
write(expr1[,expr2...exprN])	Add data to a document. Writes the values passed to the write function to the document. For example document.write("<H3>") document.writeln("This is a Header") document.write("</H3>")
writeln(expr1[,expr2...exprN])	Adds the passed values to the document appended with a new line character.

• Predefined Objects

Let us consider some of the most frequently used predefined objects provided in Javascript.

• Math object

In most applications we need to perform calculations, whether it is accounting software or scientific software. Programmers are often typecast as good mathematicians. Every mathematician needs a calculator sometimes, or in the case of JavaScript, the Math object. If we want to calculate "2.5 to the power of 8" or "Sin0.9" in your script, then JavaScript's virtual calculator is what you need. The Math object contains a number of manipulating functions:

The Math object

Methods	Description
Math.abs(x)	Return absolute value of x
Math.acos(x)	Return arc cosine of x in radians
Math.asin(x)	Return arc sine of x in radians
Math.atan(x)	Return arc tan of x in radians

Math.atan2(x, y)	Counterclockwise angle between x axis and point (x,y)
Math.ceil(x)	Rounds a number up
Math.cos(x)	Trigonometric cosine of x (x in radians)
Math.exp(x)	Exponential method e^x
Math.floor(x)	Rounds a number down
Math.log(x)	Natural logarithm of x (base e)
Math.max(a, b)	Returns the larger of two values
Math.min(a, b)	Returns the smaller of two values
Math.pow(x, y)	Returns x^y
Math.round(x)	Rounds x to the closest integer
Math.sin(x)	Trigonometric sine of x (x in radians)
Math.sqrt(x)	Square root of x
Math.tan(x)	Trigonometric tangent of x (x in radians)
Properties	Description
Math.E	Euler's constant (~ 2.718)
Math.LN10	Natural logarithm of 10 (~ 2.302)
Math.LN2	Natural logarithm of 2 (~ 0.693)
Math.LOG10E	Base 10 logarithm of Euler's constant (~ 0.0434)
Math.LOG2E	Base 2 logarithm of Euler's constant (~ 1.442)
Math.PI	The ratio of a circle's circumference to its diameter (~ 3.141)
Math.SQRT1_2	Square root of 0.5 (~ 0.707)
Math.SQRT2	Square root of 2.0 (~ 1.414)

Let us have Javascript perform some mathematical calculations:

```
//calculate e5
Math.exp(5)
//calculate cos( 2PI)
Math.cos(2*Math.PI)
```

The "with" statement

If you intend to invoke Math multiple times in your script, a good statement to remember is "with." Using it you can omit the "Math." prefix for any subsequent Math properties/methods:

```
with (Math){
var x= sin(3.5)
var y=tan(5)
var result=max(x,y)
}
```

- **Date Object**

The Date object is used to work with dates and times.

Creating a Date Instance

You should create an instance of the Date object with the "new" keyword. The following line of code stores the current date in a variable called "my_date":

```
var my_date=new Date( )
```

After creating an instance of the Date object, you can access all the methods of the object from the "my_date" variable. If, for example, you want to return the date (from 1-31) of a Date object, you should write the following:

```
my_date.getDate( )
```

You can also write a date inside the parentheses of the Date() object. The following line of code shows some of the date formats available.

```
new Date("Month dd, yyyy hh:mm:ss"), new Date("Month dd, yyyy"), new  
Date(yy,mm,dd,hh,mm,ss), new Date(yy,mm,dd), new Date(milliseconds)
```

Here is how you can create a Date object for each of the ways above:

```
var my_date=new Date("October 12, 1988 13:14:00"), var my_date=new  
Date("October 12, 1988"), var my_date=new Date(88,09,12,13,14,00), var  
my_date=new Date(88,09,12), var my_date=new Date(500)
```

- **Some Date Methods**

Methods	Explanation
Date()	Returns a Date object
getDate()	Returns the date of a Date object (from 1-31)
getDay()	Returns the day of a Date object (from 0-6. 0=Sunday, 1=Monday, etc.)
getMonth()	Returns the month of a Date object (from 0-11. 0=January, 1=February, etc.)
getFullYear()	Returns the year of the Date object (four digits)
getHours()	Returns the hour of the Date object (from 0-23)
getMinutes()	Returns the minute of the Date object (from 0-59)
getSeconds()	Returns the second of the Date object (from 0-59)

Examples:

Date

Returns the current date, including date, month, and year. Note that the getMonth method returns 0 in January, 1 in February etc. So add 1 to the getMonth method to display the correct date.

```
<HTML>
```

```
<BODY>
```

```
<SCRIPT LANGUAGE="JAVASCRIPT">
```

```
var d = new Date()  
document.write("date = ")  
document.write(d.getDate( ))  
document.write(" ")  
document.write(d.getMonth() + 1)  
document.write(" ")  
document.write(d.getFullYear())  
document.write("time = ")  
document.write(d.getHours())  
document.write(" ")  
document.write(d.getMinutes() + 1)  
document.write(" ")
```

```

        document.write(d.getSeconds())
</SCRIPT>
</BODY>
</HTML>

```

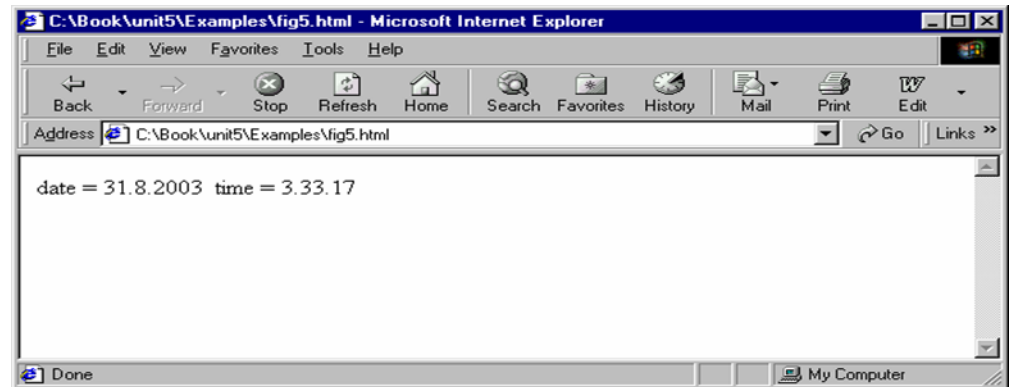


Figure 3.3: Using the Date Object

Time

Returns the current time for the timezone in hours, minutes, and seconds as shown in Figure 3.3. To return the time in GMT use `getUTCHours`, `getUTCMinutes` etc.

• Array Object

An Array object is used to store a set of values in a single variable name. Each value is an element of the array and has an associated index number. You can refer to a particular element in the array by using the name of the array and the index number. The index number starts at zero.

You create an instance of the Array object with the "new" keyword. `var family_names=new Array(5)` The expected number of elements goes inside the parentheses, in this case it is 5.

You assign data to each of the elements in the array like this:

```

family_names[0]="Sharma" family_names[1]="Singh"
family_names[2]="Gill" family_names[3]="Kumar" family_names[4]="Khan"

```

The data can be retrieved from any element by using the index of the array element you want:

```
Mother=family_names[0] father=family_names[1]
```

The Most Commonly Used Array Methods

Methods	Explanation
Length	Returns the number of elements in an array. This property is assigned a value when an array is created
reverse()	Returns the array reversed
slice()	Returns a specified part of the array
Sort()	Returns a sorted array

- **History Object**

The History object is a predefined JavaScript object which is accessible through the history property of a window object. The window.history property is an array of URL strings, which reflect the entries in the History object. The History object consists of an array of URLs, accessible through the browser's Go menu, which the client has visited within a window. It is possible to change a window's current URL without an entry being made in the History object by using the location.replace method. The History object contains 4 properties and 3 methods as summarized here:

Property	Summary for History Object
Current	Specifies the URL of the current history entry.
Next	Specifies the URL of the next history entry.
Previous	Specifies the URL of the previous history entry.
Length	Reflects the number of entries in the history list.

Method	Summary for History Object
Back()	Loads the previous URL in the history list.
Forward()	Loads the next URL in the history list.
Go()	Loads a URL from the history list.

- **Location Object**

The Location object is part of a Window object and is accessed through the window.location property. It contains the complete URL of a given Window object, or, if none is specified, of the current Window object. Syntax : All of its properties are strings representing different portions of the URL, which generally takes the following form:

<protocol>[//<host>[:<port>]]/<pathname>[<hash>][<search>]

You can create a Location object by simply assigning a URL to the location property of an object:

Syntax: window.location = "file:///C:/Projects"

Property	Description
Hash	The hash property is a string beginning with a hash (#), that specifies an anchor name in an HTTP URL
Host	The host property is a string comprising the hostname and port strings.
hostname	The hostname property specifies the server name, subdomain and domain name (or IP address) of a URL.
Href	The href property is a string specifying the entire URL, and of which all other link properties are substrings.
pathname	The pathname property is a string portion of a URL specifying how a particular resource can be accessed.
Port	The port property is a string specifying the communications port that the server uses.
protocol	The protocol property is the string at the beginning of a URL, up to and including the first colon (:), which specifies the method of access to the URL.

Property	Description
search	The search property is a string beginning with a question mark that specifies any query information in an HTTP URL.

Some Common Methods

Method	Description
reload	The reload method forces a reload of the windows current document, i.e. the one contained in the Location.href
Replace	The replace method replaces the current history entry with the specified URL. After calling the replace method to change the history entry, you cannot navigate back to the previous URL using the browser's Back button

☛ Check Your Progress 1:

1. Write a JavaScript code block using arrays and generate the current date in words. This should include the day, the month and the year.

.....

.....

.....

.....

2. Write JavaScript code that converts the entered text to upper case.

.....

.....

.....

.....

3. Write JavaScript code to validate Username and Password. Username and Password are stored in variables.

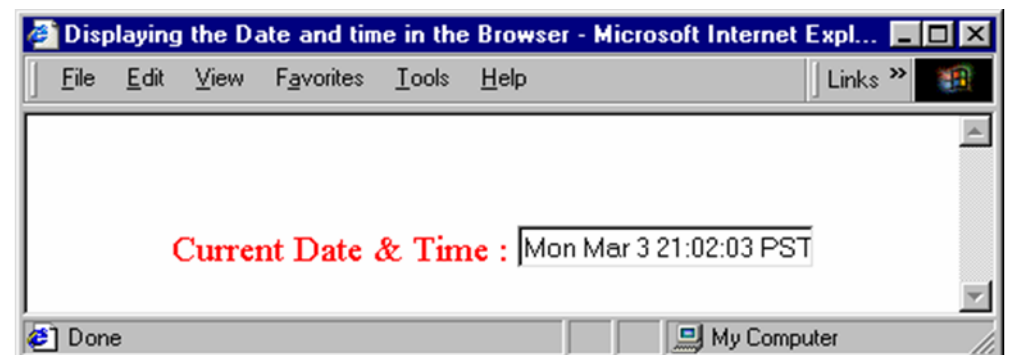
.....

.....

.....

.....

4. Design the following Web page



3.6 MESSAGEBOX IN JAVASCRIPT

In this section, we cover the topic of dialog boxes. This topic is important for understanding the way parameters are passed between different windows. This mechanism is a key component of some of the new capabilities of Internet Explorer 5.5 and 6.0, such as print templates. You use dialog boxes all over. The alert box is probably the most popular one.

3.6.1 Dialog Boxes

In this the user cannot switch to the window without closing the current window. This kind of dialog box is referred to as a modal dialog box. You create a modal dialog box with `showModalDialog()`.

Syntax: `showModalDialog ("Message")`

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JAVASCRIPT">

function fnOpenModal(){ window.showModalDialog("test.htm")
}

</SCRIPT>
</HEAD>
<BODY>
<FORM NAME = IGNOU>
<INPUT TYPE="button" VALUE="Push Me" onclick="fnOpenModal()">
</BODY>
</HTML>
```

3.6.2 Alert Boxes

Alert boxes can be utilized for a variety of things, such as to display when an input field has not been entered properly, to display a message on document open or close, or to inform someone that they have clicked a link to leave your site. Whatever the use, the construction is essentially the same.

Syntax : `alert("message")`

The following example will generate a simple alert box based on clicking either a link or a form button. Output is shown in Figure 3.4.

```

<html>
<title>Codeave.com(JavaScript: Alert Box)</title>
<body bgcolor="#ffffff">
<!-- Example of a form button that will open an alert box -->
<form>
<input type="button" value="Open Alert Box"
onClick='alert("Place your message here... \n Click OK to continue. "')>
</form>
<p>
<!-- Example of a link that will open an alert box -->
<a href='javascript:onClick=alert("Place your message here... \n Click OK to
continue. "')>
Open Alert Box</a>

</body>
</html>

```



Figure 3.4: Using an Alert Box

3.6.3 Confirm Boxes

The JavaScript confirm alert box differs from a regular alert box in that it provides two choices to the user, OK and Cancel. Typically, you'll see confirmation boxes utilized on links where the destination is outside the domain of the page you are currently on or to alert you of potentially objectionable material. The following example will display a confirmation alert box when either a link or form button is clicked. Once either OK or Cancel are selected an alert box will follow to display which was chosen.

```

<html>
<body bgcolor="#FFFFFF">
<title>CodeAve.com(JavaScript: Confirm Alert Box)</title>
<script language="JavaScript">
<!--
function confirm_entry()
{
input_box=confirm("Click OK or Cancel to Continue"); if (input_box==true)
{
// Output when OK is clicked alert ("You clicked OK");
}

-->
else
}

```

```

{
// Output when Cancel is clicked alert ("You clicked cancel");
}

</script>
Click <a href="JavaScript:confirm_entry()">here</a>
<p>
<form onSubmit="confirm_entry()">
<input type="submit" >
</form>
</body>
</html>

```

3.6.4 Prompt Boxes

The prompt box allows the user to enter information. The benefits of using a prompt are fairly limited and the use of forms would often be preferred (from a user perspective).

The prompt box title text cannot be changed and the same applies to button text. You can have 2 lines of text using \n where the new line starts (please note that the opera browser up to version 7 will only display 1 line of text)

The text entry field is similar to a form input type="text". The 2 buttons are OK and Cancel. The value returned is either the text entered or null.

The syntax for the prompt is prompt("your message",""); (script tags omitted)
 "your message","" the "," holds the default text value "" = empty.

3.7 JAVASCRIPT WITH HTML

3.7.1 Events

Events are actions that can be detected by JavaScript. An example would be the onMouseOver event, which is detected when the user moves the mouse over an object. Another event is the onLoad event, which is detected as soon as the page is finished loading. Usually, events are used in combination with functions, so that the function is called when the event happens. An example would be a function that would animate a button. The function simply shifts two images. One image that shows the button in an "up" position, and another image that shows the button in a "down" position. If this function is called using an onMouseOver event, it will make it look as if the button is pressed down when the mouse is moved over the image.

3.7.2 Event Handlers

An event handler executes a segment of a code based on certain events occurring within the application, such as onLoad or onClick. JavaScript event handlers can be divided into two parts: interactive event handlers and non-interactive event handlers. An interactive event handler is the one that depends on user interaction with the form or the document.

For example, `onMouseOver` is an interactive event handler because it depends on the user's action with the mouse. An example of a non-interactive event handler is `onLoad`, as it automatically executes JavaScript code without the need for user interaction. Here are all the event handlers available in JavaScript.

- **`onLoad` and `onUnload`**

`onLoad` and `onUnload` are mainly used for popups that appear when the user enters or leaves the page. Another important use is in combination with cookies that should be set upon arrival or when leaving your pages.

For example, you might have a popup asking the user to enter his name upon his first arrival to your page. The name is then stored in a cookie. Furthermore, when the visitor leaves your page a cookie stores the current date. Next time the visitor arrives at your page, it will have another popup saying something like: "Welcome Ajay, this page has not been updated since your last visit 8 days ago".

Another common use of the `onLoad` and `onUnload` events is in making annoying pages that immediately open several other windows as soon as you enter the page.

- **`OnFocus` and `onBlur`**

The `onFocus` event handler executes the specified JavaScript code or function on the occurrence of a focus event. This is when a window, frame or form element is given the focus. This can be caused by the user clicking on the current window, frame or form element, by using the TAB key to cycle through the various elements on screen, or by a call to the `window.focus` method. Be aware that assigning an alert box to an object's `onFocus` event handler will result in recurrent alerts as pressing the 'OK' button in the alert box will return focus to the calling element or object.

The `onFocus` event handler uses the Event object properties.

`type` - this property indicates the type of event.

`target` - this property indicates the object to which the event was originally sent.

The following example shows the use of the `onFocus` event handler to replace the default string displayed in the text box. Note that the first line is HTML code and that the text box resides on a form called 'myForm'. Output is shown in Figure 3.5.

Syntax : `onfocus = script`

```
<HTML>
<HEAD>
</HEAD>
<BODY>
```

```

<FORM NAME = "myform" >
<input type="text" name="myText" value="Give me focus" onFocus =
"changeVal()">
</BODY>

<SCRIPT LANGUAGE="JAVASCRIPT">

s1 = new String(myform.myText.value) function changeVal() {
s1 = "I'm feeling focused" document.myform.myText.value =
s1.toUpperCase()
}
</SCRIPT>
</HTML>

```

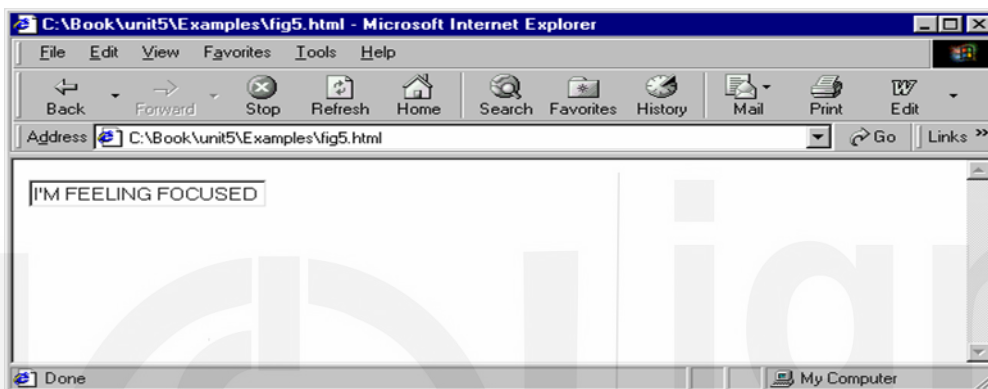


Figure 3.5: Using the onFocus Method

Syntax: onBlur = script

The onBlur event handler executes the specified JavaScript code or function on the occurrence of a blur event. This is when a window, frame or form element loses focus. This can be caused by the user clicking outside of the current window, frame or form element, by using the TAB key to cycle through the various elements on screen, or by a call to the window.blur method.

The onBlur event handler uses the Event object properties.

type - this property indicates the type of event.

target - this property indicates the object to which the event was originally sent.

The following example shows the use of the onBlur event handler to ask the user to check that the details given are correct. Note that the first line is HTML code.

```

<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM NAME = "myform" >

```

```

Enter email address <INPUT TYPE="text" VALUE="" NAME="userEmail"
onBlur=addCheck( )>
</BODY>
<SCRIPT LANGUAGE="JAVASCRIPT">
function addCheck() {
alert("Please check your email details are correct before submitting")
}
</SCRIPT>
</HTML>

```

- **OnError**

The onError event handler executes the specified JavaScript code or function on the occurrence of an error event. This happens when an image or document causes an error during loading. A distinction must be made between a browser error, when the user types in a non-existent URL, for example, and a JavaScript runtime or syntax error. This event handler will only be triggered by a JavaScript error, not a browser error.

Apart from the onError handler triggering a JavaScript function, it can also be set to onError="null", which suppresses the standard JavaScript error dialog boxes. To suppress JavaScript error dialogs when calling a function using onError, the function must return true (Example 2 below demonstrates this).

There are two things to bear in mind when using window.onerror. First, this only applies to the window containing window.onerror, not any others, and secondly, window.onerror must be spelt all lower-case and contained within <script> tags; it cannot be defined in HTML (this obviously does not apply when using onError with an image tag, as in example 1 below).

The onFocus event handler uses the Event object properties.

type - this property indicates the type of event.

target - this property indicates the object to which the event was originally sent.

The first example suppresses the normal JavaScript error dialogs if a problem arises when trying to load the specified image, while Example 2 does the same, but applied to a window, by using a return value of true in the called function, and displays a customized message instead.

Syntax : Object.onError = [function name]

Example 1

```
<IMG NAME="imgFaculty" SRC="dodgy.jpg onError="null">
```

Example 2

```

<script type="text/javascript" language="JavaScript"> s1 = new
String(myForm.myText.value) window.onerror=myErrorHandler
function myErrorHandler() { alert('A customized error message') return true

```

```

}
</script>
<body onload=nonexistantFunc( )>

```

Check Your Progress 2:

1. Design a Web page that displays a welcome message whenever the page is loaded and an Exit message whenever the page is unloaded.

.....

.....

.....

.....

3.8 FORMS

Each form in a document creates a form object. Since a document can contain more than one form, Form objects are stored in an array called forms.

3.8.1 Forms Array

Using the forms[] array we access each Form object in turn and show the value of its name property in a message box. Let us have a look at an example that uses the forms array. Here we have a page with three forms on it.

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE=JavaScript>
function window_onload()
{
var numberForms = document.forms.length; var formIndex;
for (formIndex = 0; formIndex < numberForms; formIndex++)
{
alert(document.forms[formIndex].name);
}
}
</SCRIPT>
</HEAD>
<BODY LANGUAGE=JavaScript onLoad="window_onload()">
<FORM NAME="form1">
<P>This is inside form1</P>
</FORM>
<FORM NAME="form2">
<P>This is inside form2</P>
</FORM>
<FORM NAME="form3">
<P>This is inside form3</P>

```

```
</FORM>
</BODY>
</HTML>
```

Within the body of the page we define three forms. Each form is given a name, and contains a paragraph of text. Within the definition of the <BODY> tag, the window_onload() function is connected to the window object's onLoad event handler.

```
<BODY LANGUAGE=JavaScript onLoad="return window_onload( )">
```

This means that when the page is loaded, our window_onload() function will be called. The window_onload() function is defined in a script block in the HEAD of the page. Within this function we loop through the forms[] array. Just like any other JavaScript array, the forms[] array has a length property, which we can use to determine how many times we need to loop. Actually, as we know how many forms there are, we could just write the number in. However, here we are also demonstrating the length property, since it is then easier to add to the array without having to change the function. Generalizing your code like this is a good practice to follow.

The function starts by getting the number of Form objects within the forms array and stores it in the variable numberForms.

```
function window_onload( )
{
  var numberForms = document.forms.length;
```

Next we define a variable, formIndex, to be used in our for loop. After this comes the for loop itself.

```
var formIndex;
for (formIndex = 0; formIndex < numberForms; formIndex++)
{
  alert(document.forms[formIndex].name);
}
```

Remember that since the indices for arrays start at zero, our loop needs to go from an index of 0 to an index of numberForms - 1. We do this by initializing the formIndex variable to zero, and setting the condition of the for loop to formIndex < numberForms.

Within the for loop's code, we pass the index of the desired form (that is, formIndex) to document.forms[], which gives us the Form object at that array index in the forms array. To access the Form object's name property, we put a dot at the end and the name of the property, name.

- **Form Object**

Form is a property of the document object. This corresponds to an HTML input form constructed with the FORM tag. A form can be submitted by

calling the JavaScript submit method or clicking the form SUBMIT button.
Some of the form properties are:

- Action - This specifies the URL and CGI script file name the form is to be submitted to. It allows reading or changing the ACTION attribute of the HTML FORM tag.
- Button – An object representing a GUI control.
- Elements - An array of fields and elements in the form.
- Encoding - This is a read or write string. It specifies the encoding method the form data is encoded in, before being submitted to the server. It corresponds to the ENCTYPE attribute of the FORM tag. The default is "application/x-www-form-urlencoded". Other encoding includes text/plain or multipart/form-data.
- Length - The number of fields in the elements array, that is, the length of the elements array.
- Method - This is a read or write string. It has the value "GET" or "POST".
- Name - The form name. Corresponds to the FORM Name attribute.
- Password – An object representing a password field.
- Radio – An object representing a radio button field.
- Reset - An object representing a reset button.
- Select - An object representing a selection list.
- Submit - An object representing a submit button.
- Target - The name of the frame or window to which the form submission response is sent by the server.
Corresponds to the FORM TARGET attribute.
- Text - An object representing a text field.
- Textarea - An object representing a text area field.

Some of the **Form Element Properties** are:

- Name – It provides access to an element's name attribute. It applies to all form elements.
- Type – Identifies the object's type. It applies to all form elements.
- Value - Identifies the object's value. It applies to all, button, checkbox, hidden, password, radio, reset, submit, text or textarea.
- Checked - Identifies whether the element is checked. It applies to checkbox and radio.
- Default checked - Identifies whether the element is checked by default. It applies to checkbox and radio.
- Default value – Identifies the object's default value. It applies to password, submit and textarea.
- Length - Identifies the length of a select list. It applies to select.
- Options – An array that identifies the options supported by the select list. It applies to select.

Syntax : <FORM Name = "myform" Action = "mailto:subscribe@abc.com"
Method
= POST Enctype = "multipart/form-data" onsubmit = "return check()" >

<HTML>

```

<HEAD>
<SCRIPT LANGUAGE = "javascript">
function check ()
{ if (document.myform.email.value == "") {
alert("please enter your email-id");
return false; }
}
</SCRIPT>
</HEAD>
<BODY>
Enter your Email to subscribe : <p>
<FORM Name = "myform" Action = "mailto:subscribe@abc.com" Method =
POST
Enctype = "multipart/form-data" onsubmit = "return check()" >
<Input type = "text" Name = "email">
<Input type = "submit" value = "submit">
</FORM>
</BODY>
</HTML>

```

The above code is used for obtaining the user's email-id. In case the user clicks the submit button without entering the email-id in the desired text field, he sees a message box indicating that the email-id has not been entered.

Case Study

Design a Web page with appropriate functionality to accept an order for a fast food outlet. It should check if the user has entered a valid name and email-id. It should also calculate the value of the order.

```

<HTML>
<HEAD>
<TITLE>Donald Duck</TITLE>
<SCRIPT Language="JavaScript"> var m;
function chk_name()
{
if( document.form1.txt_name.value == "")
{
alert("Please enter your name"); document.form1.txt_name.focus();
}
}
function chk_email()
{
var str = document.form1.txt_email.value ; var i;

if ( document.form1.txt_email.value == "")
{
alert("Please enter your Email-ID"); document.form1.txt_email.focus();
}
i = str.indexOf("@"); if ( i < 0)
{
alert ("Please enter a valid Email-Id");
}
}
}

```

```

}
}
function mainitem(F1)
{ var z=" ";

for(j=0;j<3;j++)
{
for(i=0;i<F1.elements[j].length;i++)
{

}
}
m=z;
}

if (F1.elements[j][i].selected)
{
var y=F1.elements[j].options[i].value; z=z+"\n"+y;
F1.elements[3].value=z;
}

function cal(F1)
{ var d=0;
for(j=0;j<3;j++)
{
for(i=0;i<F1.elements[j].length;i++)
{
if (F1.elements[j][i].selected)
{
var y=F1.elements[j].options[i].value; s=new String(y);
var a=s.indexOf(">");
var b=s.substring(a+1,a+3);
c=parseInt(b); d=d+c;
}
}
}
p="Total cost of the selected items="+d; m=m+"\n"+p;
F1.elements[3].value=m;
}

function clr(F1)
{
F1.elements[3].value=" ";
}
</SCRIPT>
</HEAD>
<BODY>
<h2><font color="blue"><center> Welcome to the World renowned online
Fast Food
Center </font>

<font color="red"> Donald Duck ! </center></font></h2>

```

<Form name="form1" ACTION = "mailto:dlc@ignou.ac.in" METHOD = POST> Select the Menu Items of your choice and then click on the Total Cost to find the bill amount-

<Table >

<TR valign=top ><td> Major dishes :

<select name="s1" MULTIPLE onBlur="mainitem(this.form)">

<option value="Onion cheese capsicum Pizza->300" selected> Onion cheese capsicum Pizza

<option value="Onion mushroom Pizza->200"> Onion mushroom Pizza

<option value="Chicken Tikka Pizza->460"> Chicken Tikka Pizza

<option value="Cheese Pizza->150"> Cheese Pizza

</select>

</td>

<td> </td><td> </td>

<td>

Soups :

<select name="s2" MULTIPLE onBlur="mainitem(this.form)">

<option value="Tomato Soup->70"> Tomato Soup

<option value="Sweet corn Soup->80">Sweet corn Soup

<option value="Sweet n Sour soup->90">Sweet n Sour soup

<option value="Mixed veg soup->50">Mixed veg soup

</select>

</td>

<td> </td><td> </td>

<td>

Miscellaneous :

<select name="s3" onBlur="mainitem(this.form)">

<option value=" ">'Desserts'

<option value="Milkshakes->35">Milkshakes

<option value="Soft drinks->20">Soft drinks

<option value="Ice cream sodas->25">Softy

</select>

</td>

<td> </td><td> </td>

</TR>

</Table>

<Table>

<TR valign=top><td>

The items selected form the Menu are :

<TEXTAREA Name="TA1" Rows=10 Cols=50>

</TEXTAREA>

</td>

<td> </td><td> </td>

<td>

<input type="button" Value="Total Cost" onClick="cal(this.form)">

<input type="button" Value="Clear" onClick="clr(this.form)">

</td>

</TR>

</Table>

<HR noshade>

<h2> Personal Details </center></h2>

```

<Table >
<TR valign=top ><td> Name:<BR>
<Input Type = "Text" Name = "txt_name" Onblur = "chk_name()">

<BR><BR></td>
<td> </td><td> </td>
<td>
Contact Address :<br>
<TEXTAREA Name="TA2" Rows=3 Cols=10>
</TEXTAREA>
<BR><BR></td>
<td> </td><td> </td>
<td>
Email :<BR>
<Input Type = "Text" Name = "txt_email" Onblur = "chk_email()">
<BR><BR></td>
<td> </td><td> </td>

</TR>
</Table>
<Table>
<TR valign=top ><td> Phone :<BR>
<Input Type = "Text" Name = "txt_phone">
<BR><BR></td>
<td> </td><td> </td>
<td>
<BR>
<Input Type = "submit" Name = "btnsubmit" Value = "    Submit">
<BR><BR></td>
<td> </td><td> </td>
</TR>
</Table>
</Form>
</BODY>
</HTML>

```

☛ Check Your Progress 3:

1. Design the following Web page.

The screenshot shows a Microsoft Internet Explorer window titled "Password Validation - Microsoft Internet Explorer". The address bar shows "C:\delete\Part_2_JS\Chap08\Handon3.html". Below the browser window, there is a form titled "FORMS - Microsoft Internet Explorer". The form contains two sections: "FIRST FORM: Survey Form 1" and "SECOND FORM: Survey Form 2".

FIRST FORM: Survey Form 1

First Name :

☐ Fresher ☐ Experienced

SECOND FORM: Survey Form 2

Name :

Password :

☐ Employed ☐ Studying

2. Design the following Web page.

The screenshot shows a Microsoft Internet Explorer window titled "FORMS - Microsoft Internet Explorer". The form contains the following fields and controls:

Client Name :

Client Address:

Client E-mail Address :

☐ Male ☐ Female

☐ Employed

3. Design the following Web page in such a way that selecting any option from the radio button displays the appropriate result in the Result box.

The screenshot shows a Microsoft Internet Explorer window with the title "Working with Radio Buttons - Microsoft Internet Explorer". The address bar shows "C:\Book\unit5\Examples\fig5.html". The form contains the following elements:

- A text input field labeled "Value:" with the value "0".
- A label "Action:" followed by two radio buttons: "Double" and "Square".
- A text input field labeled "Result:".

The status bar at the bottom shows "Done" and "My Computer".

4. Suppose you have an order form that updates automatically each time you enter or change a quantity, the cost for that item and the total cost are updated. Each field must be validated. Design the following web page. (Case Study)

The screenshot shows a Microsoft Internet Explorer window with the title "Order Form - Microsoft Internet Explorer". The address bar shows "C:\Book\unit5\Examples\case5_2.html". The form is titled "Online Cake Order Form" and contains the following elements:

- Form fields for "Name:", "Phone:", and "E-mail address:".
- A "Shipping Address:" section with a text input field and a "Products to Order:" section.
- A table for "Products to Order" with columns for "Qty.", "Cost:", and the product name.
- A "Total Cost:" field.
- A "Method of Payment:" dropdown menu with options "Check or Money Order".
- A "Credit Card or Check Number:" field.
- "Send Your Order" and "Clear Form" buttons.

The status bar at the bottom shows "Done" and "My Computer".

3.9 SUMMARY

In this section we have learned about the major features of JavaScript. Besides normal programming constructs, datatypes and variables, we have also discussed the most commonly used objects. These are: Document, Math, Date, History, Location etc. The Submit and Reset objects are used to submit the information to the server and reset the information entered, respectively. Finally, we discussed a very important object, the Form object.

3.10 SOLUTIONS / ANSWERS

Check Your Progress 1:

1.

```
<HTML>
<HEAD>

<TITLE> Date validations </TITLE>
<SCRIPT>
var monthNames = new Array(12);
monthNames[0]= "January";
monthNames[1]= "February";
monthNames[2]= "March";
monthNames[3]= "April";
monthNames[4]= "May";
monthNames[5]= "June";
monthNames[6]= "July";
monthNames[7]= "August";
monthNames[8]= "September";
monthNames[9]= "October";
monthNames[10]= "November";
monthNames[11]= "December";

var dayNames = new Array(7);
dayNames[0]= "Sunday";
dayNames[1]= "Monday";
dayNames[2]= "Tuesday";
dayNames[3]= "Wednesday";
dayNames[4]= "Thursday";
dayNames[5]= "Friday";
dayNames[6]= "Saturday";

function customDateString (m_date)
{
    var daywords = dayNames[m_date.getDay()];
    var theday = m_date.getDate();
    var themonth = monthNames[m_date.getMonth()];
```



```

    var theyear = m_date.getYear();
    return daywords + ", " + themonth + " " + theday + ", " + theyear;

}
</SCRIPT>
</HEAD>

<BODY>
<H1>WELCOME!</H1>
<SCRIPT>
document.write(customDateString( new Date()))
</SCRIPT>
</BODY>
</HTML>

```

2.

```

<HTML >
<HEAD>
<SCRIPT language="JavaScript"> function checkData(column_data)
{
if (column_data != "" && column_data.value !=
column_data.value.toUpperCase( ))
{
column_data.value = column_data.value.toUpperCase( )
}
}
</SCRIPT>
</HEAD>

```

```

<BODY>
<FORM>
<INPUT TYPE="text" NAME="collector" SIZE=10
onChange="checkData(this)">
<INPUT TYPE="text" NAME="dummy" SIZE=10>
</FORM>
</BODY>
</HTML>

```

3.

```

<HTML>
<HEAD>
<TITLE>Password Validation</TITLE>
<SCRIPT language="JavaScript">
<!--
var userpassword = new Array(4);
userpassword[0]= "Ajay";
userpassword[1]= "ajay";
userpassword[2]= "Rohan";

```

```

userpassword[3]= "rohan";

function checkOut()
{

var flag =0;
var flag1 =0;
var i =0;
var j =0;

for (x=0; x<document.survey.elements.length; x++)
{
if (document.survey.elements[x].value == "")
{
alert("You forgot one of the required fields. Please try again") return;
}
}
var user= document.survey.elements[0].value
var password = document.survey.elements[1].value
while(i<=3)
{
if(userpassword[i] == user)
{
} i+=2;
}
if(flag == 0)
{

j=i; j++;
if(userpassword[j] == password)
{
flag=1; break;
}

alert("Please enter a valid user name and password") return;

return;
}
}

}
else
{
alert("Welcome !!\n" +document.forms[0].Username.value);
}

</SCRIPT>
</HEAD>
<BODY>
<FORM ACTION="" method="POST" NAME="survey" onSubmit="return
checkOut(this.form)">

```

```

<INPUT TYPE="TEXT" NAME="Username" SIZE="15"
MAXLENGTH="15">
User Name
<BR><INPUT TYPE="PASSWORD" NAME="Pasword"
SIZE="15">Password
<BR><INPUT TYPE="SUBMIT" VALUE="Submit"><INPUT
TYPE="RESET"
VALUE="Start Over">
</FORM>
</BODY>
</HTML>

```

4.

```

<HTML>
<HEAD>
<TITLE>Displaying the Date and time in the Browser</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
function begin(form)
{
form_name = form; time_out=window.setTimeout("display_date()",500)
}

function display_date()
{
form_name.date.value=new Date();
time_out=window.setTimeout("display_date()",1000)
}
function display_clock()
{
document.write('<FONT COLOR=red SIZE=+1><FORM
NAME=time_form><CENTER><BR><BR>Current Date & Time :')
document.write(' <INPUT NAME=date size=19
value=""></FORM></FONT></CENTER>')
begin(document.time_form)
}
display_clock()
</SCRIPT>
</BODY>
</HTML>

```

Check Your Progress 2:

1.

```

HTML>
<HEAD>
<TITLE>Creating and Using User Defined Functions</TITLE>
<SCRIPT LANGUAGE="JavaScript">

```

```

var name = ""; function hello( )
{
name = prompt('Enter Your Name:', 'Name'); alert('Greetings ' + name + ',
Welcome to my page!');
}

function goodbye( )
{
alert('Goodbye ' + name + ', Sorry to see you go!');
}
</SCRIPT>
</HEAD>
<BODY onLoad="hello( );" onUnload="goodbye( );">
<IMG SRC="Images\Pinkwhit.gif ">
</BODY>
</HTML>

```

☛ Check Your Progress 3:

1.

```

<HTML>
<HEAD>
<TITLE>FORMS</TITLE>
<!-- This code allows to access the Form objects Elements Array //-->
<SCRIPT Language="JavaScript"> function Ver(form1)
{
v=form1.elements.length;
if (form1.elements[3].name=="Button1")
{
alert('First form name : '+document.forms[0].name);
alert('No. of Form elements of ' +document.forms[0].name + ' = '+v);
}
else if (form1.elements[4].name=="Button2")
{
alert('Second form name : '+document.forms[1].name);
alert('No. of Form elements of ' +document.forms[1].name + ' = '+v);
}
for(i=0;i<v;i++)
alert(form1.elements[i].name+ ' is at position '+i);
}
</SCRIPT>
</HEAD>

```

```

<BODY>
<FORM Name="Survey Form 1">
FIRST FORM: <i><b>Survey Form 1 </b></i><BR><BR>
First Name : <Input Type=Text Name="Text1" Value=""><BR><BR>
<Input Type="radio" Name="Radio1" Value=""> Fresher

```

```

<Input Type="radio" Name="Radio1" Value=""> Experienced<BR><BR>
<Input Type="Button" Name="Button1" Value="Click1"
onClick="Ver(form)">
</FORM>

```

```

<FORM Name="Survey Form 2">
SECOND FORM: <i><b> Survey Form 2 </b></i><BR><BR> Name : <Input
Type="Text" Name="Text2" Value=""> <BR><BR>
Password : <Input Type="Password" Name="Pass2" Value="">

```

```

<BR><BR>

```

```

<BR><BR>

```

```

<Input Type="CheckBox" Name="Check1" Value="" > Employed
<Input Type="CheckBox" Name="Check2" Value="" > Studying

```

```

<Input Type="Button" Name="Button2" Value="Click2"

```

```

onClick="Ver(form)">

```

```

</FORM>

```

```

</BODY>

```

```

</HTML>

```

2.

```

<HMTL>

```

```

<HEAD>

```

```

<TITLE>FORMS</TITLE>

```

```

<!-- This code checks the Checkbox when the button is clicked //-->

```

```

<SCRIPT Language='JavaScript'> function Chk(f1)

```

```

{

```

```

f1.Check.checked=true;

```

```

alert(" The Checkbox just got checked "); f1.Check.checked=false;

```

```

f1.Radio[0].checked=true; f1.Radio[1].checked=false;

```

```

alert(" The Radio button just got checked ");

```

```

}

```

```

</SCRIPT>

```

```

</HEAD>

```

```

<BODY>

```

```

<FORM>

```

```

Client Name : <Input Type=Text Name="Text" Value=""><BR><BR>

```

```

Client Address: <Input Type=Text Name="Text1" Value=""> <BR><BR>

```

```

Client E-mail Address :<Input Type=Text Name="Text2"

```

```

Value=""><BR><BR>

```

```

<Input Type="radio" Name="Radio" Value=""> Male
<Input Type="radio" Name="Radio" Value=""> Female<BR><BR>
<Input Type="CheckBox" Name="Check" Value=""> Employed <BR><BR>
<Input Type="Button" Name="Bt" Value="Set Element Array Value"
onClick="Chk(this.form)">
</FORM>
</BODY>
</HTML>

```

3.

```

<HTML>
<HEAD>
<TITLE> Working with Radio Buttons </TITLE>
<SCRIPT LANGUAGE="JavaScript">
function calculate(form)
{
if(form.elements[2].checked)
{

}
else
{

form.result.value = form.entry.value * form.entry.value;

form.result.value = form.entry.value * 2;
}
}
</SCRIPT>
</HEAD>
<BODY>
<FORM >
<CENTER><BR>
<B>Value:</B>
<INPUT TYPE="text" NAME="entry" VALUE=0>
<BR><BR>
<SPACER Size= 190>
<B>Action:<B><BR>

<SPACER Size = 225>
<INPUT TYPE="radio" NAME="action1" VALUE="twice"
onClick="calculate(this.form);">Double<BR>
<SPACER Size = 225>
<INPUT TYPE="radio" NAME="action1" VALUE="square"
onClick="calculate(this.form);">Square <BR><BR>
<B>Result:</B>
<INPUT TYPE=text NAME="result" onFocus = "this.blur();">

```

</CENTER>

</FORM>

</BODY>

</HTML>

4. (Case Study)

<HTML>

<HEAD><TITLE>Order Form</TITLE>

<SCRIPT>

// function to calculate the total cost field function Total()

{ var tot = 0;

tot += (240 * document.order.qty1.value); tot += (270 *

document.order.qty2.value); tot += (300 * document.order.qty3.value); tot +=

(600 * document.order.qty4.value); document.order.totalcost.value = tot; }

// function to update cost when quantity is changed function

UpdateCost(number, unitcost)

{ costname = "cost" + number; qtyname = "qty" + number; var q =

document.order[qtyname].value; document.order[costname].value = q *

unitcost;

Total();

}

// function to copy billing address to shipping address function CopyAddress()

{ if (document.order.same.checked)

{ document.order.shipto.value = document.order.billto.value;

}

}

//global variable for error flag var errfound = false;

//function to validate by length function ValidLength(item, len)

{ return (item.length >= len); }

//function to validate an email address function ValidEmail(item)

{ if (!ValidLength(item, 5)) return false; if (item.indexOf('@', 0) == -1)

return false; return true;

}

// display an error alert

function error(elem, text)

{

// abort if we already found an error if (errfound) return; window.alert(text);

elem.select(); elem.focus(); errfound = true;

}

// main validation function function Validate()

{

errfound = false;

if (!ValidLength(document.order.name1.value,6))

error(document.order.name1,"Invalid Name");

```

if (!ValidLength(document.order.phone.value,10))
error(document.order.phone,"Invalid Phone");
if (!ValidLength(document.order.billto.value,30))
error(document.order.billto,"Invalid Billing Address"); if
(!ValidLength(document.order.shipto.value,30))
error(document.order.shipto,"Invalid Shipping Address"); if
(!ValidEmail(document.order.email.value)) error(document.order.email,
"Invalid Email Address");
if (document.order.totalcost.value == "") error(document.order.qty1, "Please
Order at least one item."); if (document.order.payby.selectedIndex != 1)
{
if (!ValidLength(document.order.creditno.value,2))
error(document.order.creditno,"Invalid Credit/Check number");
}
return !errfound; /* true if there are no errors */ }
</SCRIPT> </HEAD>
<BODY>
<H1>Online Cake Order Form</H1>
<FORM NAME="order" onSubmit="return Validate();">
<B>Name:</B>
<INPUT TYPE="text" NAME="name1" SIZE=20>
<B>Phone: </B>
<INPUT TYPE="text" NAME="phone" SIZE=15>
<B>E-mail address:</B>
<INPUT TYPE="text" NAME="email" SIZE=20><BR><BR>
<B>Shipping Address:</B>
<BR>

<TEXTAREA NAME="shipto" COLS=40 ROWS=4
onChange="CopyAddress();"> Enter your shipping address here.
</TEXTAREA>
<B>Products to Order:</B><BR>
Qty: <INPUT TYPE="TEXT" NAME="qty1" VALUE="0" SIZE=4
onChange =
"UpdateCost(1, 240);">
Cost: <INPUT TYPE="TEXT" NAME="cost1" SIZE=6> (Rs.240 ) Vanilla
cake
<BR>
Qty: <INPUT TYPE="TEXT" NAME="qty2" VALUE="0" SIZE=4
onChange =
"UpdateCost(2, 270);">
Cost: <INPUT TYPE="TEXT" NAME="cost2" SIZE=6> (Rs.270 )
Strawberry cake
<BR>
Qty: <INPUT TYPE="TEXT" NAME="qty3" VALUE="0" SIZE=4
onChange =
"UpdateCost(3, 300);">
Cost: <INPUT TYPE="TEXT" NAME="cost3" SIZE=6> (Rs.300 ) Black
Forest
cake <BR>

```



```

Qty: <INPUT TYPE="TEXT" NAME="qty4" VALUE="0" SIZE=4
onChange =
"UpdateCost(4, 600);">
Cost: <INPUT TYPE="TEXT" NAME="cost4" SIZE=6> (Rs.600 ) Chocolate
cake
<HR> <B>Total      Cost:</B>      <INPUT      TYPE="TEXT"
      NAME="totalcost" SIZE=8><HR> <B>
Method of Payment</B>:
<SELECT NAME="payby"> <OPTION VALUE="check" SELECTED>
Check or Money Order
<OPTION VALUE="cash">Cash or Cashier's Check
<OPTION VALUE="credit">Credit Card (specify number)
</SELECT><BR> <B>Credit Card or Check Number:</B>:
<INPUT TYPE="TEXT" NAME="creditno" SIZE="20"><BR>
<INPUT TYPE="SUBMIT" NAME="submit" VALUE="Send Your Order">
<INPUT TYPE="RESET" VALUE="Clear Form">
</FORM>
</BODY>
</HTML>

```

3.11 JAVASCRIPT LAB EXERCISES

Purpose:

This section is designed to introduce learners to scripting using JavaScript. The exercises focus on developing dynamic web pages that respond to user input, perform calculations, validate forms, and manipulate webpage content interactively. The goal is to provide a foundation in client-side scripting essential for modern web development.

SESSION 1

Exercises

1. How would you write any statement using only one write() or writeln() command?
2. Embed JavaScript in HTML document asking user's name and then printing Hello <User_Name>
3. Create a dialogue box with "Welcome to my Website" message.
4. Evaluate the expression:
 - a. $7+5$
 - b. $"7" + "5"$
 - c. $7 * 5$
 - d. $7/5$
 - e. $7 \% 5$
5. Write the segment of Script that would ask the user if he wants a greeting message and if he does, display a Gif file called Welcome .gif and display "Welcome to Netscape Navigator!" in the document window following the Gif

6. Write the object definition for an object called car with four properties model, make, year & price.

SESSION 2

Exercises

1. Write a program to display a multiplication table.
2. Write a code to create a scrolling text in a text box.
3. Write a JavaScript code to create a pull down menu box.
4. Write a program to move a text with mouse pointer.
5. Write a program to change colour of text randomly.
6. Create a Web page using two image files, which switch b/w one another as the mouse pointer moves over the image. Use the On Mouse over and On Mouse out event handler

SESSION 3

Exercises

1. Write a JavaScript code to accept radius & display the area of the circle.
2. Use the date function get Date & set Date to prompt the user for an integer b/w 1 – 31 & return day of the week it represents.
3. Display time and print message accordingly e.g., ‘Good Morning’ in Morning etc.
4. Using JavaScript create a digital clock.
5. Using JavaScript create a calculator.
6. Create an HTML form that has a number of text boxes. The user fills the textboxes with data. Write a script that verifies that all textboxes have been filled. If a text box has been left empty pop up an alert message indicating the box that has been left empty. When OK button is clicked, set focus to that specific textbox. If all the textboxes are filled, display thank you.

SESSION 4

Exercises

1. Create an HTML form that inputs employee details and display the same on the HTML page.

2. Write a program, which prompts the user to enter a sum of two numbers and display whether the answer is correct or incorrect.
3. Illustrate how the reset button on form functions.
4. Create a program to check for null or empty string.
5. Create a program to generate a hit counter.
6. Create a program to verify whether email address provided by the user is valid or invalid.

SESSION 5

Exercises

1. Write a program to scroll the text on status bar.
2. Write a program to create a small window in main window.
3. The form consists of two multiple choice lists and one single choice list
 - a. The first multiple choice list displays the major dishes available.
 - b. The second multiple choice list displays the stocks available.
 - c. The single choice list displays the miscellaneous (Milkshakes, soft drinks, softy etc. available)
4. Create a Web page with two forms, one office copy and one customer copy when user enters date in customer copy it gets updated in office copy.
5. Use JavaScript for authentication and verification of the textboxes in the static site developed by the student in the HTML exercise.

3.12 FURTHER READINGS

1. JavaScript Programmers Reference, Cliff Wootton. Publisher: Wrox Press Inc. Year 1999.
2. Beginning JavaScript, Paul Wilton. Publisher: Wrox Press Inc. 1st Edition
3. JavaScript: The Definitive Guide, David Flanagan. Publisher: O'Reilly. 4th Edition 2001.
4. The JavaScript Bible, Danny Goodman. Publisher: John Wiley & Sons Inc. Edition 2001.

SECTION 4 WEB PRORAMMING LAB

| Structure | Page Nos. |
|---|-----------|
| 4.0 Introduction | 128 |
| 4.1 Objectives | 128 |
| 4.2 Development of a simple website | 129 |
| 4.3 Using JavaScript in Netbeans | 136 |
| 4.4 Creating and validating XML pages | 137 |
| 4.5 Running JSP Programs | 140 |
| 4.6 Creating Database Applications | 144 |
| 4.6.1 Creating Database and Database connections using Netbeans | 145 |
| 4.6.2 Creating Form and Connection using JSTL | 148 |
| 4.6.3 Storing Student information in the Database | 155 |
| 4.6.4 Comments on the Website | 159 |
| 4.7 List of Lab Assignments | 160 |
| 4.8 Further Readings | 162 |

4.0 INTRODUCTION

This lab course provides you information about the tool that you need to use for practical of Web programming course. This lab course supports the course BCS053: Web Programming. We propose that you use an Integrated Development Environment (IDE) such as Netbeans, Eclipse or any other IDE for web development. You are advised to install the latest version of these IDEs.

You may need a web server to display the web pagesthat you may create. Fortunately, the present day IDEs also include a web server.For example Netbeansis bundled with web servers – Apache and Glassfish. For the purpose of back end database we will demonstrate the use of MySql. However, you may use any database technology at your study centre. In such case you have to use the necessary drivers.

In this lab manual, we have used Netbeans IDE 7.3.1. We first discuss about the process of installation of Netbeans along with the web server and database system. We then explain creating HTML, XML, Web Application, JSP pages etc using it.

4.1 OBJECTIVES

After completing this lab section, you should be able to:

- Install any IDE for web programming;
- Create web pages using several technologies;
- Store data in a table using web pages
- Display web pages on a web server.

4.2 DEVELOPMENT OF A SIMPLE WEBSITE

Netbeans is an IDE which provided features for developing desktop, mobile and web applications. It supports Java, HTML5 and other languages. It is an open source software which is freely available for download. it has its own web site <https://netbeans.org/> . You can download it from the download link at this web site. You may download the bundle that includes all the tools including GlassFish Servers Open source edition and Apache's Tomcat Server. Please also note that a number of tutorials have been created by Netbeans community. These tutors are available on the web site <https://netbeans.org/kb/> . You must take a video tour of key netbeans features as an IDE.

Once you have downloaded the Netbeans IDE, you can install it on the computer using the following procedure:

Run the binary installation file

In case your JDK version is older than the supported by Netbeans, it will inform you to do so. You can obtain the latest JDK from the website: <http://www.oracle.com/technetwork/java/javase/downloads/>. You may download the Java version for windows x86, in case you are using a 32 bit computer. You should install it first and then install Netbeans.

Make sure that during this installation web servers are also installed.

Now, start Netbeans, it will take some time to start and the following window will open:

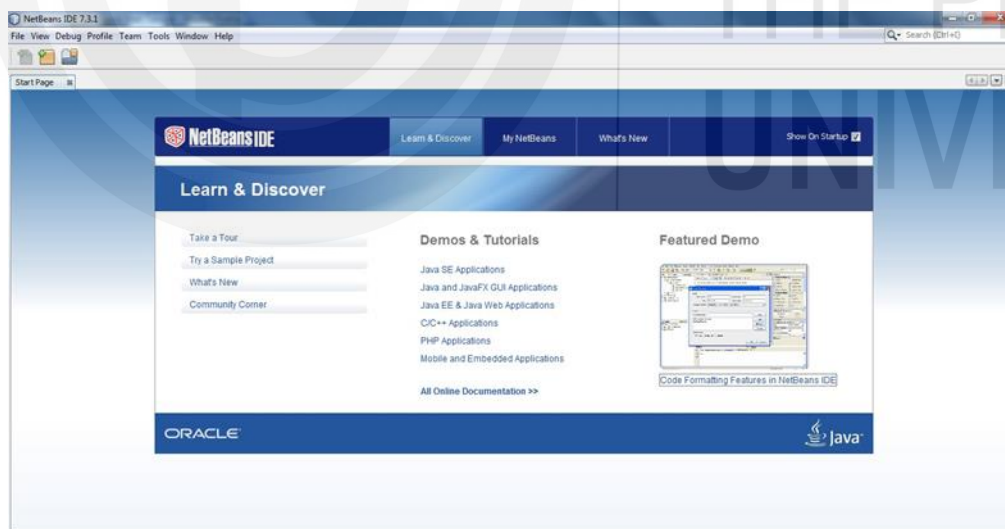


Figure 4.1: Start of Netbean Software

You can select the Demos and Tutorials or Featured Demo to see a demonstration of Netbeansprovided, your computer is connected to Internet, as shown in Figure 4.1.

Once you are a bit familiar with the environment, start a new project. Perform the following steps:

Select File→New (CTRL+SHIFT+N) Project from the Menu of Figure 4.2.

The following window will open:

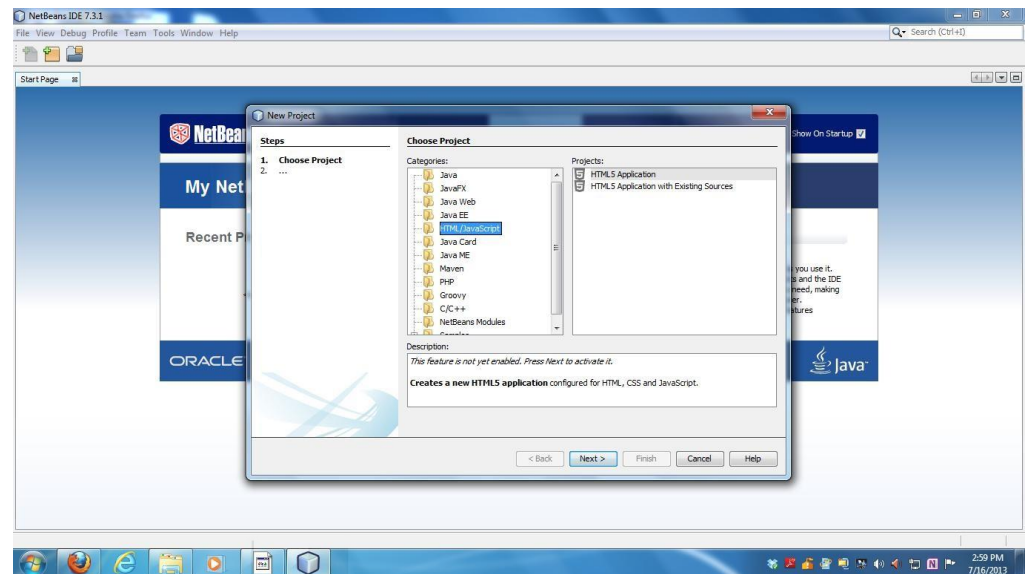


Figure 4.2: New Project Window

In this window you can select a category for the new project, for example, when you will design JSP project, you will select Java Web category. To begin with, you may open a new HTML 5 project. So select HTML5/JavaScript category and in the option HTML5 Application. Select the Next button.

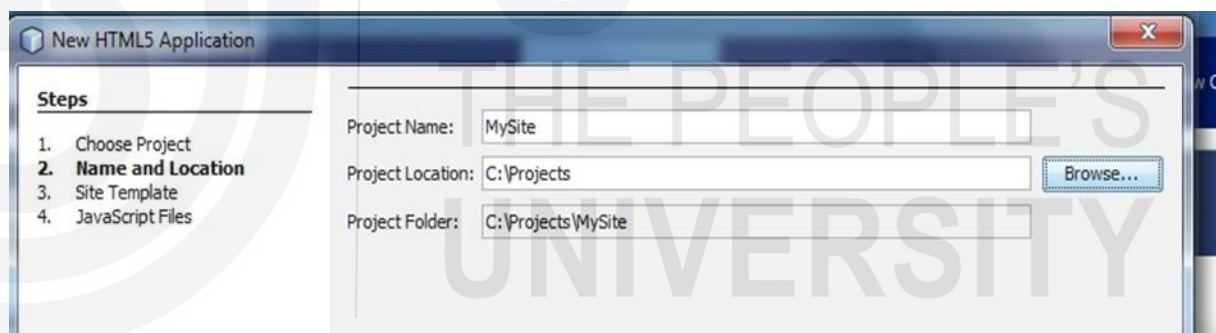
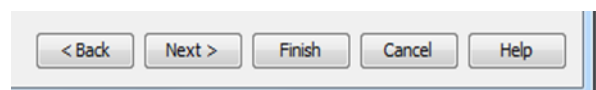


Figure 4.3: Selection of Name and Location of the Project

Enter project name and project location of your choice. You can observe that in the Figure 4.3 that we have given the project name as MySite and selected a Folder C:\Projects for its location. It is advisable to create a separate folder where you can keep all your projects. Now Select Finish from the button Panel.



You have successfully created the project. A file index.html will be created for you for editing in the Netbeans editor as show in Figure 4.4. Please note index.html is the default page of a website.

Now, you can create your home page of the website. For more details on various HTML tags and CSS, you may refer to Block 1 Unit 1 and Unit 2 of BCS053: Web Programming course. For the purpose of demonstration, we

have used the example given in Block 1 Unit 2 of BCS053. The example uses div tags for various sections of the web page. The example uses the CSS, but initially, we have removed the linkage to CSS and just created the web page. You can format this web page using the Menu Source → Format.

You need to input all the tags along with contents from the example. Figure 4.4 shows the contents of index.html after the html tags and contents of web page has been input. Please note that there is no CSS attached to this file even though it contains class names (please refer to BCS053 Block 1 Unit 2) in the tags.

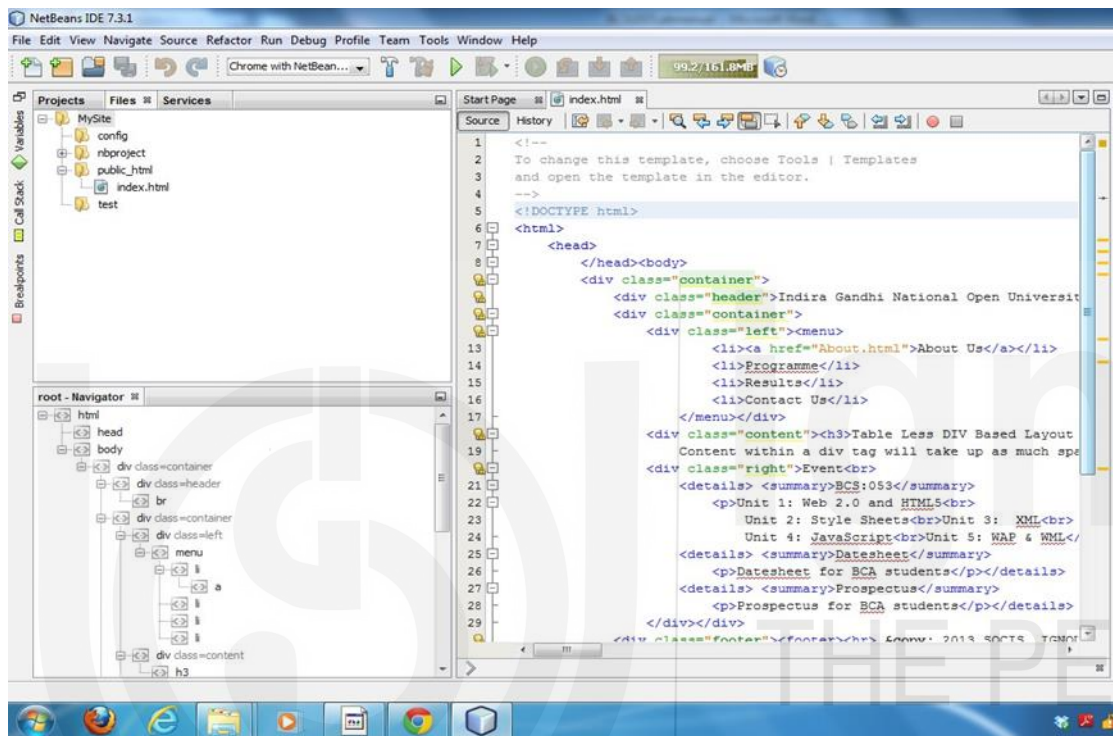


Figure 4.4: An example file

But, how to display the content of this webpage?

To see the web site that you have created, you can select Run→Run Project menu option or alternatively press F6 key. Please note that in case you are using Mozilla or Chrome browsers then you need to install “Netbeans connector chrome extension” which is available from the website <https://chrome.google.com/webstore/>. Please also note that when you will run the project nothing much will happen in Netbeans window, but the display can be seen in the browser window (chrome in this lab manual). Figure 4.5 shows the output of the web page of Figure 4.4.

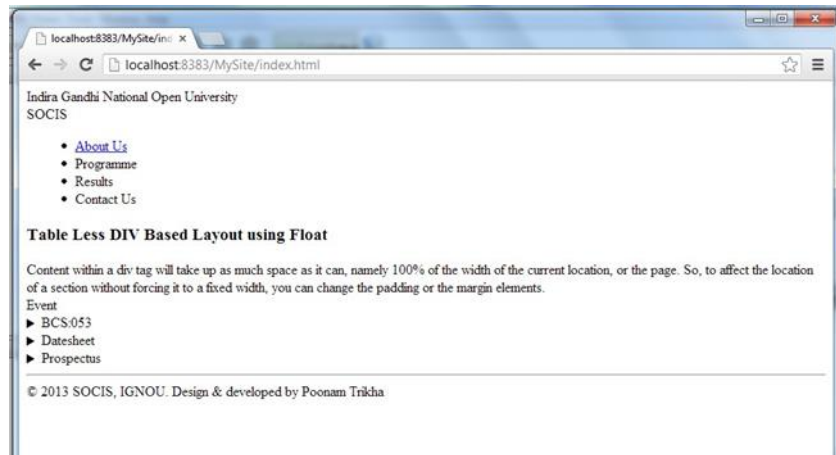
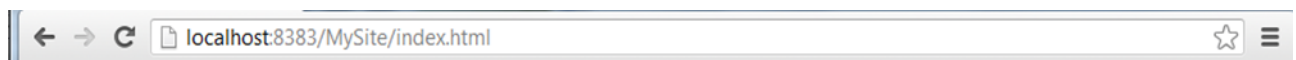


Figure 4.5: The display of web page of figure 4.4 in browser

You may please notice that the address bar of Figure 4.5 shows:



which means that the web page displayed is not displayed from a file but is a part of website hosted at localhost at 8383 port.

The display of web page in Figure 4.5 is in the sequence of appearance of different tags of html. You can change this display to a good structured display using CSS. To create a CSS file, as given in the example of BCS053 Block 1 Unit 2 named CSSLayout.css, you need to perform the following:

From the File Menu of Netbeans select New File ...a New File Window will open

In the New File window in the Category HTML/JavaScript select the file type as Cascading Style Sheet and press the Next button in the button panel. A window containing title "New Cascading Sheet" will open

In this new window, enter the name of the stylesheet- CSSLayoutin our case and press the Finish button in the button panel.

A new file will open in the Netbeans window with the name CSSLayout.css. In this file type the commands of the CSS file. Figure 4.6 shows the resultant state.

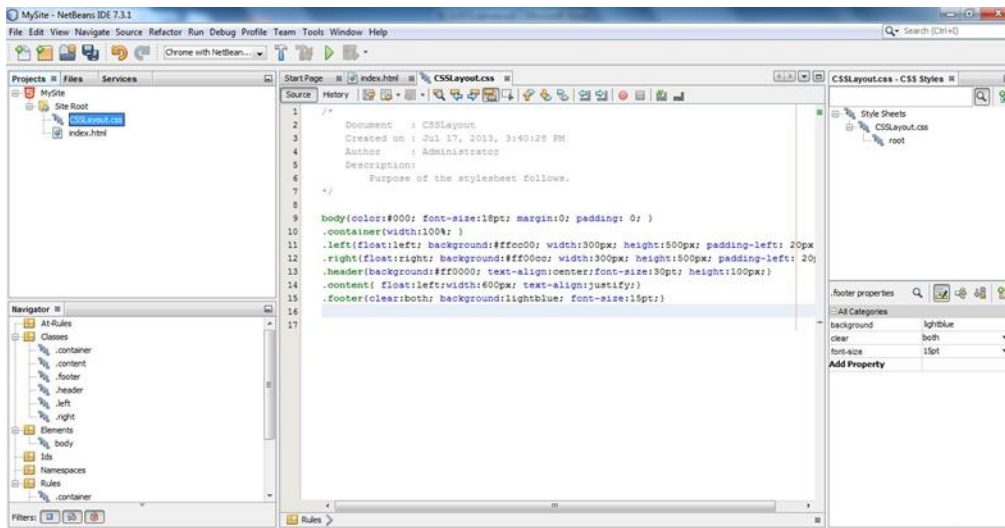


Figure 4.6: The CSSLayout.css file with contents.

To link this CSSLayout.css file to index.html file, you perform the following steps:

Select the tab “index.html” and add the following line in the `<head></head>` tags: `<link type="text/css" rel="stylesheet" href="CSSLayout.css" />`

Now, your index.html file is linked to the CSSLayout.css file. You can see how it changes the display of your webpage by pressing F6 key. Please observe how different the display is now by comparing the display of Figure 4.5 and Figure 4.7.

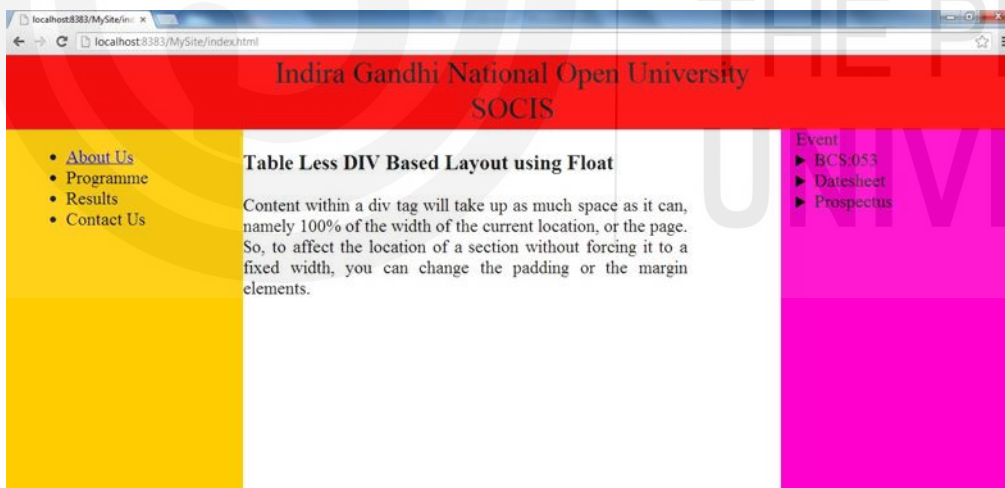


Figure 4.7: Display of web page of Figure 4.4 using CSS

Observe that in the event section only summary information is displayed. You may click on BCS053 and detailed information on it will be shown. In the left you see a simple selection options like About us, Programme etc. For every such option, you may have to develop a page. We demonstrate how you can implement *About Us* option. This time you can follow different strategy using Netbeans. In this strategy, we demonstrate the use of Palette Window. We may keep some of the sections on the website identical to the one as displayed in

Figure 4.7, but only changing therelevant information. The figure 4.8 shows the web browser display on pressing About Us option on Figure 4.7.

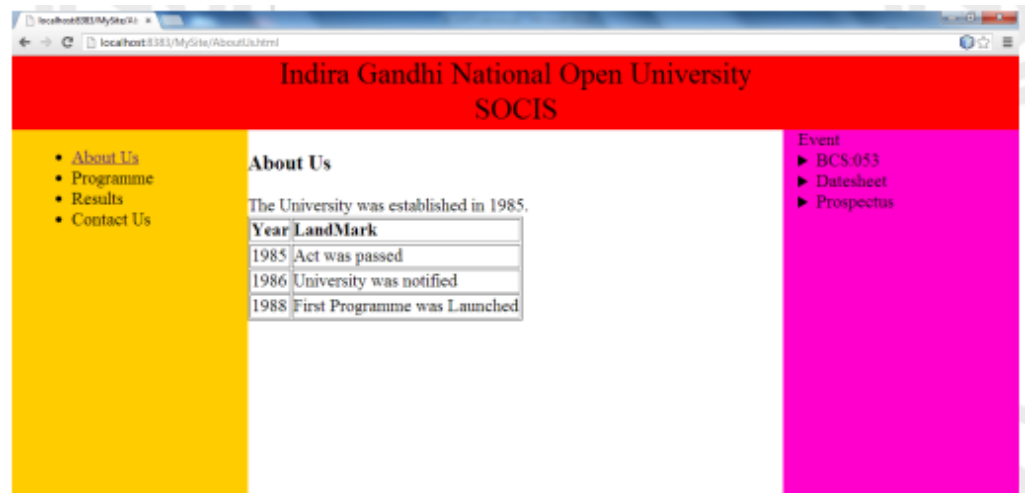


Figure 4.8: AboutUs page of website.

In order to develop this page you can perform the following steps:

From the File Menu of Netbeans select New File ...a New File Window will open.

In the New File window in the Category HTML/JavaScript select the file type as HTML File and press the Next button in the button panel. A window containing title “New HTML File” will open.

In this new window, enter the name of the file – AboutUS and press the Finish button in the button panel.

A new file with the name AboutUs.html will open in editor window. You copy the entire content of index.html file and replace all the contents of AboutUs.html by it.

Now change

```
<div class="content"><h3>Table Less DIV Based Layout using Float</h3>
Content within a div tag will take up as much ... ..elements. </div>
```

to

```
<div class="content"><h3>About Us</h3><b>The University was established
in 1985</b></div>
```

However, you need to add a table of various landmarks, perform the following steps:

From the Window Menu of Netbeans select Palettea New Palette Window will open along with other windows.

Click the mouse just after 1985 but before the </div> tag and in the Palette window double click on Table.

An Insert Table window will open, in which select the number of rows to 3 and press Ok button.

All the opening and closing tags for table headers and table rows will be created automatically, you just insert the appropriate values between an opening and closing tags.

Figure 4.9 shows all the changes made using the steps given above.

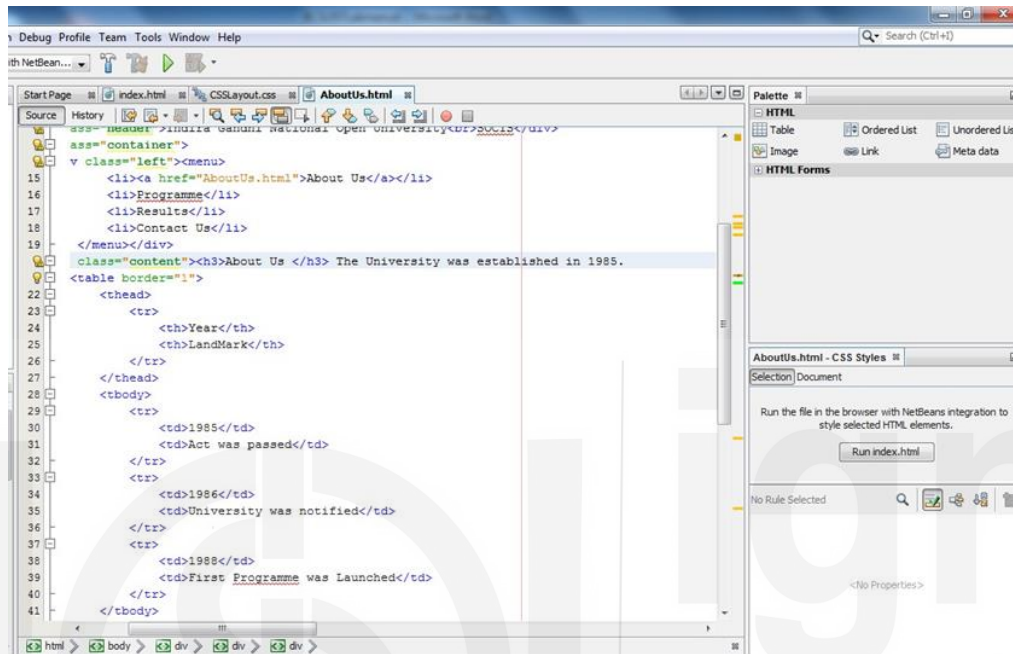


Figure 4.9: Use of Palette to enter HTML tags.

But before you display this web page, just make one more change both in index.html and AboutUs.html by changing the content

```
<li><a href="About.html">About Us</a></li>html  
to  
<li><a href="AboutUs.html">About Us</a></li>html
```

You can now press F6 key and browser will display contents as shown in Figure 4.9. Please note the following weaknesses in the webpages we have designed:

All the html and css files are in the same folder. In actual websites have large number of pages and cssfiles, therefore, are organized into folders setting the paths suitably through href or configuration files. You must find more about web site design from further readings.

The second weakness is that since there is redundancy of content in the two html pages, any change in common content will be difficult to maintain. This problem will be further compounded if the number of such pages is large. One of the solutions to this problem is to move common contents to a different file or files and include those files in all the files through programming language or configuration files.\

A detailed discussion on these issues are beyond the scope of this Section, but we have introduced these aspects just to remind you that you have to keep exploring about the web programming beyond the completion of this course.

Another weakness of the website developed so far is that it is static and does not involve significant interaction with the user. In the next section, we demonstrate how to add some dynamism in the website with the help of JavaScript.

4.3 USING JAVASCRIPT IN NETBEANS

Netbeans allows you to use javascript in two ways, either you can create JavaScript file and use that file in a script tag to run or you can add your own script tags in head or the body tags. We will put the JavaScript in the head tag. The event related action will be put on the body part. We demonstrate the function for change of background colour given in BCS053: Block 1 Unit4. The function changes the background colour, if you mouse over a space and changes it back to original when mouse moves out of that pace. The code to do so is reproduced below from the said Unit (with few changes)

```
<!DOCTYPE html>
<html>
<head>
<script> functionchangebackgroundcolour(bg)
{
document.body.style.background = bg;
}
</script>
</head>
<body>
<p>Mouse over the squares and the background color will change!</p>
<table width="200">
<tr>
<td onmouseover="bgChange('lightblue')"
onmouseout="bgChange('transparent')"
bgcolor="lightblue"> Light Blue
</td>
<td onmouseover="bgChange('lightgreen')"
onmouseout="bgChange('transparent')"
bgcolor="lightgreen">Light Green
</td></tr>
</table>
</body>
</html>
```

Figure 4.10: JavaScript code for change of background colour

But, how to use this code in our existing webpages? We demonstrate it using AboutUs.html. You put the code shown in bold above in the head tag after the link tag and add the code shown in italics is added in the division having the class “content” after the table showing year and landmark. Now, you run the

project using F6 and select About Us link in the browser. The browser display is shown in Figure 4.11. You can take your mouse over the two colour boxes with titles light blue and light green to see changes in the background colour. Figure 4.11 is captured when mouse pointer was over the box light green.



Figure 4.11: Browser display demonstrating use of JavaScript code of Figure 4.10

The event based programming along with JavaScript and dynamic features of html can be used to develop simple animations, interactive pages, form verification etc. You should search for such examples through the WWW for further learning on these. In the subsequent section we discuss about creating XML pages and also how to create a simple web page for mobile devices.

4.4 CREATING AND VALIDATING XML PAGES

So far you have created HTML5 page and JavaScript code using Netbeans. Can you create XML pages in Netbeans? Can you check these XML pages? This section shows you an example of XML page along with the related DTD. It also shows you the process of checking the XML file using DTD. You may also create XML schema and XSL stylesheet using Netbeans. However, this is left as an exercise. We first create a DTD file for customer with the following statements:

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

Figure 4.12: The DTD file content

This DTD can check the following XML document created by you:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE customer SYSTEM "Customer.dtd">
<customer>
  <name3>Arvind Gupta</name3>
```

```
<address>IGNOU New Delhi</Address>
</customer>
```

Figure 4.13: The XML file content with errors

Please note there are two mistakes in the XML as above. First the tag `<name3>` is declared as `<name>` and the tag `<address>` is closed by tag `</Address>` which is incorrect as the A is capital. Let us see if these errors are flagged by the Netbeans XML validation. But first you need to create the two files. You can first create a New project with the name XML project. In this project now create a new DTD file using the steps:

Select the option File → New File ...

In the New File window in the categories section select XML. You will see the following window:

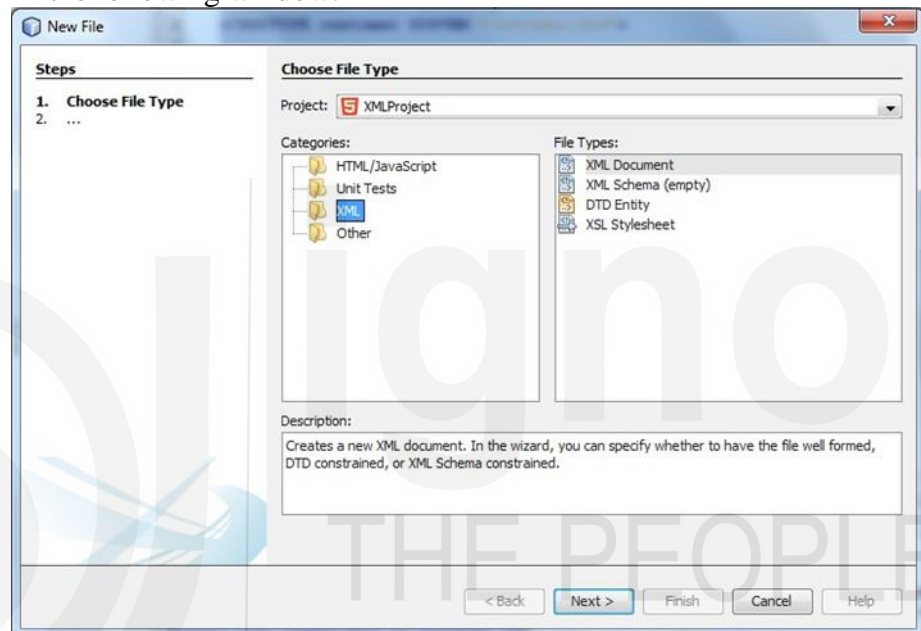


Figure 4.14: The XML Category files

In the File Types section select DTD entity and press the Next button in the button panel at the bottom of this window.

Name the File as Customer and press the Finish Button in the button panel

A new DTD file with the name Customer.dtd will open. Type the contents as shown in Figure 4.12 in the file.

Now, you have created the DTD file. Next create the XML file using the following steps:

Select the option File → New File ...

In the New File window in the categories section select XML and in the File Types section select XML Document and name it SingleCustomer

In the file so created namely SingleCustome.xml type in the content as shown in Figure 4.14.

Now, your documents are ready for validation. Open the SingleCustomer.xml file in the editing window and select Run → Validate XML

It will open the following window:

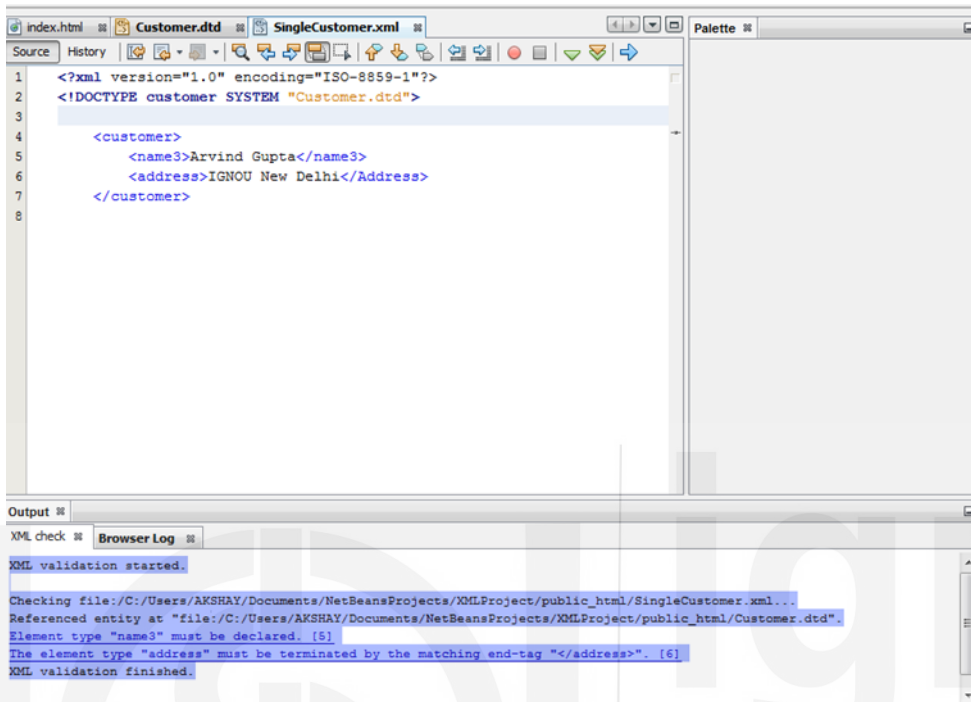


Figure 4.15: The highlighted output window showing errors

The window shows the following messages and errors as shown in Figure 4.15.

XML validation started.

Checking

file:/C:/Users/AKSHAY/Documents/NetBeansProjects/XMLProject/public_html/SingleCustomer.xml...

Referenced entity at

"file:/C:/Users/AKSHAY/Documents/NetBeansProjects/XMLProject/public_html/SingleCustomer.xml".

Element type "name3" must be declared. [5]

The element type "address" must be terminated by the matching end-tag "</address>". [6]

XML validation finished.

The messages clearly identify the errors. You just need to correct the errors by changing tag <name3> to <name>, </name3> to </name> and </Address> to </address>. After making these changes, you may run the validation again. This time no error will be reported.

4.5 RUNNING JSP PROGRAMS

Java Server Pages (JSP) programs are very useful in performing server based interactive tasks. In this section, you will be demonstrated a simple example on how you can use Netbeans to create simple website using JSP.

You have already seen an example of HTML and CSS project in section 1.2, we will modify that project to create a basic web page structure and add two important menu items – Display of current date using a file DisplayDate.jsp and Display of Visitor Number using DisplayCounter.jsp that uses a java class file VisitorCounter.java. In addition, we create two jsp files top and bottom which will be included in both the display files. The project will show the output as given in Figure 4.16.



Figure 4.16: Display of web page in the Browser

Please note that display is from a web server (check address line which reads localhost:8080/...). On the left, you see three items namely Home, Today's Date and Your Visitor No. You can click any one, the page has been created such that it only changes the middle section, for example, if you click on the option "Your Visitor No" the display line "**Please watch change in contents of this division only**" will change to "**Your visitor number is : 1**". If you click this option again, the number will shown be 2 and so on. But, how can you create this page. To create this simple web site, we have created a simple web site consisting of the following files:

File Name : index.html

```
<!DOCTYPE html>
<html>
<head>
  <link type="text/css" rel="stylesheet" href="CSSLayout.css" />
</head>
<body>
  <div class="container">
    <div class="header">
      Indira Gandhi National Open University
    </div>
    <div class="left">
      <menu>
```



```

    <li><a href="index.html">Home</a></li>
    <li><a href="DisplayDate.jsp">Today's Date</a></li>
    <li><a href="DisplayCounter.jsp">Your Visitor No.</a></li>
</menu>
</div>
<div class="content">
    <h3>Welcome to Lab Sessions</h3>

```

Please watch change in contents of this division only

```

</div>
<div class="right"> Course<br>
    <details>
    <summary>BCS:053 Block 1
    </summary>
    <p>Unit 1: Web 2.0 and HTML5<br> Unit 2: Style Sheets<br>
    Unit 3: XML<br>
    Unit 4: JavaScript<br> Unit 5: WAP & WML
    </p>
    </details>
</div>
</div>
<div class="footer">
<footer>
    <hr>&copy; 2013 SOCIS, IGNOU.
</footer>
</div>
</body>
</html>

```

The index.html consists of three parts shown in three boxes. The content in the first and third boxes are common to both the DisplayDate.jsp and DisplayCounter.jsp files, as you want consistent background and sections in the web site. Is it a good idea to copy and paste the content of first and third box in both the files? Obvious answer is no as this will create unnecessary maintenance related problems. Therefore, you can copy the content of first box and third boxes into two different files. We have named these files as top.jsp and bottom.jsp files. You can include these files in the DisplayDate.jsp and DisplayCounter.jsp files. The content of these two files are shown below. But how to create a new jsp file? You can create a file using the following steps:

Create a new file using File → New File ... option. A New File window will open.

In the New File Window in the Categories section select Web and select JSP in the File types section and press Next button, a New JSP window will open type in the name of the file and press Finish button in this window.

The new jsp file will opened for editing.

File Name : DisplayDate.jsp

```
1.    <% @page contentType="text/html" pageEncoding="UTF-8"% >
2.    <% @include file= "top.jsp"% >
3.    Todays Date is :
4.    <% //Scriptlet for initialising and printing date object%>
5.    <%      java.util.Date date = new java.util.Date();
6.    out.println(" ");
7.    out.println(date);
8.    %>
9.    <% @include file= "bottom.jsp"% >
```

Please note that there are two include commands in the jsp file and scriptlet code for printing the date object. If you want to make any changes in the menu options, or heading, footer etc., you just need to do so in the top.jsp or bottom.jsp. Please note that line 1 describes the content type for this web page. Line 2 and 9 contains the jsp script to include the top.jsp and bottom.jsp files respectively. Line 3 just displays HTML content. Line 4 is a comment line in jsp. Line 5 initialises a object date of class. *Date* with the current date and time. Line 6 output a blank line and line 7 prints the date object.

File Name : VisitorCounter.java

```
package JavaPackage;
public class VisitorCounter
{
    private static int ctr;
    public static int readCounterValue()
    {
        ctr++;
        return ctr;
    }
}
```

This file creates a java class file namely *VisitorCounter*. This class file is used by the server to store a static variable *ctr*. The variable *ctr* is incremented every time the *readCounterValue()* method of the class is called. This class is used by *DisplayCounter.jsp* file to keep track of Visitors who click on Visitor No's link. Please note that the first line of this file is *package JavaPackage*. You need to create this file using the following commands:

File → New File ... A New File window will open.

Select **Java** in the categories section and **Java Class** in the File Types in the New File window and press the Next button. A New Java Class window will open.

In the New Java Class window, type Class Name - *VisitorCounter* and Package *JavaPackage* as shown in Figure 4.17 and press the Finish button in the button panel.

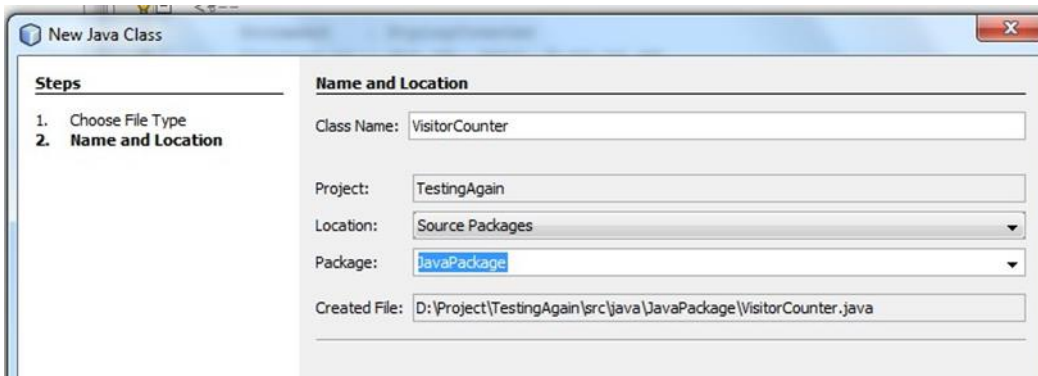


Figure 4.17: Creating the VisitorCounter class in the package JavaPackage

A new class file with the name VisitorCounter.java will open in the Netbeans editing window. The file will include the following lines:

```
package JavaPackage;
public class VisitorCounter {

}
```

Type the remaining part of the file. Please note this file will be created in folder src— java—javapackage folder as shown in the Figure 4.18.

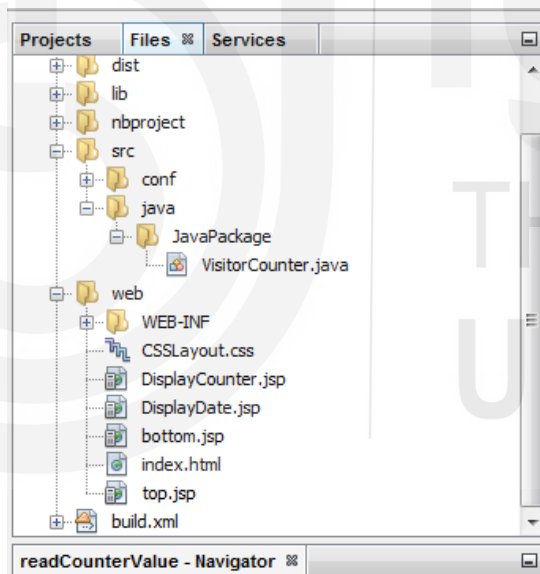


Figure 4.18: The Files Tab showing all the files

Now, you can create the new jsp file DisplayCounter.jsp

File Name : DisplayCounter.jsp

1. `<% @page import="JavaPackage.VisitorCounter"%>`
2. `<% @page contentType="text/html" pageEncoding="UTF-8"%>`
3. `<% @include file= "top.jsp"%>`
4. Your visitor number is :
5. `<% out.println(VisitorCounter.readCounterValue());`
6. `%>`
7. `% @include file= "bottom.jsp"%`

You need to tell the class name and location of VisitorCounter.java file, line 1 does this task. Line 3 and 7 are for including files. Line 6 calls the method readCounterValue() of VisitorCounter class to display the visitor number.

Once you have created all the files, you can run this project using Run→Run Project option on the Netbeans menu. It will open the browser window as shown in Figure 4.16. You can click on the three options on the left section and observe the changes in the web pages.

You must be wondering why we have typed the complete content in the index.html? Remember index.html is an html file and NOT jsp file. Therefore, you can not include files in it. If you want to do so, you need to create an index.jsp.

Is the method of including files in jsp files the only method of creating maintainable web sites? No, you can create fragment files having jspf extension and use configuration file which determines in which files these jspfs are to be included.

However, these are beyond the scope of this section and you should go through Netbeans tutorials for more details on these options.

It is a good idea to create a number of folders for a website. For example, images can be put in the image folder, similarly main pages can be put together in a folder, fragments can be put in separate folder. Once again, you are advised to go through Netbeans tutorial especially Netbeans E-commerce tutorial at the web address: <https://netbeans.org/kb/docs/javaee/ecommerce/intro.html> for more details.

4.6 CREATING DATABASE APPLICATION

So far you have used the Netbeans environment for creating simple JSP programs. However, a good website may need to connect to a backend data server. Some such tasks may involve entering information, checking for login etc. In this section, we will explain, how a database based application can be created using netbeans. For this section, you will also require MySQL database. (You may any database instead of MySQL such as PostgreSQL also). You can download MySQL/PostgreSQL database from their respective web site. You must install the file and create the password for the root.

First let us define the problem that you need to solve in this section. You need to create a Student Information form that stores the information of a student in a database. You are also required to query the database to retrieve programme wise student information. This application requires interfaces and database support. As a first step, you need to analyse and design the problem, however, we have shown here a very simple implementation (without error checks) using JSP. The following sub- sections illustrate this example.

4.6.1 Creating Database and Database connections using netbeans

Once you have successfully installed MySQL client, it will be ready to use. You can connect to it from the Netbeans, using the following:

Open the Netbeans and select the Services tab on the Project pane and click on Drivers. You will see the following in your window (as shown in Figure 4.19).

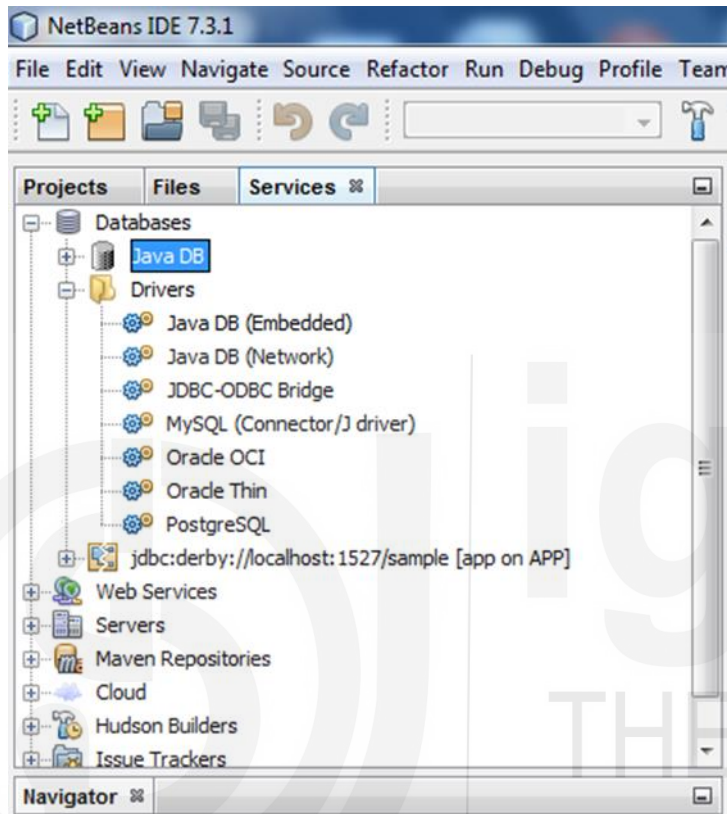


Figure 4.19: Display of database drivers

The Services tag is highlighted and under the Drivers folder, you can see MySQL (Connector/J driver). (if you are using PostgreSQL then you can use the PostgreSQL driver listed in this window). Right click on the Driver you want to use and select option “Connect Using ...”, the following window will open:

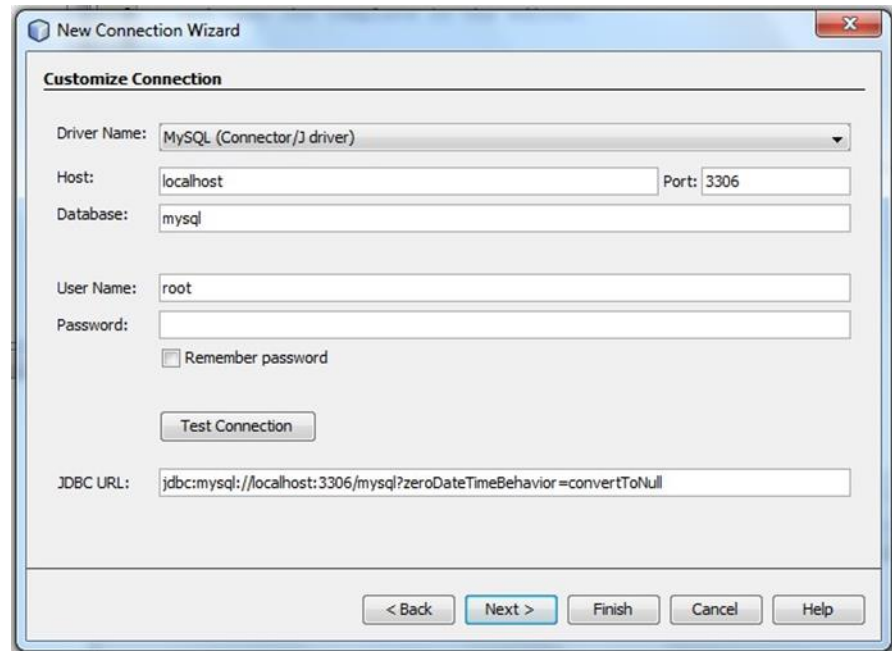


Figure 4.20: New Database Connection using MySQL

Please note that in Figure 4.20, user name is already specified as root. You can type in the password in the Password text box, you can also check the Remember password check box, if you want so. Test the connection by pressing the Button “Test Connection”. In case your connection is ok, you will see a message “Connection Succeeded”. Press Finish once you have completed the tasks.

A new MySQL connection with the name will be displayed below the driver folder.

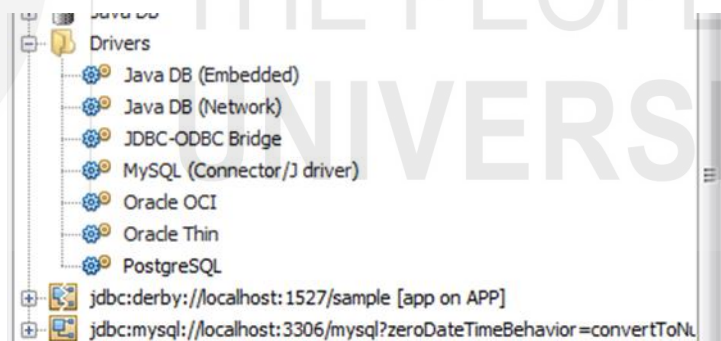


Figure 4.21: The database connection to default database named mysql

Now, you can run the following SQL commands to create Student database and a new database user named netbean having the password netbean7. Please note that GRANT command of SQL gives user netbean all types of access over the database student.

```
create database Student;
create user 'netbean'@'localhost' identified by 'netbean7';
grant all on student.* to 'netbean'@'localhost';
```

But how can you run these commands using netbeans?

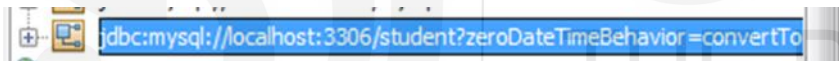
You can Right click on the MySQL connection as shown in Figure 4.21 and select the option “Execute Command ...”, a tab with the name “SQL Command 1” will open in the editor window. Type all the SQL commands in that window and press CTRL+SHIFT+E keys to run these SQL commands.

Once you have created the database student, you can create tables in this database. But for this you may have to create a new connection to the student database. You may be wondering why do you need to create this new connection? This will make sure that you work with the students database only. In addition, we also propose that you use different user name and password to for this connection. Why, well the main reason is that the root user has access to every database resource and giving such permission to everyone will be dangerous. Therefore, it may be a good idea to give limited access to user accounts. Please note that netbean account only have access to student database only. It can perform all the database creation and manipulation commands (specified by ALL in the grant command) on the database student only.

So, create another MySQL database connection following the same procedure as given earlier, but enter the following in Figure 4.20.

Datbase: student
User Name: netbean
Password: netbean7

The following new connection will be created:



Now, you can execute the following SQL commands using the new database connection that you have created.

You should execute the following commands to create two tables programme and studentmaster. Please note that in the command Primary key, check constraint and Foreign key constraint has been specified.

```
create table programme (  
  ProgCode char(5) PRIMARY KEY,  
  Pname varchar(40) NOT NULL,  
  NoOfSemester int NOT NULL,  
  check (NoOfSemester > 0 and NoOfSemester < 15)  
);
```

```
create table studentmaster ( stID  
  char(9) PRIMARY KEY,  
  stname varchar(25) NOT NULL,  
  stphone char(12) NOT NULL,  
  ProgCode char(10) NOT NULL,  
  FOREIGN KEY (ProgCode) references programme(ProgCode) on delete  
  restrict on update cascade);
```


You can now insert some data in the programme table using the following commands.

```
insert into programme VALUES ("MCA","Master of Computer Applications",6);
insert into programme VALUES ("BCA","Bachelor of Computer Applications",6);
insert into programme VALUES ("CIT","Certificate in Information Technology",1);
insert into programme VALUES ("MBA","Master of Business Administration",6);
```

4.6.2 Creating Form and Connection using JSTL

Once the database is ready to use, you need to create a JSP form to input student data in the studentmaster table. The form that has been created in this application is shown in the following figure.



Figure 4.22: Form for Entering Student Information

But how can you create this form? Please note that this is the same web site that you had created in the previous sections. However, you have to add two more options in this web site – Enter Student Information and Student List for a Course. First to add these options, you need to make changes in the index.html and top.jsp files. In both these files add the following two lines as the last two lines of the <menu> tag:

```
<li><a href="StudentInformationForm.jsp">Enter Student  
Information</a></li>  
<li><a href="CourseStudentList.jsp">Student List for a Course</a></li>
```

Now, you need to create a form - StudentInformationForm.jsp. In your project create a new jsp file and name it StudentInformationForm. This form has been created using the Palette window of the netbeans and JSTL (JSP Standard Tag Library). Please note that data entry options (Label and Information) has been created using table. The Select Programme is a drop down list. First let us look into the jsp code of the form.


```

1      <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
2      <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
3      <sql:setDataSource var="student" driver="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost/student" user="netbean"
        password="netbean7"/>
4      <sql:query dataSource="${student}" var="ProgData">
5      SELECT ProgCode, Pname from programme;
6      </sql:query>

7      <%@page contentType="text/html" pageEncoding="UTF-8"%>

8      <%@include file="top.jsp"%>
9      <form name="InputStudentData"
        action="StudentDataInput.jsp" method="POST">
10     <table border="0">
11     <thead>

12     <tr>
        <th>Label</th>
        <th>Information</th>

13     </tr>
14     </thead>
15     <tbody>
16     <tr>
        <td>Student ID :</td>
        <td><input type="text" name="StudentID" value="" size="9"
          /></td>

17     </tr>

18     <td>Student Name:</td>
        <td><input type="text" name="StudentName" value=""
          size="25" />
        </td>

19     <td> Student Phone :</td>
        <td><input type="text" name="StudentPhone" value=""
          size="12" />
        </td>

20     <td> Select Programme :</td>
        <td>

21     <select name="ProgData">
        <c:forEach var="row" items="${ProgData.rows}">
22     <option value="<c:out value="${row.ProgCode}"/>">
        <c:out value="${row.Pname}"/>
        </option>
        </c:forEach>
      </select>
    </td>

```

```

23      /tr>

24      </tbody>
25  </table>
26  <input type="submit" value="Submit Information" name="Submit" />
27  </form>
28  <% @include file= "bottom.jsp"%>

```

Let us discuss about the code in more details:

Lines 1 and 2 are typical defining the tag libraries for JSTL. Please note these lines are automatically added when you use Palette for inserting Database code using JSTL.

Line 3 contains code for connecting JSP code that is run on the web server to database server. The connection establishment as discussed in the previous sub-section connects netbeans to database. But when you are running a web application involving JSP, the JSP code is run at the web server (localhost:8080 is the address for the local Glassfish server that we are using in the Netbeans project). This server code is to be connected to the database server (which is also MySQL local host in this case) which has the database. The code at line 3 tells the server which source of data are you using. For the given code the driver to be used is the default SQL JDBC driver named "com.mysql.jdbc.Driver". This tag also informs the web server about the database name and location, user name and password (in our example, "jdbc:mysql://localhost/student", "netbean" and "netbean7" respectively). Please note that this database connection is attached with a variable name "student" in this line.

Lines 4-6 is the code for the query that is a standard SQL query. This query extracts the ProgCode and Pname from the Programme table. Please observe the data source given here is \${student} which refers to variable, i.e. student, that you have created in line 3. This statement also specified a new variable name ProgData which will be assigned the output of the query when it is run.

Line 7 defines the content type of this JSP page.

Line 8 and 28 includes the top.jsp and bottom.jsp files in the code.

Line 9 typically identifies the form and the file that will be executed when submit button is pressed. The method used to transfer parameter is POST in this case.

Line 11-25 describes various label and input fields such as text box. Line 26 creates the Submit Information button. Line number 22 needs some more discussion. The HTML code for the Drop Down List of Programme as can be seen in the page source of the browser is:

```

<select name="ProgCode">
    <option value="BCA"> Bachelor of Computer Applications</option>
    <option value="CIT"> Certificate in Information Technology</option>
    <option value="MBA"> Master of Business Administration</option>

```

```

        <option value="MCA"> Master of Computer Applications</option>
</select>
While the JSP code is:
<select name="ProgData">
    <c:forEach var="row" items="${ProgData.rows}">
        <option value="<c:out value="${row.ProgCode}"/>">
            <c:out value="${row.Pname}"/>
        </option>
    </c:forEach>
</select>

```

The first and last lines of both the codes are identical. The JSP code contains for each loop that processes all the output provided as a result of execution of query in lines 4-6. The query produces the result:

Prog Code	Pname
BCA	Bachelor of Computer Applications
CIT	Certificate in Information Technology
MBA	Master of Business Administration
MCA	Master of Computer Applications

<c:forEach var="row" items="\${ProgData.rows}"> is a JSTL tag. It selects rows of the query variable ProgCode one by one, for example, it will first extract the row of BCA.

<option value="<c:out value="\${row.ProgCode}"/>"> is an HTML tag which contains a JSTL tag <c:out/>. The HTML tag output the content:

```
<option value="
```

Next the JSTL tag <c:out value="\${row.ProgCode}"/> executes and produces the ProgCode (the SQL field name) of the first row. This happens to be BCA.

Thus, the output is:

```
    BCA
```

Finally the last portion "> produces the output ">.

Thus, this line produces the tag:

```
    <option value="BCA">
```

The line <c:out value="\${row.Pname}"/> for the first row simply produces the output:

```
        Bachelor of Computer Applications
```

The last line produces the output:

```
    </option>
```

So when you put together the output of the all these three rows together you get the HTML code:

```
    <option value="BCA"> Bachelor of Computer Applications</option>
```

Which is the first line of HTML code of the browser.

Please note that the JSP code is repeated for each row of the data output as shown in the table, thus, creating the complete HTML code of Select options.

But how do you create this file using Netbeans. Perform the following steps:

Create a new JSP file, give it a name *StudentInformationForm*. In the page so created by Netbeans, remove all the contents after the `<page ...>` tag and insert the two include tags including `top.jsp` and `bootom.jsp`. Between these two include insert a form using Palette (in case you cannot see palette window then press `CTRL+SHIFT+8`). From the HTML Forms portion of palette drag Form between the two include tags. The following insert form window will appear. Insert the file name that processes the form in this case it is `StudentDataInput.jsp`. Select method to POST and give it a name and click OK button. As shown in Figure 4.23.

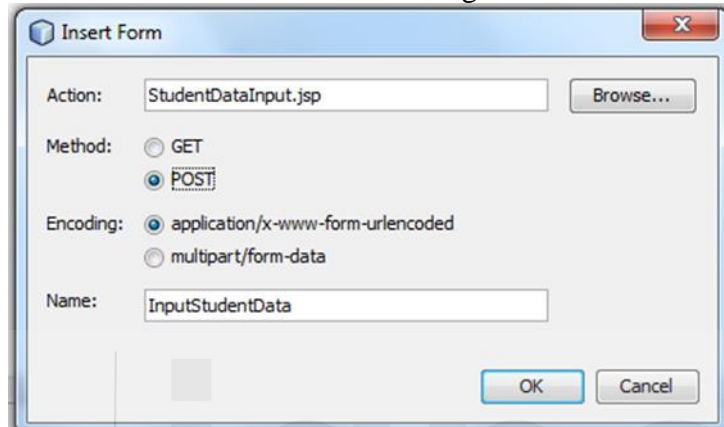


Figure 4.23: The Insert Form Window

The form opening and closing tags will appear between the two include tags. Now you can insert a table between the two form tags to create various input options (except the select options). You can also type in the code from the lines 10 to 26 except the line 22.

Now, you need to insert the HTML code given at line 22. Place your cursor between the code of line 21 and 23 and create type

```
<tr>
    <td> Select Programme :</td>
    <td>
        <select name="ProgCode">
            <option value="BCA">
                Bachelor of Computer Applications
            </option>
        </select>
    </td>
```

Please note the BCA and Bachelor of Computer Applications are to be replaced by JSP code which will make sure all the programmes are included in the drop down list. But in which database source programme data is available? As indicated earlier this data is available in MySQL database student in the programme table. You have already established a connection with the database along with user name and password. So enter all this information at line 1 of your JSP program. Enter the following information at line 1 of this JSP file.

```
<sql:setDataSource var="student" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/student"
```

```
user="netbean" password="netbean7"/>
```

Once you have informed about the connection and database, you will have to write a query that will list various programme codes and programme names. The query in SQL for this purpose is:

```
SELECT ProgCode, Pname from programme;
```

Using this data a programme name drop down list is to be created, but for this list the data that is being passed through POST method will be the ProgCode such as BCA, MCA only. You need to write JSP code that performs this task. The data for this purpose will be retrieved as a collection of rows (one row for each course) and columns (two columns ProgCode and Pname).

Put your cursor after the select tag and from the Database section of the Palette select and drag DBReport. The following window will open (As shown in Figure 4.25).

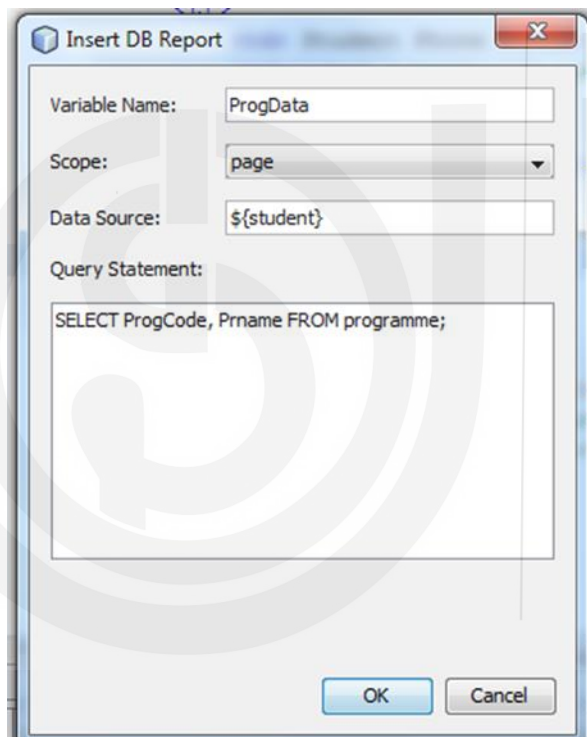


Figure 4.24: Inserting Database report for Programme code and name

Enter the required data as shown and press OK. Why did you enter Data Source as `${student}`? You may check the data source tag that you created in the previous step, the name of the variable given is student. To access this variable you need to enclosed it in `${name of the variable}`.

Please note that two new tags `<taglib ...>` has been added at line 1 and 2 of the JSP file and a new query tag has been created with the required query. Cut and put this query after the `setDataSource` tag (this is being done just to put all related instructions together – no technical reason).

You will find the following code also has been added. Since you are not interested in the column headers and table tags. Therefore you need to delete them. We have shown that code using strikeout font type:

```

<table border="0">
  <!-- column header -->
  <tr>
    <c:forEach var="columnName" items="${ProgData.columnNames}">
      <th><c:out value="${columnName}"/></th>
    </c:forEach>
  </tr>
  <!-- column data -->
  <c:forEach var="row" items="${ProgData.rowsByIndex}">
    <tr>
      <c:forEach var="column" items="${row}">
        <td><c:out value="${column}"/></td>
      </c:forEach>
    </tr>
  </c:forEach>
</table>

```

However, we have to create code that is equivalent to code (it is to be repeated for every programme.):

```

<option value="BCA">
  Bachelor of Computer Applications
</option>

```

Therefore, we must keep the outer for each loop, with only the change that we want data by rows. However, the inner loop should contain option tags, therefore, you can replace the inner loop by the option code as given above. So the new code line is:

```

<c:forEach var="row" items="${ProgData.rowsByIndex}">
  <option value="BCA">
    Bachelor of Computer Applications
  </option>
</c:forEach>

```

Now, the term BCA is actually programme code which is stored in ProgCode part of variable row. This will be replaced by <c:out value="\${row.ProgCode}"/>, likewise the Bachelor of Computer Applications will be replaced by <c:out value="\${row.Pname}"/>. Thus, you will get the following code.

```

<select name="ProgData">
  <c:forEach var="row" items="${ProgData.rows}">
    <option value="<c:out value="${row.ProgCode}"/>">
      <c:out value="${row.Pname}"/>
    </option>
  </c:forEach>
</select>

```

Now, your form is ready and you can test it, to get the display as shown in Figure 4.22. In the next section, we describe the process of processing the form data and entering it in SQL database.

4.6.3 Storing Student information in the Database

The next step will be input the information submitted by the student into the SQL student database. The file that will process the form data has been specified in the <form ...> tag itself. This content of this file is given below:

StudentDataInput.jsp

```
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<sql:setDataSource      var="student"      driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/student"
user="netbean" password="netbean7"/>

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<% @include file="top.jsp"%>

<sql:update var="insert" dataSource="${student}">
INSERT INTO studentmaster (stID,stname,stphone,ProgCode)
VALUES(?,?,?,?);
<sql:param value="${param.StudentID}" />
<sql:param value="${param.StudentName}" />
<sql:param value="${param.StudentPhone}" />
<sql:param value="${param.ProgCode}" />

</sql:update>
<sql:query var="studentmaster" dataSource="${student}">
SELECT * FROM studentmaster WHERE stID = ? <sql:param
value="${param.StudentID}" />;
</sql:query>
```

The Record Entered by you is:

```
<table border="1">
<!-- column headers -->
|  |
| --- |
|
</tr>
<!-- column data -->
<c:forEach var="row" items="${studentmaster.rowsByIndex}">
|  |
| --- |
|
<c:forEach var="column" items="${row}">
 <c:out value="${column}" /></td> </c:forEach> |

</c:forEach>

```

```

</tr>
</c:forEach>
</table>
<% @include file= "bottom.jsp"%>

```

You should create all the components of the file as explained in the previous sub- section. The portion of the this jsp program that needs explanation is the SQL statement itself. Please note the use of question marks (?) in the VALUES clause, followed by `<sql:param ...>` tags. There are four VALUES clauses and thus four `<sql:param ...>` tags. You may observe that the parameters used in `<sql:param ...>` Tags are `${param.StudentID}`, `${param.StudentName}`, `${param.StudentPhone}` and `${param.ProgCode}` conforming to the name fields of the input tages of the form `StudentInformationForm.jsp`. The names of these fields are `StudentID`, `StudentName`, `StudentPhone` and `ProgCode` respectively. Thus, the data of the form is transferred through the Insert statement to the `studentmaster` table of the student database. Once data is entered in the database, then the second query displays the content entered for this student. Thus, after filling up the form and submitting it through Submit Information Button, the data will be entered in the database and displayed in the browser window as shown in Figure 4.25.

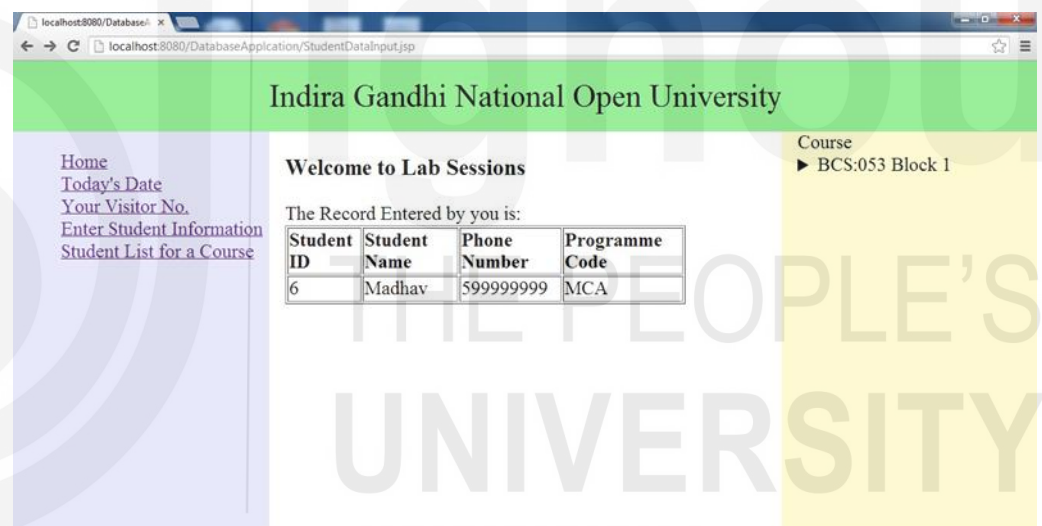


Figure 4.25: Display of Entered Student Record

You can enter several such records. Now, the final task is to create the list of the students who have enrolled in a programme. Once again, you need to create a form and the action program. The following files contain the code:

CourseStudentList.jsp

```

<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<sql:setDataSource var="student" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/student"
    user="netbean" password="netbean7"/>

<sql:query dataSource="${student}" var="ProgData">
SELECT ProgCode, Pname from programme;
</sql:query>

```



```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<% @include file= "top.jsp"%>
<form name="InputProgramme" action="CourseStudentListProcess.jsp"
method="GET">

```

```

<table border="0">
<tbody>
<td> Select Programme :</td>
<td>
<select name="ProgCode">
<c:forEach var="row" items="{ProgData.rows}">
<option value="<c:out value="{row.ProgCode}" />">
<c:out value="{row.Pname}" />
</option>
</c:forEach>
</select>
</td>
</tr>
</tbody>
</table>
<input type="submit" value="Submit" />
</form>

```

```

<% @include file= "bottom.jsp"%>

```

CourseStudentLisProcess.jsp

```

<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

```

```

<sql:setDataSource var="student" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/student"
user="netbean" password="netbean7"/>

```

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
% @include file= "top.jsp"%

```

```

<sql:query var="programname" dataSource="{student}"> SELECT
Pname FROM programme
WHERE ProgCode = ? <sql:param value="{param.ProgCode}" />;
</sql:query>

```

```

<sql:query var="studentmaster" dataSource="{student}"> SELECT s.stID,
s.stname FROM studentmaster s, programme p
WHERE s.Progcode = p.ProgCode AND p.ProgCode = ? <sql:param
value="{param.ProgCode}" />;
</sql:query>
<h4>Programme:<br>
<c:forEach var="progrname" items="{programname.rows}">
<c:out value="{progrname.Pname}" />
</c:forEach> ({param.ProgCode})
</h4>

```

```

<table border="1">
<!-- column headers -->
<tr>
<th>Student ID</th>
<th>Student Name</th>
</tr>
<!-- column data -->
<c:forEach var="row" items="${studentmaster.rows}">
<tr>
<td><c:out value="${row.stID}"/></td>
<td><c:out value="${row.stname}"/></td>
</tr>
</c:forEach>
</table>
% @include file= "bottom.jsp"%

```

You can create these files. The code is not every different to what we have already explained. The following is the browser displays when you execute these files, as shown in Figure 4.26 and 4.27. Please notice that one of the query is from two table.



Figure 4.26: Display on selection of Student List for a Course Option.

The option that is selected is Certificate in Information Technology.

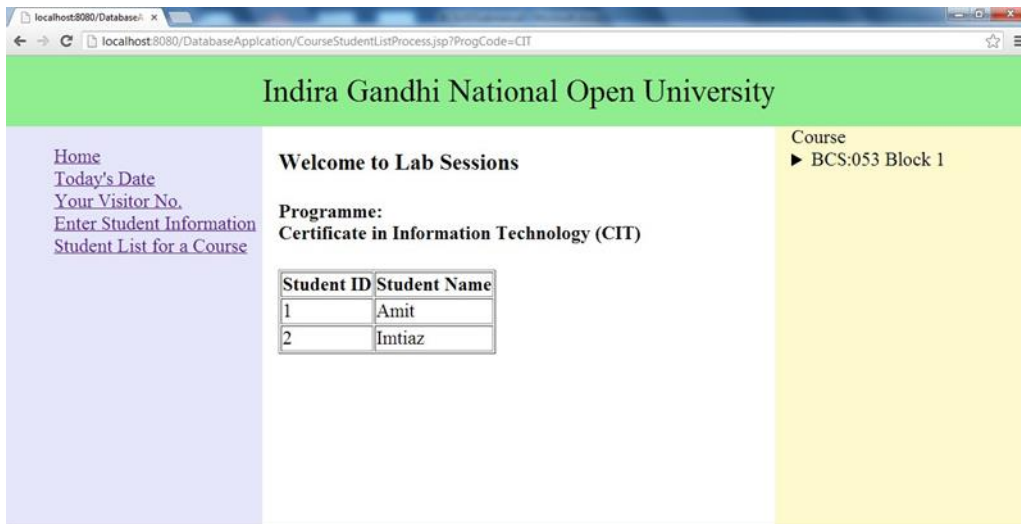


Figure 4.27: Display of Course List for CIT programme

Thus, you can develop some basic JSP based websites using netbeans using database as the backend. However, you should remember the website developed by us is very elementary and in no way a commercial web site. One of the major problem is that web server does not remember about the users. Therefore, if you are developing e- commerce application, specifically implementing Shopping Cart, you need to remember the users' identity. You can create such applications using sessions and cookies in your applications. For form processing, you can also use Servlets instead of JSP.

Some of the improvements in are developing website are suggested in the next sub- section.

4.6.4 Comments on the Website

The web pages that you have developed during the course of this lab manual can be improved in many ways. Some of the suggestions are given below:

- You may develop and use a professional structure of the web site with images, colours and many more options. This will require an exhaustive study of the requirements and purpose of the web site that you want develop.
- CSS may be further enhanced and you can also use several drop down menus.
- You can use JavaScript for checking form validations.
- You must perform thorough check on the data filled by the user by using automated ways. You may also check if data is being send by a proper user or not.
- You may add various error checks and exception handling.
- You may use servlets instead of JSP for form validations.
- You may use configuration files of Netbeans which keep track of connections.
- You may use XML and file Fragments to include HTML code in web pages.
- You may use connection pools that can enhance efficiency of web site access.

- You may use entity classes, javabeans and persistence classes of Java to enhance efficiency of access.
- You may organize web site in folders for better maintenance.
- You may use some framework for creating a website.
- You can create sessions and cookies in your applications.

There are many more possibilities. Discussion on these topics is beyond the scope of this Section. But remember a web site designer always keep looking for technologies for better, efficient, maintainable, dynamic web sites. So keep leaning.

Access to Web through Mobile devices

One final point about the development of web sites for mobiles, you have been introduced in theory to WML. However, that is now an old technology for development of mobile based web sites. A new initiative called Mobile Web Initiative (MWI) has been set by World Wide Web Consortium (W3C). This initiative is aimed at developing the good practices and tools for the access of WWW through mobiles. The small screen sizes, speed, lack of multiple windows are only some of the many limitations of web access through mobile. If you notice the current trends, then you may observe that if you are accessing some good websites through mobiles, the web address is automatically prefixed with “m.” indicating that the information being displayed on the mobile device is from the website specially designed for mobiles.

Thus, the present practice is to design a simpler website for the mobile, mostly using simpler tags and having limited information for the web pages. In the present, case you are advised to design simpler websites using jsp, servlets for display on mobile devices. Thus, the WML section of web site design is merged with the Section 9 and 10. You are advised to use either WML or design simpler web pages for the mobile access.

4.7 LIST OF LAB EXERCISE

Purpose:

This section focuses on server-side programming using PHP and database connectivity with MySQL. The exercises aim to give learners practical experience in creating dynamic, data-driven web applications. By completing these tasks, students will understand how to process user input, manage data storage, and generate web content dynamically using PHP and MySQL integration.

Session 1: Using Web 2.0 and creating pages using XHTML

1. List the features of at least 5 Web 2.0 technologies.
2. Create a simple website about you. The website should contain at least two to three pages about you. It should have a table, a menu and some photographs. You should try to create an image map in your website. You should demonstrate the use of summary tags, headings, colours etc.

3. Create a feedback form for the website that you have created. Do not create the action button. Your form should have all possible types of form input options.
4. Create a form for data entry for the marks of the students of a class. Identify the requirements for such a form by collecting few sample mark lists and then design and implement it.

Session 2: Creating Style Sheets for the web pages created in session 1

1. In the HTML pages that you have designed in Session 1, create several divisions. Identify some classes such that alternative rows of a table have different shading. The background colour of each division should be different. Some text should have different display colour and background.
2. Insert a CSS for all the forms designed in session 1. The forms should have a image as the background.
3. Create four simple web pages linked together. All the web pages should share a CSS and a drop down Menu that uses CSS and HTML code only. The menu options should be: Home, Departments, Student Support and Contact Us.

Session 3: Creating sample XML document and displaying it

1. Customer list of an organization includes the name of customers, their Home or permanent addresses, and at least two phone numbers. Create a customer list of at least four customer using XML.
2. Create the DTD for the customers you created in problem 1.
3. Create an XML document to that stores data as XML document, checks it using XML schema and displays the information as shown in the table below:

Subject	Programme	Name of Students
Database Systems	MCA	Ramesh Riaz
Advanced Operating System		Sandeep Cristopher Salim
Web Programming	BCA	Farhin Rajan

4. Create a database of books titles, authors, year of publication, publisher name, price and number of copies purchased using XML. The list should have at least 10 books and every book must have at least one author. Create the XML schema for this books data.

Session 4 Using and writing JavaScript in web pages

1. Write a JavaScript program that displays a Drop Down Menu
2. Write a JavaScript program that creates a sequence of automatically changing pictures on a web page.

3. Write JavaScript code to check if data has been properly entered in the forms you have created so far. This activity may be performed when you press the Check Button that you may create on each form.
4. Write a JavaScript program that displays the current time and updates it after every minute.
5. Write a JavaScript program that counts the number of times a Button is clicked.
6. Create a web page with two pictures. Write a JavaScript program that displays the description of the picture when mouse is brought over the picture.
7. Write a JavaScript program to demonstrate simple animation on a web page.

Session 5: Using JSP/Servlet

1. Write a JSP/Servlet program that takes your name and address from an HTML Form and displays it on a web page.
2. Write a JSP program that output current time only.
3. Write a JSP program that counts the number of times a link is clicked.
4. Create five pages of a web site having similar top and left panels. The top should have a logo on the left and name of the organization in the middle. The left should have a drop down menu. Use JSP to include it in all the web pages.
5. Create a login form and check if the user name and password entered by the user are correct.
6. Create a quiz of at least five questions and check if the questions have been answered correctly.
7. Write a JSP program that displays “Good Morning” or “Good Evening” based on the present time.
8. Writing simple applications using JSP and JDBC and deploying it on web or mobile devices
9. Create a website using JSP and JDBC that creates employee database of an organization. The employee database has two tables – employee having fields empID, name, department, present designation, present salary, emailed, year of joining the organization, and department having fields department, department name and manager. Create forms to enter information of employees and departments. You must also create a form to display list of employees of a department.
10. Create an application that creates a simple banking database with tables for customers and customer transactions. You must create a login form to verify login details from the customer table. You may create a session or use cookies if possible to perform the transactions.
11. Create a simple Web application for maintaining records of students and teachers in a School. You may study such a system. Make necessary assumptions while developing this application.
12. Design and develop any of the above website for mobile devices.

4.8 FURTHER READINGS

- 1 <https://netbeans.org/>
- 2 <http://www.w3schools.com/>

- 3 <http://www.oracle.com/>
- 4 www.wikipedia.com/

