

SYLLABUS

Web Technology [IT8501]

UNIT I : WEB SITE BASICS AND HTML

Web Essentials : Clients, Servers, and Communication. The Internet-Basic Internet Protocols -The World Wide Web-HTTP request message-response message-Web Clients Web Servers. Markup Languages: XHTML. An Introduction to HTML History-Versions-Basic XHTML Syntax and Semantics-Some Fundamental HTML Elements-Relative URLs-Lists-tables-Frames-Forms-HTML 5.0. **(Chapters - 1, 2)**

UNIT II : CSS AND CLIENT SIDE SCRIPTING

Style Sheets : CSS-Introduction to Cascading Style Sheets-Features-Core Syntax-Style Sheets and HTML-Style Rule Cascading and Inheritance-Text Properties-Box Model Normal Flow Box Layout-Beyond the Normal Flow-CSS3.0. Client-Side Programming: The JavaScript Language-History and Versions Introduction JavaScript in Perspective - Syntax - Variables and Data Types - Statements - Operators - Literals-Functions-Objects-Arrays-Built-in Objects-JavaScript Debuggers. **(Chapters - 3, 4)**

UNIT III : SERVER SIDE SCRIPTING

Host Objects : Browsers and the DOM-Introduction to the Document Object Model DOM History and Levels-Intrinsic Event Handling-Modifying Element Style-The Document Tree-DOM Event Handling-Accommodating Noncompliant Browsers Properties of window. Server-Side Programming: Java Servlets- Architecture -Overview-A Servlet-Generating Dynamic Content-Life Cycle- Parameter Data-Sessions-Cookies-URL Rewriting-Other Capabilities-Data Storage Servlets and Concurrency-Databases and Java Servlets. **(Chapters - 5, 6)**

UNIT IV : JSP AND XML

Separating Programming and Presentation : JSP Technology Introduction-JSP and Servlets-Running JSP Applications Basic JSP-JavaBeans Classes and JSP-Tag Libraries and Files-Support for the Model-View-Controller Paradigm- Databases and JSP. Representing Web Data: XML-Documents and Vocabularies-Versions and Declaration-Namespaces- DOM based XML processing Event-oriented Parsing: SAX-Transforming XML Documents-Selecting XML Data: XPATH-Template based Transformations: XSLT-Displaying XML Documents in Browsers. **(Chapters - 7, 8)**

UNIT V : AJAX AND WEB SERVICES

AJAX : Ajax Client Server Architecture-XML Http Request Object-Call Back Methods. Web Services: JAX-RPC-Concepts-Writing a Java Web Service-Writing a Java Web Service Client-Describing Web Services: WSDL- Representing Data Types: XML Schema-Communicating Object Data: SOAP Related Technologies-Software Installation-Storing Java Objects as Files. **(Chapters - 9, 10)**

TABLE OF CONTENTS

UNIT - I

Chapter - I	Web Site Basics	(1 - 1) to (1 - 20)
1.1	Web Essentials	1 - 2
1.1.1	What is Web Technology ?	1 - 2
1.1.2	Concept of Internet	1 - 2
1.1.3	Concept of World Wide Web	1 - 2
1.1.4	Internet or WWW?	1 - 3
1.2	Internet Protocol	1 - 3
1.3	HTTP Request and Response Message	1 - 5
1.4	Features of HTTP Protocol	1 - 10
1.5	Web Clients	1 - 10
1.5.1	Functions Defined by Web Browser	1 - 11
1.5.2	URL	1 - 11
1.6	Web Server	1 - 12
1.6.1	Apache	1 - 13
1.6.2	IIS	1 - 13
1.7	Client, Sever and Communication	1 - 14
	Two Marks Questions with Answers	1 - 15
Chapter - 2	HTML	(2 - 1) to (2 - 72)
2.1	Markup Languages : XHTML	2 - 2
2.2	Introduction to HTML	2 - 3
2.3	History and Versions	2 - 4
2.4	XHTML Syntax and Semantics	2 - 5
2.5	Some Fundamental HTML Elements	2 - 6

2.5.1	Heading	2 - 6
2.5.2	Paragraphs	2 - 7
2.5.3	Line Break	2 - 8
2.5.4	Setting Font Style	2 - 9
2.5.5	Text Alignment	2 - 10
2.5.6	Setting the Font and Color	2 - 11
2.5.7	Special Character	2 - 16
2.6	Relative URLs	2 - 18
2.6.1	Plying an Audio File	2 - 21
2.6.2	Uses of Links	2 - 22
2.7	Lists	2 - 22
2.7.1	Unordered List	2 - 23
2.7.2	Ordered List	2 - 24
2.8	Tables	2 - 28
2.8.1	Basic Table Tag	2 - 28
2.8.2	Setting Background to Table	2 - 31
2.8.3	Rowspan and Colspan	2 - 33
2.9	Frames	2 - 43
2.9.1	Frames with Scrollbars	2 - 46
2.10	Forms	2 - 51
2.10.1	Text	2 - 52
2.10.2	Text Area	2 - 53
2.10.3	Checkbox	2 - 54
2.10.4	Radio Button	2 - 56
2.10.5	Button	2 - 57
2.10.6	Menus	2 - 58
2.11	HTML 5.0	2 - 66
2.11.1	Features of HTML 5.0	2 - 66
2.11.2	Difference between HTML and HTML5	2 - 67
	Two Marks Questions with Answers	2 - 67

UNIT - II

Chapter - 3 CSS**(3 - 1) to (3 - 70)**

3.1 Introduction to Cascading Stylesheet.....	3 - 2
3.2 Features.....	3 - 2
3.3 Core Syntax.....	3 - 2
3.4 Selectors	3 - 4
3.4.1 Simple Selector Form.	3 - 4
3.4.2 Class Selectors	3 - 6
3.4.3 Generic Selectors	3 - 8
3.4.4 Id Selectors.	3 - 9
3.4.5 Universal Selectors	3 - 10
3.4.6 Attribute Selector	3 - 11
3.4.7 Pseudo Classes	3 - 12
3.4.8 Contextual Selector	3 - 15
3.4.9 At Rules.	3 - 16
3.5 Style Sheet and HTML	3 - 16
3.5.1 Inline Style Sheet	3 - 16
3.5.2 Document Level Style Sheet	3 - 17
3.5.3 External Stylesheet	3 - 19
3.6 Style Rule Cascading and Inheritance.....	3 - 21
3.6.1 Style Rule Cascading	3 - 21
3.6.2 Inheritance	3 - 23
3.7 Text Properties	3 - 26
3.7.1 Font Families	3 - 26
3.7.2 Font Sizes	3 - 28
3.7.3 Font Variants	3 - 29
3.7.4 Font Styles	3 - 29
3.7.5 Font Weights	3 - 30
3.7.6 Font Shorthands	3 - 31
3.7.7 Text Decoration	3 - 32

3.7.8	Alignment of Text	3 - 33
3.8	Box Model	3 - 35
3.8.1	Borders	3 - 36
3.8.2	Margins and Padding	3 - 41
3.9	Color	3 - 43
3.9.1	Color Groups	3 - 43
3.9.2	Color Properties	3 - 44
3.10	Background Image	3 - 45
3.11	Normal Flow Box Layout	3 - 49
3.11.1	Basic Box Layout	3 - 49
3.11.2	The Display Property	3 - 50
3.11.3	Margin Collapse	3 - 52
3.11.4	Inline Box	3 - 52
3.12	Beyond Normal Flow	3 - 54
3.12.1	Absolute Positioning	3 - 54
3.12.2	Relative Positioning	3 - 56
3.12.3	Float Position	3 - 57
3.13	CSS 3.0	3 - 59
3.13.1	CSS Border	3 - 59
3.13.2	CSS3 Text Effects	3 - 62
Two Marks Questions with Answers		3 - 64

Chapter - 4 Client Side Scripting

(4 - 1) to (4 - 84)

4.1	Introduction to JavaScript Language	4 - 2
4.1.1	History and Versions	4 - 2
4.1.2	Features of JavaScript	4 - 2
4.1.3	Difference between JavaScript and Java	4 - 3
4.2	Writing First JavaScript	4 - 3
4.3	Identifier, Keywords and Comments	4 - 6
4.4	Data Types	4 - 7

4.5	Variable	4 - 7
4.6	Operators	4 - 8
4.6.1	String Concatenation Operator	4 - 9
4.7	Literals	4 - 10
4.8	Input and Output.....	4 - 11
4.8.1	The document.write	4 - 11
4.8.2	Popup Box	4 - 11
4.9	Control Structures	4 - 15
4.10	Functions	4 - 23
4.10.1	Returning Value from the Function	4 - 24
4.10.2	Passing the Parameters to the Function	4 - 25
4.10.3	Passing an Array to the Function	4 - 26
4.10.4	Global Functions of JavaScript	4 - 40
4.11	Arrays	4 - 43
4.11.1	Array Declaration.....	4 - 43
4.11.2	Array Initialization	4 - 43
4.11.3	Two Dimensional Array.....	4 - 47
4.12	Standard Objects.....	4 - 51
4.12.1	Math Objects	4 - 51
4.12.2	Number Objects.....	4 - 53
4.12.3	Date Objects.....	4 - 54
4.12.4	Boolean Objects.....	4 - 56
4.12.5	String Objects.....	4 - 57
4.12.6	Object Creation and Modification	4 - 59
4.13	Form Processing in JavaScript	4 - 61
4.14	JavaScript Debuggers	4 - 70
4.15	Examples	4 - 71
	Two Marks Questions with Answers	4 - 79

UNIT - III

Chapter - 5 Document Object Modeling	(5 - 1) to (5 - 38)
5.1 Introduction to the Document Object Model	5 - 2
5.2 DOM History and Levels	5 - 2
5.3 The Document Tree	5 - 2
5.4 Modifying Element Style	5 - 4
5.4.1 Accessing Elements using DOM	5 - 4
5.4.2 Modifying Elements using DOM	5 - 6
5.5 Intrinsic Event Handling	5 - 10
5.5.1 Handling Events from the Body Elements	5 - 15
5.6 DOM2 Event Model	5 - 16
5.6.1 Event Propagation	5 - 17
5.6.2 Event Handler Registration	5 - 19
5.6.3 Examples	5 - 21
5.7 Accommodating Noncompliant Browsers	5 - 31
5.7.1 Detecting Host Objects	5 - 31
5.8 Properties of Window	5 - 32
Two Marks Questions with Answers	5 - 34
Chapter - 6 Server Side Scripting	(6 - 1) to (6 - 64)
6.1 Introduction to Server Side Programming	6 - 2
6.2 Java Servlets	6 - 2
6.2.1 Need and Advantages	6 - 3
6.3 Architecture	6 - 4
6.4 Life Cycle	6 - 9
6.5 Servlet API	6 - 10
6.5.1 The javax.servlet package	6 - 10
6.5.1.1 Interfaces	6 - 10
6.5.1.2 Classes	6 - 13

6.5.2	The javax.servlet.http Package	6 - 13
6.5.2.1	Interface.	6 - 14
6.5.2.2	Classes	6 - 15
6.6	Parameter Data	6 - 17
6.7	Session Management Technique	6 - 24
6.8	Session Management using Session ID	6 - 25
6.9	Cookies	6 - 28
6.9.1	Servlet Support for Cookies	6 - 29
6.9.2	Examples.....	6 - 30
6.10	URL Rewriting.....	6 - 36
6.11	Data Storage.....	6 - 39
6.12	Servlet and Concurrency	6 - 39
6.13	Database and Java Servlets	6 - 40
6.13.1	Structured Query Language using MySQL.....	6 - 40
6.13.2	JDBC	6 - 45
6.13.3	Connectivity between Database and Servlet	6 - 47
6.13.4	Servlet-Database Examples.....	6 - 48
Two Marks Questions with Answers		6 - 57

UNIT - IV

Chapter - 7 Separating Programming and Presentation (7 - 1) to (7 - 44)

7.1	JSP Technology	7 - 2
7.2	JSP and Servlet	7 - 2
7.3	JSP Document Structure	7 - 3
7.4	Running JSP Application	7 - 4
7.5	Declaration	7 - 5
7.6	Directives.....	7 - 7
7.7	Comments in JSP	7 - 8

7.8 Scripting Elements.....	7 - 9
7.9 Actions and Templates	7 - 12
7.10 JavaBean Classes and JSP	7 - 17
7.11 Tag Libraries and Files	7 - 21
7.11.1 Core Tags	7 - 21
7.12 Support For Model View Controller Paradigm	7 - 31
7.13 Database and JSP.....	7 - 32
Two Marks Questions with Answers	7 - 40

Chapter - 8 Representing Web Data:XML (8 - 1) to (8 - 38)

8.1 Introduction.....	8 - 2
8.2 Documents and Vocabularies.....	8 - 3
8.2.1 Elements and Attributes	8 - 3
8.2.1.1 Declaring Elements.	8 - 4
8.2.1.2 Declaring Attributes	8 - 5
8.2.1.3 Declaring Entities	8 - 7
8.2.2 Rules	8 - 8
8.3 XML Versions and XML Declaration	8 - 9
8.4 Namespace.....	8 - 9
8.5 Document Type Definition (DTD)	8 - 10
8.6 Introduction to DOM and SAX.....	8 - 15
8.6.1 Difference between DOM and SAX	8 - 15
8.7 DOM based XML Processing.....	8 - 16
8.8 Event-Oriented Parsing : SAX	8 - 19
8.9 XSLT : Displaying XML Documents in Browsers.....	8 - 22
8.9.1 Transforming XML into XSLT.	8 - 22
8.9.2 XSL Elements	8 - 23
8.10 Displaying XML Documents in Browser using CSS.....	8 - 30
Two Marks Questions with Answers	8 - 32

UNIT - V

Chapter - 9 AJAX **(9 - 1) to (9 - 14)**

9.1	AJAX Client Server Architecture	9 - 2
9.1.1	Introduction to AJAX	9 - 2
9.1.2	Architecture	9 - 2
9.2	XMLHttpRequest Object.....	9 - 4
9.3	Call Back Methods	9 - 7
9.4	Coding AJAX Script.....	9 - 8
Two Marks Questions with Answers		9 - 13

Chapter - 10 Web Services **(10 - 1) to (10 - 44)**

10.1	Concept of Web Service	10 - 2
10.2	JAX- RPC Concepts.....	10 - 3
10.2.1	Concept of RPC	10 - 3
10.2.2	JAX-RPC	10 - 3
10.3	Writing a Java Web Service	10 - 5
10.4	Describing Web Services: WSDL.....	10 - 23
10.5	Representing Data Types: XML Schema	10 - 26
10.5.1	Data Types	10 - 28
10.5.2	Advantages and Disadvantages of Schema	10 - 30
10.6	Communicating Object Data: SOAP.....	10 - 31
10.6.1	Structure of SOAP	10 - 32
10.6.2	SOAP and HTTP	10 - 33
10.6.3	SOAP Encoding	10 - 34
10.6.4	RPC Representation	10 - 35
10.7	Storing Java Objects as Files	10 - 36
10.7.1	The Serializable Interface	10 - 37
Two Marks Questions with Answers		10 - 39

Web Technology Laboratory **(L - 1) to (L - 30)**

STUCOR APP

Unit I

<h1>1</h1>	<h2>Web Site Basics</h2>
------------	--------------------------

Syllabus

Web Essentials : Clients, Servers, and Communication. The Internet-Basic Internet Protocols -The World Wide Web-HTTP request message-response message-Web Clients Web Servers.

Contents

1.1	Web Essentials	
1.2	Internet Protoco.....	Dec.-09,11, May-14, Marks 8
1.3	HTTP Request and Response Message.....	June-09, May-12, 13,14, Dec.-12, 13 Marks 8
1.4	Features of HTTP Protocol	
1.5	Web Clients	Dec.-11, Marks 8
1.6	Web Server	May-10,12,13 Marks 12
1.7	Client, Sever and Communication	
	Two Marks Questions with Answers	

1.1 Web Essentials

1.1.1 What is Web Technology?

- Web technology is a technology that acts as an interface between web client and web server. It includes markup languages, programming interfaces, online presentation tools, Java applets, video editing tools and so on.
- The commonly used client side scripting languages are HTML, JavaScript, CSS, XML and so on.
- The commonly used server side technologies are ASP, JSP, Java Servlets, PHP and so on.

1.1.2 Concept of Internet

Definition : Internet is global system in which millions of computers are connected together. It is basically a network of networks.

- Using internet many people can share resources and can communicate with each other. To have internet service one must go to the service providers. That means your computer must be connected to the Internet Service Providers (ISP) through phone-line modem or DSL.
- There are some privately owned internet service providers from which we can hire the internet services.

1.1.3 Concept of World Wide Web

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

- The world wide web is a **information system** in which various documents containing information are interlinked together. User can access this information or write the information via computers. This information is typically stored on the web pages and through web browsers we can access these web pages.
- The web pages may contain the information in the form of text, audio, video, images and graphics. We can navigate between the web pages using hyperlinks.
- The concept of WWW was introduced by **Sir Tim Berners-Lee** the contractor at the **European Organization for Nuclear Research (CERN)**, Switzerland in 1980. He built a personal database of people and software models and used hypertext so that each new page of information was linked to an existing page.

1.1.4 Internet or WWW?

- The term internet and WWW is often used interchangeably, but these are two different terms.
- The **internet** is collection of computers and other devices (such as printers, scanners etc.) connected together whereas World Wide Web (WWW) is collection of software and corresponding protocols used to access the resources over the network.
- The world wide web contains huge amount of documents, images and other resources which can be accessed using the hyperlinks.
- Thus people use internet through the Web.

1.2 Internet Protocol

Dec.-09, 11, May - 14

Various protocols used in internetworking are -

(1) File Transfer Protocol (FTP)

- The file transfer protocol sets the rules for transferring files between computers.
- When user wants to download a file from the server FTP is used.
- FTP uses **two connections** between client and server. One connection is used for actual **data transfer** and other is used for **control information** (used for commands). This separation of data and commands makes the FTP more efficient.

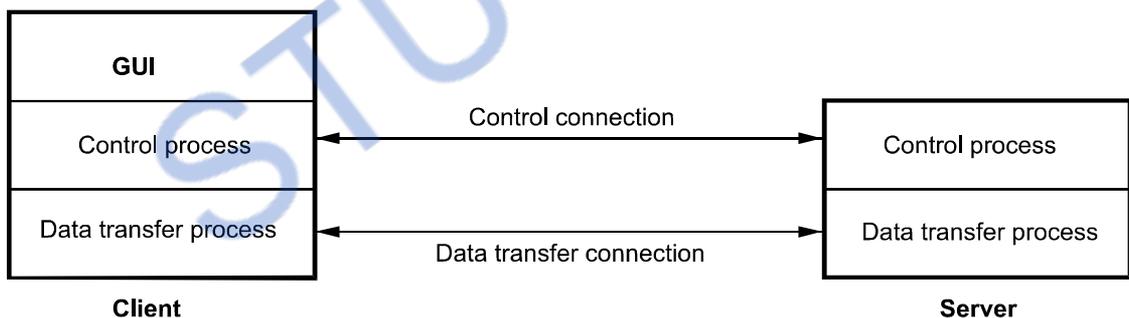


Fig. 1.2.1 Two connections used in FTP processing

- When client makes a request for particular file download then using the data transfer connection actual data gets transmitted from server to the client.
- At the same time server keeps track of how much data is sent so far and how much is remaining. This tracking can be done using the control transfer connection.
- Hence during the file downloading/uploading we can see a message about how many bytes are getting transferred and how much time is remaining.

(2) HTTP

- **The Hyper Text Transfer Protocol (HTTP)** is a **request/response** protocol.
- It is a **communication protocol** used to transfer the information on local area network and World Wide Web (WWW).
- It is the network protocol used to deliver virtually all files and other data (collectively called resources) on the World Wide Web, whether they're HTML files, image files, query results, or anything else. Usually, HTTP takes place through TCP/IP sockets.
- It is also called as a **stateless protocol** because this protocol is not able to maintain the previous conversation/information.

(3) SNMP

- **Simple Network Management Protocol (SNMP)** is a protocol which enables network administrators to manage network devices and to diagnose network problems.
- The network management system is based on two main elements : a **supervisor** and **agents**.
 - The supervisor is the terminal at which the network administrator requests for network management.
 - The agents are found at the level of each interface connecting the managed devices to the network. With the help of these agents information on the different objects (such as switch, hub, routers) can be collected.

(4) SMTP

- **Simple Mail Transfer Protocol (SMTP)** is a simple protocol which is extensively used for transfer of **e-mails to remote servers**.
- It is an asynchronous protocol, because it allows delayed delivery of message.
- With the help of mail transfer agent and user agent the SMTP sends and receives the emails.

(5) POP3

- **Post Office Protocol version 3 (POP3)** is used by local email clients (such as Microsoft Outlook Express).
- The POP3 protocol works only at the receivers end and has no work at the sender's end.
- The POP protocol has two parts, a client POP i.e. receiver's POP and a server POP i.e. receiver's email server. The client i.e. the receiver opens TCP connection with receiver's POP server. This client must be authenticated first by using the user name and password. Then the client can receive the emails from the mailbox.

(6) TCP

- The Transmission Control Protocol is used for,
 1. Safe delivery of data
 2. Error detection
 3. Assurance of the correct sequencing of data being received.
- This protocol is called **connection oriented protocol** because before sending the data this protocol requires that two computers have established connections.
- The TCP allows the transmission of arbitrary amount of data by breaking it into stream of separate **IP packets**.
- These IP packets are numbered so that it could be reassembled properly at arrivals. Along with the data an **acknowledgement** is also sent/received in order to know whether the **reliable connection** has occurred or not.

(7) UDP

- The user datagram protocol is a **connectionless** protocol without any error detection facility.
- This protocol is used for simply transmission of data.
- The UDP is known as an **unreliable protocol** however this is much faster than TCP.

(8) IP

- Internet Protocol (IP) is a network layer protocol which consists of addressing information.
- Using this information the communication between uniquely addressed computers is possible.

University Questions

1. Explain the various Internet protocols used for client server communication.

AU : Dec.-09, Marks 8, May-14, Marks 8

2. Explain TCP/IP in detail.

AU : Dec.-11, Marks 8

1.3 HTTP Request and Response Message

AU : June-09, May-12, 13,14, Dec.-12, 13, Marks 8

- The basic feature of HTTP protocol is that it follows the **request response** model.
- The client makes a request for desired web page by giving the URL in the address bar. This **request** is submitted to the web server and then web server gives the **response** to the web browser by returning the required web page.

1. HTTP Request Message Structure

The basic structure of request message is given by following general form -

<Start line>
<Header fields>
<Blank Line>
<Message Body>

<start Line>

The start line consists of three parts which are separated by a single space. These parts are -

i) Request method ii) Request-URI iii) HTTP version

(i) **Request Method** : Various methods used for making the request are enlisted in the following table

HTTP method	Description
GET	The GET method is used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation by browsers.
POST	The POST method is used to request the server for desired web page and the request made is accepted as a new subordinate of the resource identified. The POST method is used for operations that have side effects and cannot be safely repeated. For example, transferring money from one bank account to another has side effects and should not be repeated without explicit approval by the user.
HEAD	The HEAD method is identical to GET. The only difference is that the server should not return a message-body in the response. The meta-information contained in the HTTP headers in response to a HEAD request should be similar to the information sent in response to a GET request.
OPTION	This method supports for the specified URL. It can be used to check the functionality of a web server by requesting '*' instead of a specific resource.
PUT	This method uploads a representation of the specified resource.
DELETE	This method is useful in deleting the specified resource.
TRACE	When request is made using TRACE method the server echoes back the received request so that a client can see what intermediate servers are adding or changing in the request.

(ii) Request-URI:

- The **Uniform Resource Identifier (URI)** is a string used to identify the names or resources on the Internet.
- The URI is a combination of URL and URN.
- The URL stands for **Uniform Resource Locator** and URN stands for **Uniform Resource Name**.
- The web address denotes the URL and specific name of the place or a person or item denotes the URN. For example

urn : ISBN 978-81-8431-123-2

specifies the address of some book.

- If the URI is written in the form of http: then it is both an URI and URL but there are some other URI which can also be used as URL. For example

URL	Intended server
ftp://ftp.mywebsite.com/index.txt	File can be located on FTP server
telnet://mywebsite.org	Telnet server
mailto:myself@mywebsite.org	Mail box
http://www.mywebsite.com	Web server

(iii) HTTP Version : The first HTTP version was HTTP/0.9 but the official version of HTTP was HTTP/1.1

<Header Field> and <Message Body>

- The host header field is associated with the http request.
- The header fields are in the form of field name and field value.
- Thus typical structure of http request is given by following example -

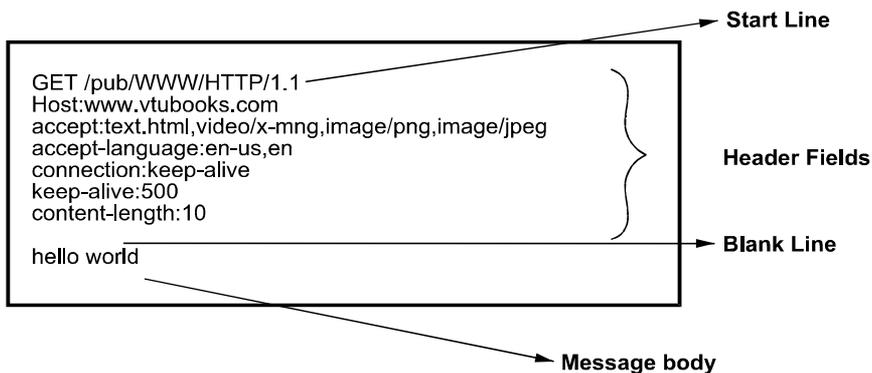


Fig. 1.3.1 HTTP request message structure

2. HTTP Response Message Structure

The basic structure of response message is given by following general form -

<Status line>
<Header fields>
<Blank Line>
<Message Body>

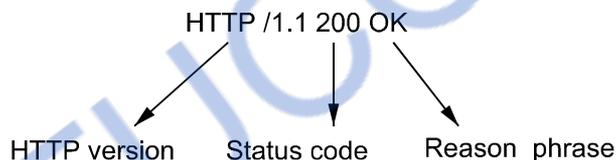
<Status Line>

Status line is similar to the start line in the request message. It consists of three fields.

HTTP version	Status code	reason phrase
--------------	-------------	---------------

The HTTP version denotes the HTTP version such as HTTP/1.1. The **status code** is a numeric code indicating the type of response. The **reason phrase** is in the text string form and presents the information about the status code.

For example -



Following table explains some commonly used status codes.

Status code	Reason phrase	Description
200	OK	This is a standard response for successful request.
201	Created	It shows that the request is fulfilled and a new resource is being created.
202	Accepted	When the request is accepted for processing but is not processed yet is denoted by this status code.
301	Moved permanently	The URI for requested resource is moved at some another location.
401	Unauthorized	The requested resource is protected by some password and the user has not provided any password.
403	Forbidden	The requested resource is present on the server but the server is not able to respond it.

404	Not Found	The requested resource is not present currently but may be available in future.
500	Internal Server Error	It is a generic error message that appears due to software internal failure.

<Header Fields>

The **header field** in response message is similar to that of request message.

<Message Body>

The message body consists of response message.

For example

```
HTTP/1.1 200 OK
Date: Sat, 30 Mar 2019 07:59:01 GMT
Server: Apache/2.0.50 (Unix) mod_perl/1.99_10 Perl/v5.8.4
mod_ssl/2.0.50 OpenSSL/0.9.7d DAV/2 PHP/4.3.8
Last-Modified: Mon, 23 Feb 2009 08:32:41 GMT
Accept-Ranges: bytes
Content-Length: 2010
Content-Type: text/html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>...</html>
```

The response header fields are enlisted in following table

Header field	Description
Date	It represents the date and time at which the response is generated.
Server	The name of the server which is responding
Last-Modified	The date and time at which the response is last modified.
Accept-ranges	It specifies the unit which is used by the client to accept the range request. For example if there is a large document and only a single web page is currently needed then this specifies the Accept-range.

University Questions

1. Write short note on HTTP protocol **AU : June-09, Marks 6**
2. Explain the structure of HTTP request and response messages. **AU : May-12, Dec.-12, Dec.- 13 Marks 8**
3. Write and explain HTTP request message format. **AU : May-13, Marks 4**
4. Write the header format of Request and Response between Client/Server and explain it. **AU : May-14, Marks 8**



1.4 Features of HTTP Protocol

1. It is a **communication** protocol used between web browser and web server.
2. This protocol is based on **request-response** messaging. That means client makes the request of desired web page and then the server responds it by sending the requested resource.
3. It is a **stateless protocol**. That means HTTP protocol can not remember the previous user's information nor it remembers the number of times the user has visited particular website.
4. The request-response message consists of plain text in fairly **readable form**.
5. The HTTP protocol has a **cache control**. This is an advanced feature of HTTP. Most of the web browsers automatically store(Cache) the recently visited web pages. This is very useful feature because if the user requests the same web page that has been visited already then it can be displayed from the cache memory instead of requesting the web server and bringing it from there.

1.5 Web Clients

AU : Dec.-11, Marks 8

- Web client is a kind of software that **runs on the clients machine**.
- This software sends the HTTP request to the server and then processes the HTTP response.
- Various forms of web client software are -
 - (1) Web browsers
 - (2) Browsers running on mobile phones
 - (3) Software robots which crawl on the web for retrieving the information
 - (4) User agents which assists the user in browsing the web.
- **Web browser** is a kind of software which is basically used to use resources on the web.
- Over the networks, two computers communicate with each other. In this communication, when request is made by one computer then that computer is called a **client** and when the request gets served by another computer then that computer is called **server**. Thus exchange of information takes place via Client-Server communication.
- When user wants some web document then he makes the request for it using the web browser.
- The browsers are the programs that are running on the clients' machines. The request then gets served by the server and the requested page is then returned to the client. It is getting displayed to the client on the web browser.
- The **commonly used web browsers** are (i) Internet explorer (ii) Mozilla Firefox (iii) Netscape Navigator (iv) Opera (v) Google Chrome (vi) Safari.
- Web browser supports variety of **protocols** but the most commonly used protocol on the web browser is Hyper Text Transfer Protocol(HTTP). This protocol is typically used when browser communicates with the server.



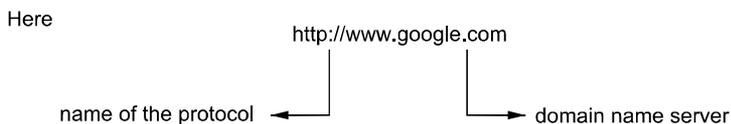
1.5.1 Functions Defined by Web Browser

Various functions of web browser are -

1. Reformats the URL and send a valid HTTP request.
2. When user gives the address of particular web site it is in the form of domain name. The web browser coverts the DNS to corresponding IP address.
3. The web browser establishes a TCP connection with the Web browser while processing the user’s request.
4. The web browsers send the HTTP request to the web server.
5. The web server processes the HTTP request sent by the web browser and returns the desired web page to the client’s machine. The web browser on the client’s machine displays this web page in appropriate format.

1.5.2 URL

- The Uniform Resource Locator (URL) is unique address for the file that has to be accessed over the internet.
- When we want to access some website we enter it’s URL in the address bar of the web browser.
- For example if we want to access **www.google.com** then we must specify its URL in the address bar as shown



However any other file such as some text file or image file or some HTML file can also be specified. The URL contains name of the protocol such as **http://**

- The URL may contain the name of the protocol as such as ftp. For example
ftp://ftp.funet.fi/pub/standards/RFC/rfc2166.txt
- The protocol identifier and the resource name are separated by a colon and two forward slashes. The syntax of writing URL is as given below,
protocol://username@hostname/path/filename
- Sometimes instead of domain name servers IP addresses can also be use. For example,
http://192.168.0.1

But use of IP address as URL is not preferred because human can not remember numbers very easily but they can remember names easily.

Absolute and Relative URL

- The absolute URL is a URL which directly point to a file. It exactly specifies exact location of a file or directory on the internet. Each absolute URL is unique. For example -
http://www.vtubooks.com/home.aspx
- The relative URL points to the file or a directory in relation to the present directory. For example
- Consider the **absolute address** which refers an image mother.jpg
http://www.mywebsite.com/myphotos/mother.jpg
- For the above given absolute address the **relative address** will be -
../myphotos/mother.jpg

That means from the current URL the directory **myphotos** will be searched for the image **mother.jpg**. The two periods .. instruct the server to move up one directory which is the root directory, then enter myphotos directory (**myphotos**) and finally point at **mother.jpg**. Thus using relative URL writing of long path name can be avoided.

University Question

1. Explain the use of relative URL with example.

AU : Dec.-11, Marks 8

1.6 Web Server

AU : May-10,12,13 , Marks 12

- Web server is a special type of server to which the web browser submits the request of web page which is desired by the client.
- There are some popularly used web servers such as Apache and IIS from Microsoft.

Functions of web server

Various functions of web server are -

1. The web servers **accepts the requests** from the web browsers.
2. The user **request is processed** by the web server.
3. The web servers respond to the users by **providing the services** which they demand for over the web browsers.
4. The web servers **serve the web based applications**.
5. The DNS translate the **domain names into the IP addresses**.
6. The servers **verify** given address exists, **find necessary files**, run appropriate scripts exchange cookies if necessary and returns back to the browser.
7. Some servers actively participate in **session handling techniques**.

1.6.1 Apache

- It is an excellent server because of its two important features : **Reliability** and **Efficiency**.
- Secondly it is more popular because it is an **open source software**. That means it is freely available to anybody. Apache web server is best suitable for UNIX systems but it can also be used on Windows box.
- The apache web server can be configured as per the requirements using the file **httpd.conf**. This file is present in the Apache software package.

1.6.2 IIS

- The **Internet Information Services** or Internet Information Server is a kind of web server provided by Microsoft.
- This server is most popular on Windows platform.

Following are some differences between Apache and IIS servers

Sr. No.	Apache web server	IIS web server
1.	Apache web server is useful on both Unix based systems and on Windows platform.	IIS web server is used on Windows platform.
2.	It is an open source product.	It is a vendor specific product and can be used on windows products only.
3.	The Apache web server can be controlled by editing the configuration	For IIS server, the behaviour is controlled by modifying the window based management programs called IIS

	file httpd.conf	snap in. We can access IIS snap-in through the Control-Panel->Administrative Tools.
--	-----------------	--

University Questions

- | | |
|---|-----------------------------|
| 1. Explain briefly the IIS Web Server. | AU : May-10, Marks 8 |
| 2. Explain in detail the functions of web server. | May-12, Marks 8 |
| 3. Explain the capabilities of web client and web server. | May-13, Marks 12 |

1.7 Client, Sever and Communication

- In this architecture, there are 3 components.
 - 1) Client PC or Web client
 - 2) An application server or web server
 - 3) A database server.

Working

- Step 1 :** The client PC or web client submits the request for desired web page to the web server.
- Step 2 :** The work of server is distributed among application server and database servers. Application server possess the required communication functions.
- Step 3 :** The data required by this business logic is present on database server. The required data is returned to application servers
- Step 4 :** The web server or application server prepares the response page and sends it to the web client.

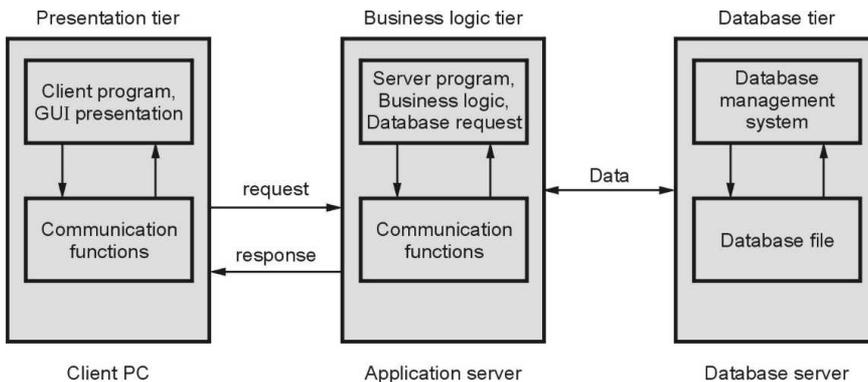


Fig. 1.7.1 Client Server Communication

Advantages :

1. High performance and persistent objects take part in communication.
2. This architecture is scalable. That means it can support increased number of users and resources.
3. High degree of flexibility in deployment platform and configurations.
4. Code/data reusability can be achieved.
5. Improved data integrity
6. Improved security as clients can not directly access the database.
7. The maintenance and modifications can be done effectively without disturbing other modules in the architecture.

Disadvantages :

1. The complexity gets increased.
2. Cost for network maintenance and deployment is increased.

Two Marks Questions with Answers**Q.1 List any two types of web servers with its application usage.****AU : May-09**

Ans. : There are two types of web servers one is vendor specific and another one is open source. The example of vendor specific web server is IIS and example of open source web server is Apache.

The IIS web server is used on Windows and Apache web server can be used on both Unix and Windows platform.

Q.2 Why is HTTP called as stateless protocol ?**AU : Dec.-09**

Ans. : The HTTP protocol can not remember the previous user's information nor it remembers the number of times the user has visited the particular web page. That means it can not remember the previous states. Hence HTTP is called stateless protocol.

Q.3 What is a web server ? Explain its functionalities.**AU : Dec.-09**

Ans. : Web server is a special type of server to which the web browser submits the request of web page which is desired by the client. For example: IIS ,Apache.

Web server processes the request of the user which it submits using the web browser. Sometimes the web servers access the database system and fetch the required information. The required web page is compiled and sent to the user in his/her web browser.

Q.4 State the use of web server logs and list the contents of a message log.**AU : May-11**

Ans. : Web server log records the information about the server activity. It stores the information about every http request processed by the server. A web server may produce more than one message logs.

Every message log contains variety of debugging information and other information generated by web application. Following information is present in the message log -

- A. Host name or IP address of the client machine making the request.
- B. Name of the user.
- C. Date and time of response along with the time zone.
- D. Start line of HTTP request.
- E. HTTP status code of response.
- F. Number of bytes sent in body of response.

Q.5 What is the use of cache control in HTTP response header ?

Ans. : Cache is used as a temporary repository. Use of cache improves the performance. Many web browsers store web pages viewed by the client in the cache memory. This brings efficiency in browsing the web page. The cache control help the user to display the web page quickly.

Q.6 List some basic internet protocols.

AU : May-13

Ans. : Following are some basic Internet protocols.

- i. http
- ii. ftp
- iii. smtp
- iv. pop3.

Q.7 For handling the mails which protocol is used ?

Ans. : There are two commonly used protocols for handling the emails. Simple Mail Transfer Protocol (SMTP) is a simple protocol which is used to transfer emails to remote servers. Another protocol is POP3 i.e. Post Office Protocol Version 3. This protocol works only at the receivers end and has no work at the sender's end.

Q.8 Give the functionalities of IP protocol.

Ans. : There are two functionalities provided by Internet Protocol -

Decomposition of the initial information flow into packets of standardized size and reassembling of data at the destination.

The Internet protocol **routes** the packet through successive networks, from the source machine to the destination which can be identified by its IP address.

Q.9 What is the difference between WWW and Internet ?

Ans. : The WWW and Internet are the two different terms. The Internet is a collection of computers and other devices such as printers, scanners etc. connected together whereas WWW is collection of software and corresponding protocol used to access the resources over the network.

Thus World Wide Web (WWW) contains huge amount of documents, images and other resources which can be accessed using the hyperlinks. People use Internet through web.

Q.10 Explain any two functionalities of web browser.

Ans. : Various functions of web browser are -

1. Reformat the URL and send a valid HTTP request.
2. When user gives the address of particular website it is in the form of domain name. The web browser converts the DNS to corresponding IP address.
3. The web browser establishes a TCP connection with the web browser while processing the user's request.
4. The web browsers send the HTTP request to the web server.

Q.11 What is URL ? Write different parts of URL.

Ans. : The general format of URL is -

Scheme:Address

That is

protocol://username@hostname/path/filename

The scheme specifies the **communication protocol**. Different schemes have different forms of addresses.

Various schemes that are used are http, ftp, gopher, file, mailto, news and so on.

Example : <http://www.vtubooks.com/index.aspx>

Q.12 Define URI.

AU : May-11

Ans. : The Uniform Resource Identifier (URI) is a string used to identify the names or resources on the Internet. The URI is a combination of URL and URN. The URL stands for **Uniform Resource Locator** and URN stands for **Uniform Resource Name**. The web address denotes the URL and specific name of the place or a person or item denotes the URN. For example

urn:ISBN 978-81-8431-123-2

specifies the address of some book.

Every URI consists of two parts, the part before the colon : denotes the **scheme** and the part after colon depends upon the scheme. The URIs are case insensitive but generally written in lower case. If the URI is written in the form of http: then it is both an URI and URL.

Q.13 What is GET and POST request ?

Ans. : The GET method is used to retrieve the information from a specified URI. The POST method is used to request the server for desired web page. This is supposed to be a safe method of retrieving the information.

Q.14 What is www ?

Ans. : The WWW is a World Wide Web which is a collection of software and corresponding protocols used to access the resources over the network. The WWW contains huge amount of documents, images, and other resources which can be accessed using hyperlinks. Thus Internet can be used through the web.

Q.15 What is TCP ?

Ans. : The TCP stands for Transmission Control Protocol. This is the connection oriented protocol which help in communicating two machines. Using TCP the message gets transferred in an orderly manner.

Q.16 List any four common browsers.**AU : Dec.-11**

Ans. : The common browsers are Internet Explorer, Mozilla Firefox, Google Chrome, Safari

Q.17 State the uses of Internet protocol.**AU : May-12**

Ans. : Following are the uses of Internet protocols -

- i) The Internet protocol is used to decompose the initial information flow into packets of the standard size. At the destination these packets are reassembled back.
- ii) The Internet protocol routes the data packets so that transmission of data is possible to the machine having desired IP address.

Q.18 State the functions of DNS and protocol used.**AU : Dec.-12**

Ans. : Following are the functions that are carried out by DNS –

1. Accepting and then requesting the programs to convert domain names to IP addresses.
2. Accepting and then requesting the other DNS servers to convert domain names to IP addresses.

Q.19 Explain the way in which a DNS server resolves the address ?**AU : Dec.-13**

Ans. : Using the domain name space the computer which is to be accessed is identified uniquely. The internet logically arranges the domain names in an hierarchical form. Secondly, the DNS server maintains the huge database of domain names. The desired address is searched in this database and if the IP address of corresponding name is found then IP address is returned to user computer.

Q.20 List the two forms of URL and its uses.**AU : May-14**

Ans. : The two forms of URL are Absolute URL and Relative URL. The absolute URL directly points to a file whereas the relative URL points to the file or directory in relation to the present directory. The URL is a unique address of the file which is accessed over the internet.

Q.21 Differentiate between internet and intranet.**AU : CSE : Dec.-15****Ans. :**

Sr. No.	Internet	Intranet
1.	Internet is a network of computers and it is open for all.	Intranet is a network of computers but it is designed for specific group of users.
2.	The number of users of internet are unlimited.	The number of users of intranet are limited.
3.	Internet contains different sources of information on variety of subjects.	The intranet contains limited information on limited subjects.
4.	Internet contains large number of intranets.	Internet can be accessed from the intranet but with restrictions.
5.	It is a collection of various LANs, WANs and MANs.	It is mostly any of LAN, WAN or MAN.

□□□

Notes

STUCOR APP

Unit I

2	HTML
----------	-------------

Syllabus

Markup Languages : XHTML. An Introduction to HTML History-Versions-Basic XHTML Syntax and Semantics-Some Fundamental HTML Elements-Relative URLs-Lists-tables-Frames-Forms-HTML 5.0.

Contents

2.1	Markup Languages: XHTML	Dec.-11,	Marks 4
2.2	Introduction to HTML		
2.3	History and Versions		
2.4	XHTML Syntax and Semantics	Dec.-11,	Marks 4
2.5	Some Fundamental HTML Elements	Dec.-13,	Marks 8
2.6	Relative URLs	Dec.-08, 09,	Marks 2
2.7	Lists	Dec.-11, May-11,	Marks 8
2.8	Tables	May-12, Dec.-13,	Marks 8
2.9	Frames	May-13, 14, Dec.-09,	Marks 16
2.10	Forms	May-09, Dec.-12,	Marks 16
2.11	HTML 5.0		
	Two Marks Questions with Answers		

2.1 Markup Languages : XHTML**AU : Dec.-11, Marks 4**

- XHTML stands for EXtensible HyperText Markup Language
- The XHTML syntax rules are specified by the file xhtml11.dtd file. Hence specify the name of this file at the beginning of XHTML script.

Difference between HTML and XHTML

Sr. No.	HTML	XHTML
1.	The HTML tags are case insensitive . Hence <body> or <BODY> or <Body> are treated as one and the same.	The XHTML is case sensitive and all the tags in XHTML document must be written in lower case.
2.	We can omit the closing tags sometimes in HTML document.	For every tag there must be a closing tag . Some browsers get confused if the closing tag is not given. There are two ways by which we can mention the closing tags <pre> </pre> or <pre></pre>
3.	In HTML the attribute values it not always necessary to quote the attribute values. In fact numeric attribute values are rarely quoted in HTML. Only if some special characters or white spaces are present in the attribute values then only it is essential to put quotes around them in HTML.	In every XHTML document the attribute values must be quoted.
4.	In HTML there are some implicit attribute values.	In every XHTML the attribute values must be specified explicitly.
5.	In HTML even if we do not follow the nesting rules strictly it does not cause much difference.	In XHTML document the nesting rules must be strictly followed. These nesting rules are- A form element can not contain another form element. An anchor element does not contain another form element. List element can not be nested in the list elements. If there are two nested elements then the inner

element must be enclosed first before closing the outer element.

Text elements can not be directly nested in form elements.

University Question

1. List the two differences between HTML and XHTML with respect to elements.

AU : Dec.-11, Marks 4

2.2 Introduction to HTML

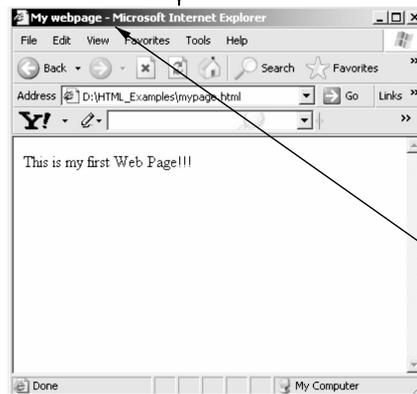
- HTML stands for HyperText Markup Language. **Hypertext** is simply a piece of text that works as a link.
- **Markup Language** is language of writing layout information within documents.
- Basically an HTML document is a plain text file.
- It contains rich text. The rich text means text with tags.
- HTML is **not a case sensitive** language.
- Any HTML program can be written in simple Notepad or WordPad text editors. The extension to this program should be either **html** or **htm**.
- This program then can be opened in some web browser and the corresponding web page can be viewed. Let us create our first web page using HTML.

HTML program *mypage.html*
written in Notepad



```
<html>
<head>
<title>My webpage</title>
</head>
<body>
This is my first Web Page!!!
</body>
</html>
```

This HTML webpage can be
viewed in web browser
(Internet Explorer). This is an **output**



HTML Document

```
<html>
<head>
<title>My webpage</title>
</head>
<body>
This is my first Web Page!!!
</body>
</html>
```

Script Explanation

- 1) Note that the program consists of some strings enclosed within **angular brackets**. Such strings are called **tags**.
- 2) The HTML program should be written within **<html>** and **</html>**. The **<html>** indicates the start of html program and **</html>** denotes end of html program. Use of slash (/) in the angular bracket indicates end of that particular tag.
- 3) Any HTML program has two prominent parts : **head and body**. The head part acts as a header of a file and contains some information like setting the title of web page.

Comments in HTML

- The comment in HTML can be denoted as follows -
`<!--It is a comment statement -->`
- There should **not be a space** between angular bracket and exclamation mark. This comment is beginning with `<!--` and ending with `-->`. Also note one thing that there should not be any `--` inside the comment.

2.3 History and Versions

- **Tim Berners-Lee** is the person who defined **HTML**. In 1990, **Tim Berners-Lee** was working at CERN (European Organization for Nuclear Research). He included the elements that could define title, paragraphs, hyperlink, headings, simple lists, address blocks and so on. But in that version of HTML there was no facility for producing tables or fill-in forms and images within a document.
- In 1994, **Tim Berners-Lee** launched the **World Wide Web Consortium (W3C)** for producing standards for web technologies. W3C also defined the standards for HTML. As an outcome of W3C's efforts HTML 2.0 adopted web standard within it.
- The W3C released its **HTML 4** recommendation in December 1997. The version 4.01 is then introduced.

- The meta language used to define the syntax for HTML 4.01 is SGML (Standard Generalized Markup Language).
- In 1998, the W3C introduced the EXtensible Markup Language (XML) which is a restricted version of SGML. Then a new version of HTML came up, which is called XHTML. (eXtensible HyperText Markup Language). The syntaxes of XHTML are defined using XML, rather than SGML.
- The HTML 5.0 is the fifth revision of HTML standard of World Wide Consortium(W3C) which is finalized in October 2014.

2.4 XHTML Syntax and Semantics

AU : Dec.-11, Marks 4

- An XHTML document consists of three main parts :

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

- The basic document structure is :

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

- The DOCTYPE declaration should be as given below -

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
```

- The DOCTYPE specifies the document Type. The Document type is specified by the Document Type Definition (DTD).
- The XHTML syntax rules are specified by the file **xhtml11.dtd** file. Hence we are specifying the name of this file at the beginning.

Three Flavors of XHTML :

There are three types of XHTML DTDs and those along with their uses are as given below -

1. **XHTML 1.0 Strict** : When we want a clean markup code then this type of dtd is used.
2. **XHTML 1.0 Transitional** : When we want to use some html features in the existing

XHTML document.

3. **XHTML 1.0 Frameset** : When want to make use of frames in the XHTML document.

University Question

1. Explain about XHTML DTD.

AU : Dec.-11, Marks 4

2.5 Some Fundamental HTML Elements

AU : Dec.-13, Marks 8

- In all HTML/XHTML documents the root element is `<html>` which has two children : head and body.
- Any text contained within the head element does not appear directly in the **client area** of web browser. The head element is used for providing certain information to web browser. The `<title>` element is used to give title to the web page. Note that this **title** does not appear in client area. Rather it appears at the top of web browser.
- The `<body>` element contains the information which is to be displayed in the client area of web browser.
- The body element can contain several fundamental elements such as `<p>`, `<h1>`, `<div>`, `<a>`, `<hr>`, `
` and so on. The elements within the body part is for deciding the layout of your web page.

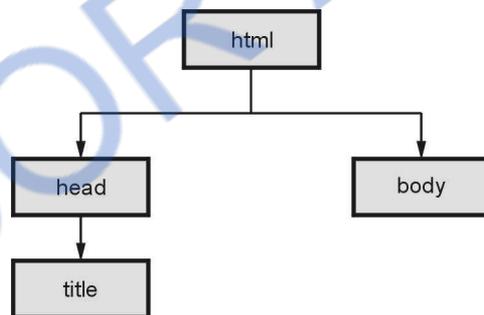


Fig. 2.5.1 Element tree

2.5.1 Heading

- There are header tags which help to display the text as some header.
- The header tag is denoted by h1, h2 and so on upto h6.
- Following HTML document along with its output is itself self explanatory.

HTML Document [headerDemo.html]

```

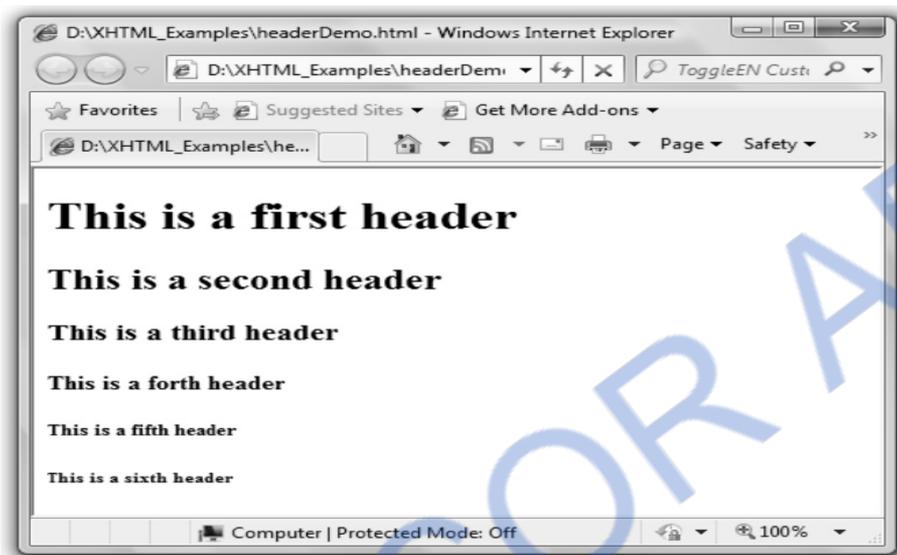
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h1> This is a first header </h1>
    <h2> This is a second header </h2>
    <h3> This is a third header </h3>
    <h4> This is a forth header </h4>
    <h5> This is a fifth header </h5>
  </body>
</html>
  
```

```

    <h6> This is a sixth header </h6>
  </body>
</html>

```

Output



2.5.2 Paragraphs

Following are the tags that are used for creating paragraphs

Tag	Meaning
<p>	This tag should be put at the end of every paragraph.
<pre>	This tag is used to preserve the white spaces and lines in the text.
<div>	This tag is used to make division of sections in the XHTML document.

For example

HTML Document[TextFormat.html]

```

<!DOCTYPE html>
<html >
  <head>
  </head>
  <body>
    <p>
      Once upon a time, there was a king who kept a monkey as a pet.
      The monkey served the king by all the possible ways.
    </p>
  </div>

```

```

It was very hot in those days. So one day, when the king was sleeping, monkey was
fanning the king. Suddenly monkey noticed a fly on the chest of the king. The monkey
tried to swish it away. But fly would go away for one moment and come back again.

```

```

</div>

```

```

<div> So monkey decided to teach a lesson to the fly.</div>

```

```

<pre>

```

```

The monkey then looked for something to hit a fly.

```

```

It then found a stick.

```

```

It picked up the stick and tried to beat the fly using stick.

```

```

Hmmm      at last it found a fly and

```

```

with all the force it hit the fly as a result of which the king died.

```

```

Lesson from the story:Never keep a fool servant !!!

```

```

</pre>

```

```

</body>

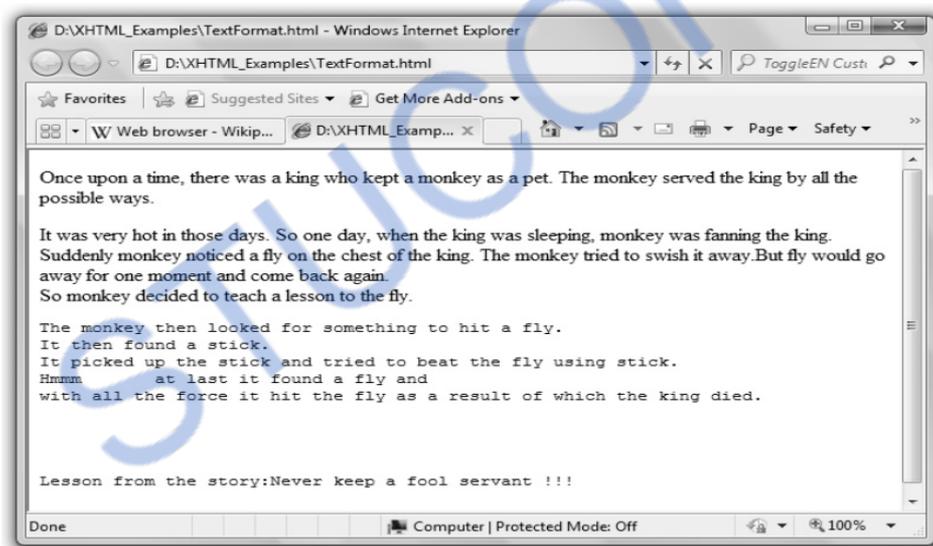
```

```

</html>

```

Output



2.5.3 Line Break

For the line break the **
** tag is used. This is a single line break tag. Generally this tag is kept at the end of every line.

For example

```

<!DOCTYPE html>

```

```

<html>

```

```

<head>

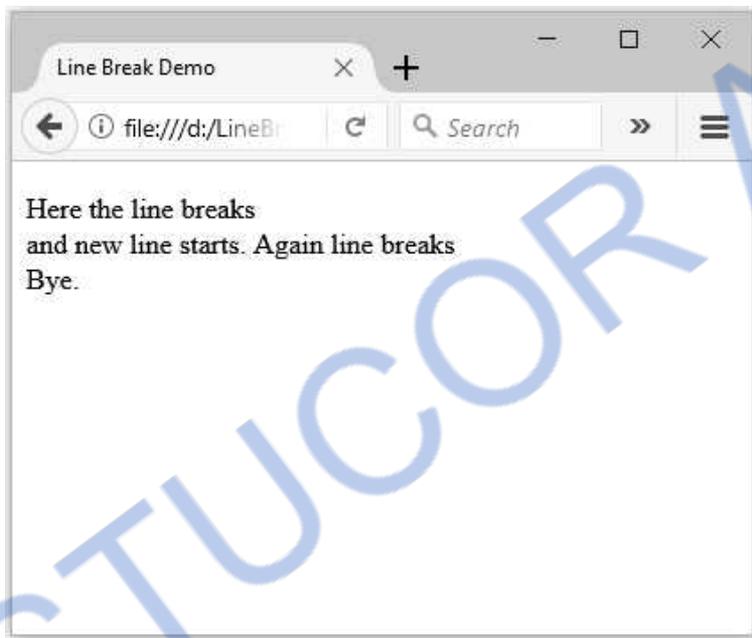
```

```

<title>Line Break Demo</title>

```

```
</head>
<body>
  <p>
    Here the line breaks <br/> and new line starts.
    Again line breaks<br/> Bye.
  </p>
</body>
</html>
```

Output**2.5.4 Setting Font Style**

The font style can be of various types such as boldface, italics, and strikethrough.

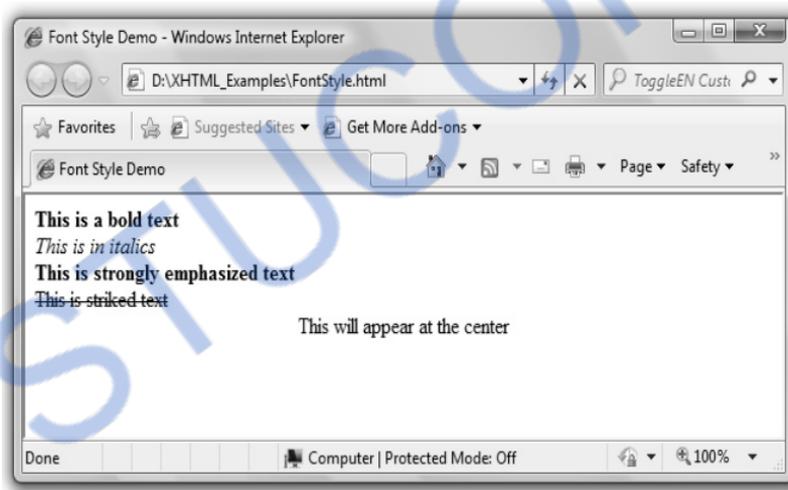
Tag	Purpose
 	Displays the text in bold
<i> </i>	Displays the text in italics
 	Displays the text in bold
<strike> </strike>	Displays the text with strike

Following HTML document illustrates this –

HTML Document [FontStyle.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Font Style Demo</title>
  </head>
  <body>
    <b> This is a bold text </b> <br/>
    <i> This is in italics</i> <br/>
    <strong> This is strongly emphasized text</strong> <br/>
    <del> This is striked text </del> <br/>
    <center> This will appear at the center</center> <br/>
  </body>
</html>
```

Output



2.5.5 Text Alignment

We can align the text at left, right or at the center using a **<div>** tag. Here is a HTML program which shows the text aligned left, right and centre.

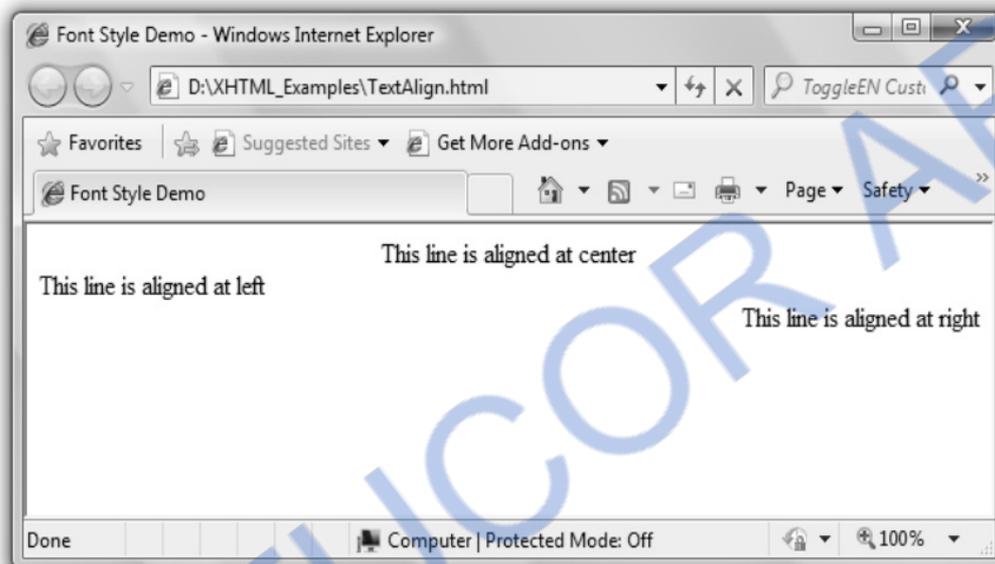
HTML Document[TextAlign.html]

```
<!DOCTYPE html >
<html>
  <head>
    <title>Font Style Demo</title>
  </head>
```

```

<body>
    <div align="center">This line is aligned at center</div>
    <div align="left">This line is aligned at left</div>
    <div align="right">This line is aligned at right</div>
</body>
</html>

```

Output**2.5.6 Setting the Font and Color**

We can set the font, size and color of the text in the web page. The tag **<basefont>** is used for this purpose. The optional attributes used with **<basefont>** tag are as given below :

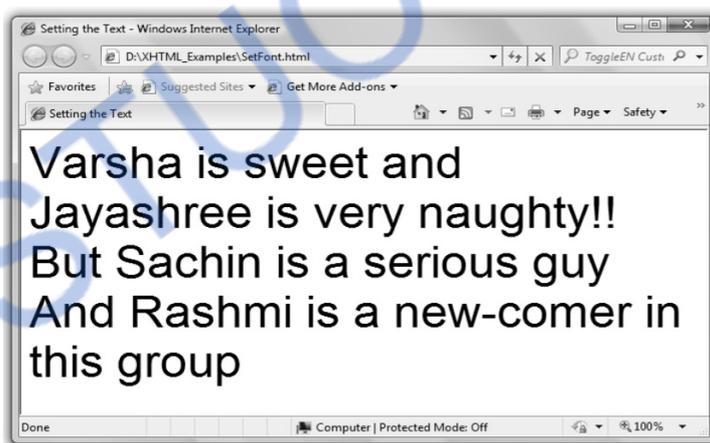
Attribute	Value	Purpose
color	Color value such as “red”, “yellow” and so on.	It displays the text with specified color.
face	Font-family such as “arial”, “verdana”, “sans-serif” and so on.	It displays the text with specific font family.
size	Number	The text of specified size will be displayed using this attribute value.

For example –

HTML Document[SetFont.html]

```
<!DOCTYPE html>
<html >
  <head>
    <title>Setting the Text </title>
  </head>
  <body>
    <basefont face="arial" size="10">
    Varsha is sweet and <br/>
    Jayashree is very naughty!!<br/>
    But Sachin is a serious guy<br/>
    And Rashmi is a new-comer in this group
  </body>
</html>
```

Output



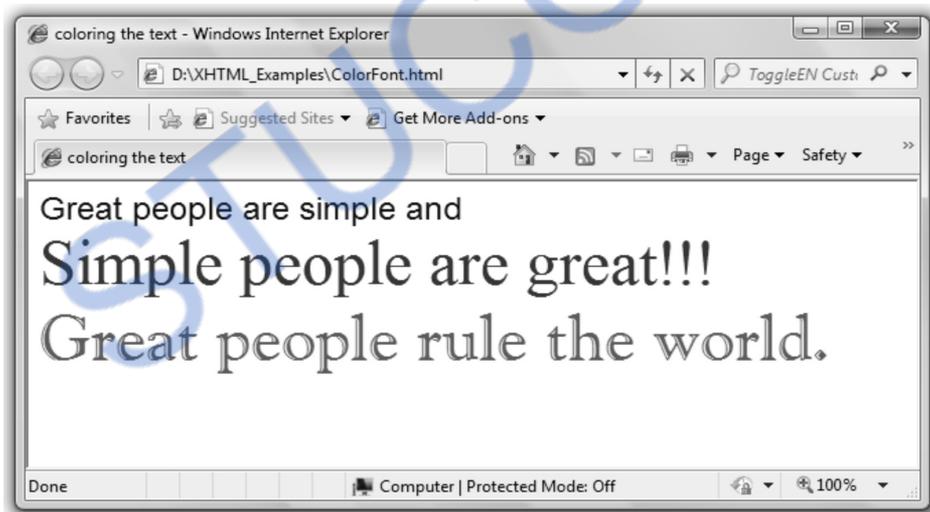
The **<basefont>** tag is supported in Internet Explorer 9 and not by FireFox, Chrome, Opera and so on.

Most of the web designers prefer to have their web pages white in color. HTML allows the web developer to use **color** attribute for coloring the text or for setting the background color.

First of all we will learn how to display a text colourful on the webpage.

HTML Document[ColorFont.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>coloring the text</title>
  </head>
  <body>
    <basefont face="arial" size="5" color="blue">
    Great people are simple and <br>
    <font face="Times new roman" color="red" size="8">
    Simple people are great!!!<br>
    <font face="GoudyHandtooled BT" color="green" size="14">
    Great people rule the world.
  </body>
</html>
```

Output**Script Explanation :**

- 1) In the above HTML document, we have used **font face** tag to set the font.
- 2) Using the attributes **color** and **size** we can specify values color and size in double quotes.

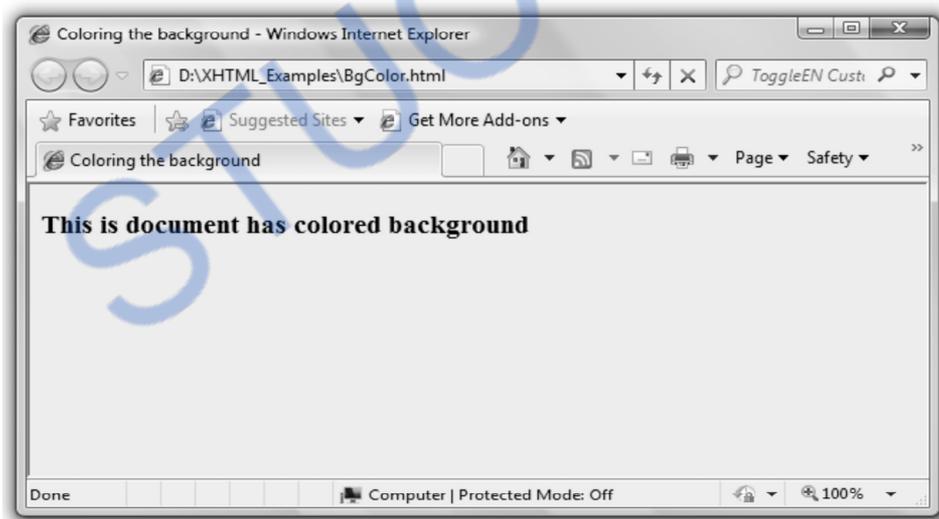
We can set the **background color** of the web page using the attribute **bgcolor**. Following program sets the background color of the web page to red.

HTML Document[BgColor.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Coloring the background</title>
  </head>
  <body bgcolor="#FFFF00">
    <h3>This is document has colored background</h3>
  </body>
</html>
```

Script Explanation :

- 1) In above document we have specified background color using the **bgcolor** attribute.
- 2) The color can be specified by either using the color name in double quotes or using the hex.code as given in above document.
- 3) The first two digits of Hex code represents the Red value, then next two digits specify the green value and final two digits specify the blue value. Hence the corresponding output will be as follows -

Output

Note that the background color is specified by the hexadecimal value. Following table shows the hex. and corresponding decimal values -

Decimal value	Hexadecimal value
1	1
2	2
3	3

4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

In HTML, the colors are specified by beginning with # and the FF denotes 255. In this color code first two digits specify the amount of red color then next two digits specify the green color and the last two digits specify blue color

FF 00 00
}
 red colour

Hence is used to set the background color red. Each pair of digits specifies 0 to 255 color values. Thus hexadecimal color coding helps us to specify.

$$256 * 256 * 256 = 16777216 \text{ colors.}$$

The color can also be specified by its name as follows -

```
<html>
<body bgcolor=yellow>
</html>
```

Following is a list of colors that can be specified for setting the background colors

Aqua	Navy
Black	Olive
Blue	Purple
Fuchsia	Red
Gray	Silver
Green	Teal
Lime	White
Maroon	Yellow

Method 2:

```
</h3>
```

```
<p>
```

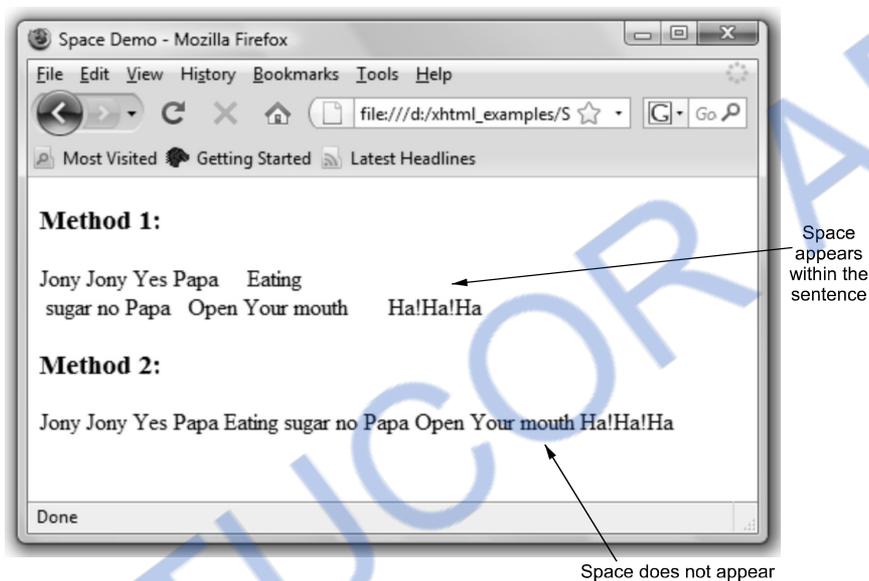
```
Jony Jony Yes Papa    Eating sugar no Papa    Open Your mouth    Ha!Ha!Ha
```

```
</p>
```

```
</body>
```

```
</html>
```

Output



Ex. 2.5.1 : How Divide and Pound symbol can be put on HTML document ?

Sol. : With the help of entity references denoted using & these two symbols can be put in HTML document. The code can be

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

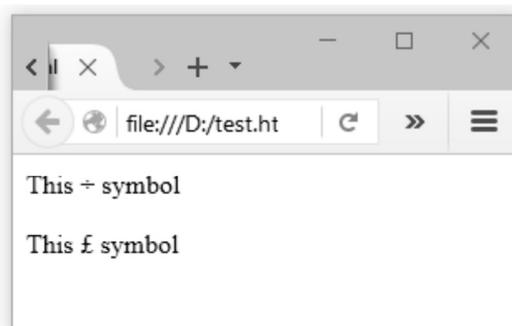
```
<p>This &divide symbol</p>
```

```
<p>This &pound symbol</p>
```

```
</body>
```

```
</html>
```

Output



University Question

1. State and explain any four HTML elements in detail.

AU : Dec.-13, Marks 8

2.6 Relative URLs

AU : Dec.-08, 09, Marks 2

- There is a common practice to specify the web link in the web page. The link acts as a pointer to some web page or some resource.
- Use of hyperlink in the web page allows that page to link logically with other page.
- We can use hyperlinks by using a tag `<a>` and by specifying the URL for **href**.
- The value assigned to **href** specifies the target of the link.
- The `<a>` means beginning of the web link and `` means end of the web link.
- The most important attribute of the `<a>` element is the **href** attribute, which indicates the link's destination.
- By default, links will appear as follows in all browsers :
 - 1) An unvisited link is underlined and blue.
 - 2) A visited link is underlined and purple
 - 3) An active link is underlined and red
- Following are the steps to be followed to specify web link in the web page.

Step 1 : The beginning of web link can be specified by the tag `<a href = “ “`. Inside the double quotes mention the URL of desired link.

Step 2 : Write some text that should act as a hyperlink.

Step 3 : End the web link ``

Here is a sample program which implements the above given idea -

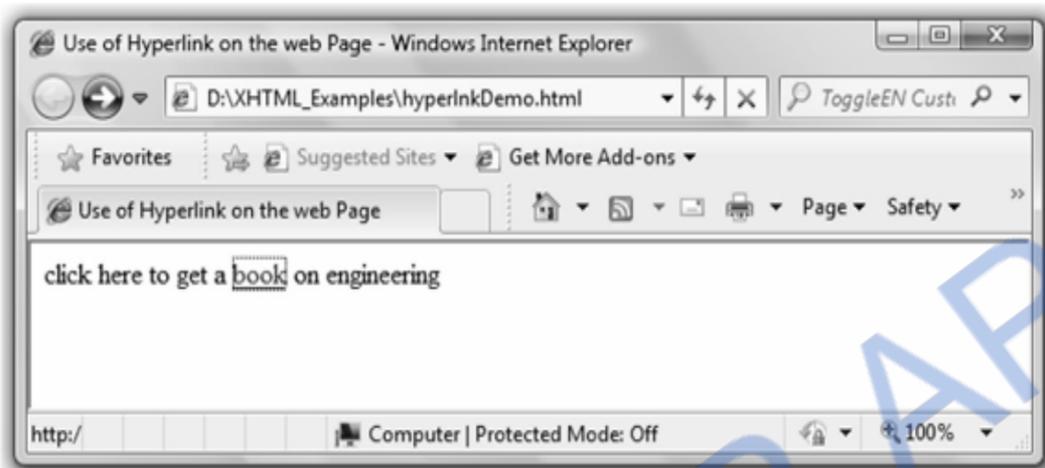
HTML Document [HyperInkDemo.html]

```

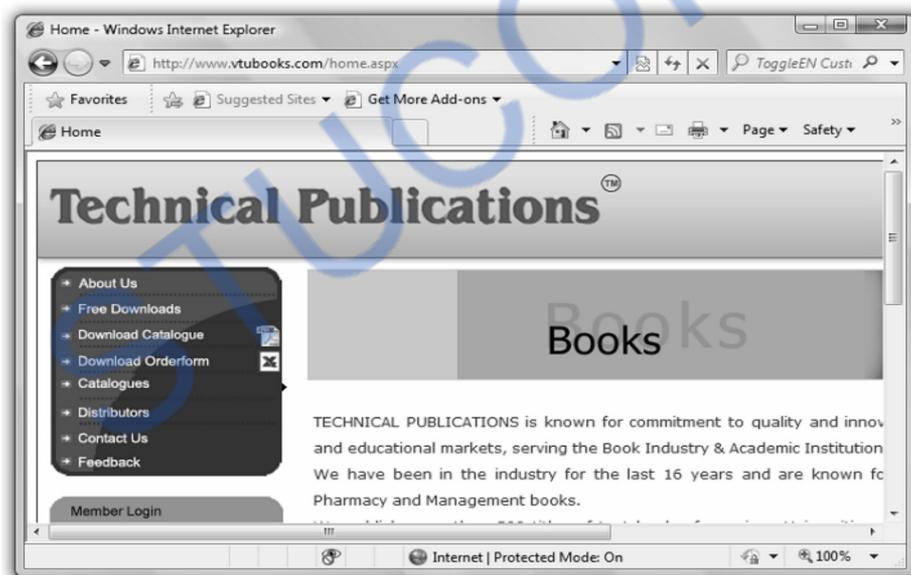
<!DOCTYPE html>
<html>
  <head>
    <title> Use of Hyperlink on the web Page </title>
  </head>
  <body>
    click here to get a
    <a href="http://www.vtubooks.com">book</a> on engineering
  </body>
</html>

```

Output



If you click on the hyperlink book then you will get the following output.



The target attribute

- If we want to get that link opened in another window we can mention **_target** property. Various targets can be -
- **_self** loads the page into the current window.
- **_blank** loads the page into a new separate browser window.

Ex.2.6.1 : Develop a HTML page to use image as a link to another page.

AU : Dec.-08, Marks 2

Sol. :

Step 1 : Write the HTML document containing an hyperlink to an image. When we click on the image the html page referred by the `<a href>` tag must get opened.

XHTML Document[ImgLink.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title> Image Demo </title>
  </head>
  <body>
    <h1> Indian Heritage </h1>
    <p>
      <a href="TajMahal.html" target = "_blank">
      <img src= "TajMahal.jpg" alt= "Taj Mahal!!" />
    </a>
    <br/>History of Taj Mahal
  </body>
</html>
```

Step 2 : Write the HTML document(TajMahal.html) mentioned by the `<a href>` tag in Step 1 as follows.

HTML Document[TajMahal.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title> Information on TajMahal </title>
  </head>
  <body>
    <h1> Indian Heritage </h1>
    <p>
      Taj Mahal is one of the wonders of the World. It is located in Agra,India.
    <br/>
      Taj Mahal is built by <strong>Shah Jahan</strong> in memory of his wife <strong>Mumtaz Mahal</strong>
    <br/>
      It is considered to be the excellent example of Mughal Architecture
    </p>
  </body>
</html>
```

Step 3 : Open some web browser and specify the file name **ImgLink.html** in the address bar and you will get following output.

Output

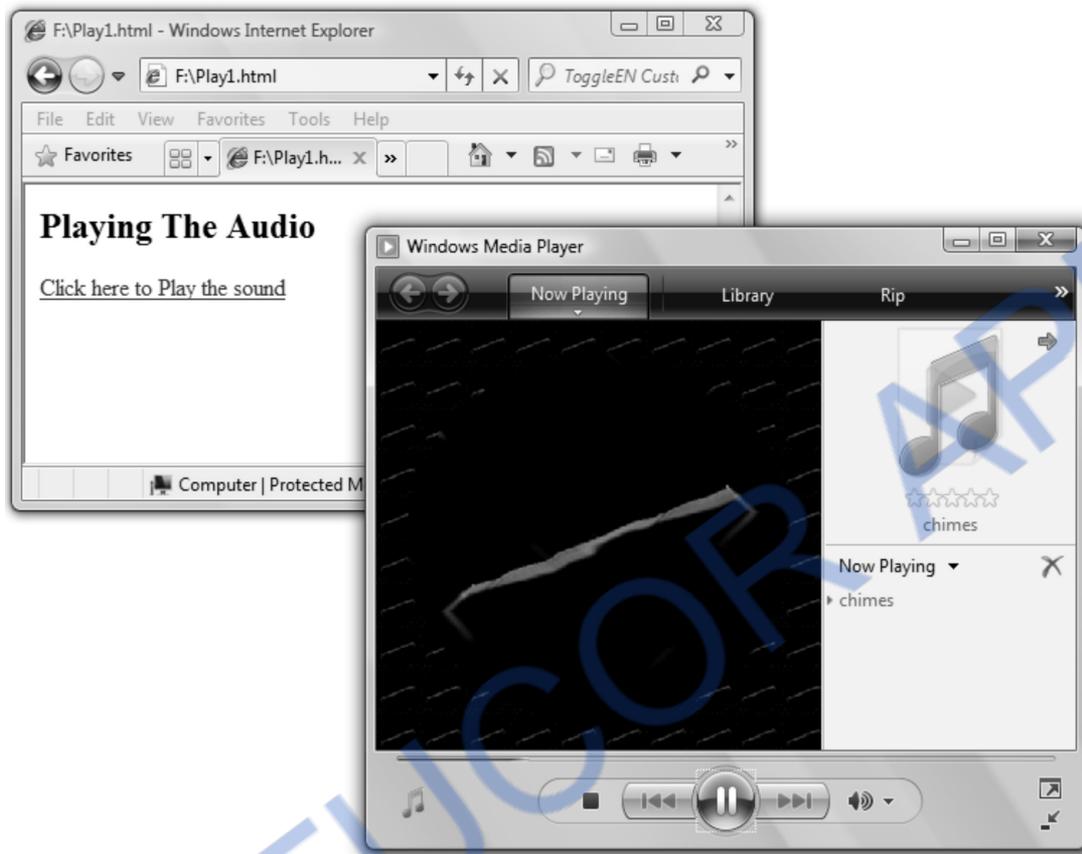


2.6.1 Plying an Audio File

Using HTML script we can play the audio files. The simplest way to play audio file is the use of hyperlinks. Following script shows how simple it is to a play a sound file using HTML

```
<html>
<body>
  <h2>Playing The Audio</h2>
  <form>
    <a href="chimes.wav">Click here to Play the sound</a>
  </form>
</body>
</html>
```

The above script executes in almost all the web browsers such as Internet explorer, Mozilla Firefox, Opera and so on. But the above HTML document launches the Media player software and then the audio file is played within it. For playing the audio files using hyperlinks it is necessary to get the audio file downloaded first.



2.6.2 Uses of Links

Following are the uses of the hyperlinks for a web document -

1. One can link logically related documents together using the links in the web page.
2. Use of link enhances the readability of the web document.
3. User can click on the link and can learn more about a subtopic and then can return to the main topic. This navigation within the web pages is possible due to the links.

University Question

1. How are audio files played through HTML? Explain with an example.

AU : Dec.-09, Marks 2

2.7 Lists

AU : Dec.-11, May-11, Marks 8

- List is nothing but the collections of items or elements. There are two types of lists - **unordered lists** and **ordered lists**.

- The unordered list is useful for simply listing the items but if we want the items in some specific sequence then the ordered lists are used. Let us discuss how to use these types lists in the HTML document.

2.7.1 Unordered List

Following HTML document makes use of unordered list.

HTML Document [UnordLstDemo.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title> Use of Unordered List </title>
  </head>
  <body>
    <h2>All About Computer ...</h2>
```

Following are some popular operating systems used in computer

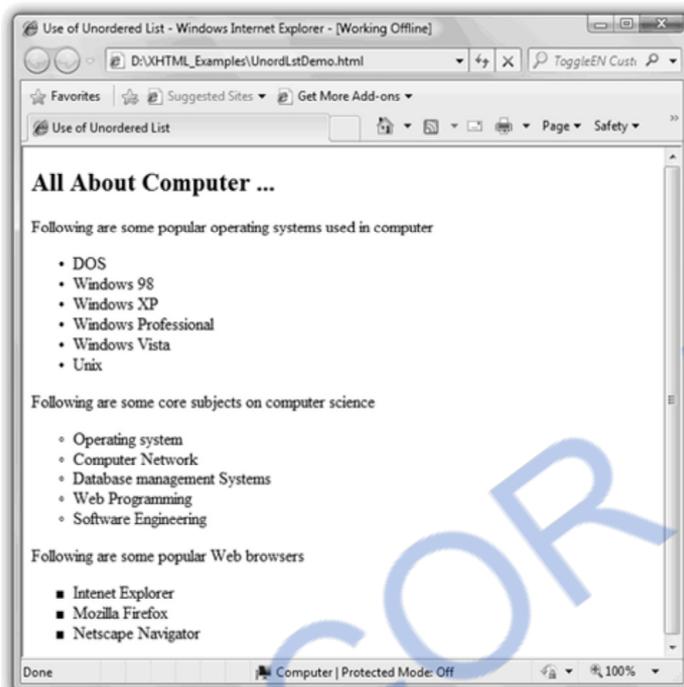
```
<ul type="disc">
  <li>DOS</li>
  <li>Windows 98</li>
  <li>Windows XP</li>
  <li>Windows Professional</li>
  <li>Windows Vista</li>
  <li>Unix</li>
</ul>
```

Following are some core subjects on computer science

```
<ul type="circle">
  <li>Operating system</li>
  <li>Computer Network</li>
  <li>Database management Systems</li>
  <li>Web Programming</li>
  <li>Software Engineering</li>
</ul>
```

Following are some popular Web browsers

```
<ul type="square">
  <li>Intenet Explorer</li>
  <li>Mozilla Firefox</li>
  <li>Netscape Navigator</li>
</ul>
</body>
</html>
```

Output**Script Explanation :**

In above script we have used the tag `` for specifying the unordered list. The list items can be mentioned using the tag `` Due to which the list items can appear in bulleted form. The style of the bullet can be as given below -

tags	meaning
<code></code> and <code></code>	Beginning and end of the bulleted list
<code></code> and <code></code>	Displays the bulleted text on separate line.
<code><ul type="circle"></code>	Displays the circular bullets.
<code><ul type="disc"></code>	Displays the solid round bullets.
<code><ul type="square"></code>	Displays the squared bullets.

2.7.2 Ordered List

The ordered list is a list of items which must follows some specific sequence. We can number the text using `` tag. Following table shows various styles by which the list can be numbered.

tags	meaning
<code></code> and <code></code>	Beginning and end of the numbered list
<code></code> and <code></code>	Displays the numbered text on separate line.
<code><ol type="A"></code>	Displays the list in following manner
	A.
	B.
	C.
	...
<code><ol type="I"></code>	Displays the list in following manner
	i.
	ii.
	iii.
	...
<code><ol type="I"></code>	Displays the list in following manner
	I.
	II.
	III.
	...
<code><ol type="1"></code>	Displays the list in following manner
	1.
	2.
	3.
	...

Here is one illustrative program -

HTML Document [OrdLstDemo.html]

```

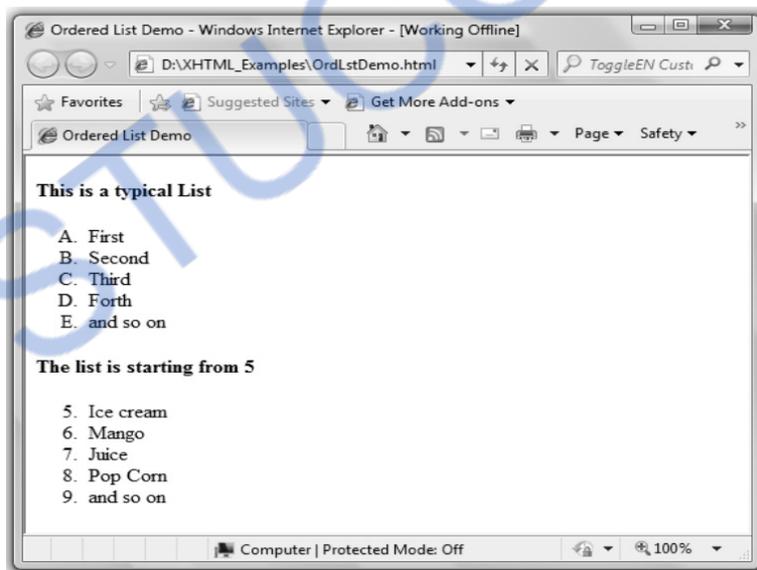
<!DOCTYPE html>
<html>
  <head>
    <title> Ordered List Demo </title>
  </head>
  <body>

```

```
<h4>This is a typical List</h4>
<ol type="A">
<li>First</li>
<li>Second</li>
<li>Third</li>
<li>Forth</li>
<li> and so on</li>
</ol>
<h4>The list is starting from 5</h4>
<ol start="5">
<li>Ice cream</li>
<li>Mango</li>
<li>Juice</li>
<li>Pop Corn</li>
<li> and so on</li>
</ol>
</body>
</html>
```

We can have the nested ordered list. Here is an example of it

Output

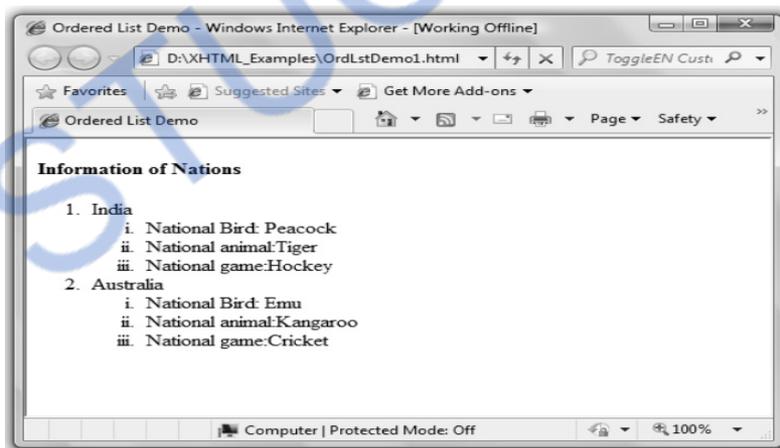


HTML Document [OrdLstDemo1.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title> Ordered List Demo </title>
  </head>
```

```
<body>
  <h4>Information of Nations</h4>
  <ol type="1">
    <li>India
      <ol type="i">
        <li>National Bird: Peacock</li>
        <li>National animal:Tiger</li>
        <li>National game:Hockey</li>
      </ol>
    </li>
    <li>Australia
      <ol type="i">
        <li>National Bird: Emu</li>
        <li>National animal:Kangaroo</li>
        <li>National game:Cricket</li>
      </ol>
    </li>
  </ol>
</body>
</html>
```

Output



University Questions

1. Explain the purpose and way of creating lists in HTML documents.

AU :Dec.-11, Marks 8

2. State the types of lists supported by HTML and explain them in detail.

AU : May-11, Marks 8

2.8 Tables**AU : Dec.-13, May-12, Marks 8**

- For systematic arrangement of information we often require tabular structure. HTML allows us to create tables on the web pages.
- Many web designers are using invisible tables on the web pages.
- The **biggest advantage of using tables** on the web pages is that the information gets arranged systematically.
- The table is a matrix of rows and columns and the area formed due to intersection of a row and a column is called cell.

2.8.1 Basic Table Tag

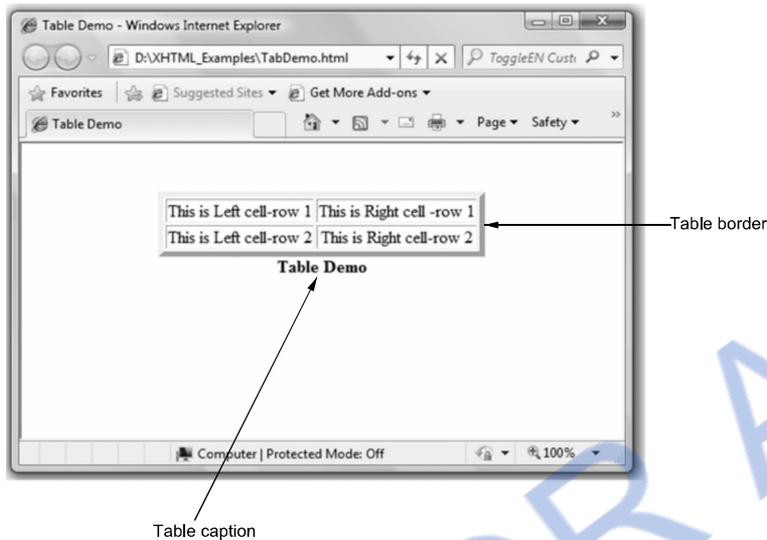
- To create a table on the web page the table beginning tag is `<table>` and `</table>` tag is used for ending the table.
- Within `<table> ... </table>` tag we can create rows and columns.
- The rows are created using `<tr> </tr>` and columns are created using `<td> </td>`
- **Example**

HTML Document[TabDemo.html]

```

<!DOCTYPE html>
<html>
  <head>
    <title> Table Demo </title>
  </head>
  <body>
    <br/><br/>
    <center>
      <table border="5">
        <caption align="bottom">
          <b> Table Demo </b>
        </caption>
        <tr>
          <td>This is Left cell-row 1</td>
          <td>This is Right cell -row 1</td>
        </tr>
        <tr>
          <td>This is Left cell-row 2</td>
          <td>This is Right cell-row 2</td>
        </tr>
      </table>
    </center>
  </body>
</html>

```

Output**Script Explanation**

- 1) In above program, the parameter border="5" is set in order to set the table border. You can give any value to set the desired border.
- 2) The caption to the table is given by the parameter caption, we can set this caption either at the top or at the bottom by using the parameter align.
- 3) Then using the tag <tr> the table row can be set. The <td> tag is used to create columns from left to right.
- 4) Thus in the above program we are filling up the table values from top to bottom and from left to right. Just refer the output provided along with the program for clear understanding of look and feel of the table.

Ex. 2.8.1: Write HTML document to create a table with header to each column.

Sol. : We can set the **header** to each column of the table by <th> tag. Here is a simple HTML program that simply adds the header to each column.

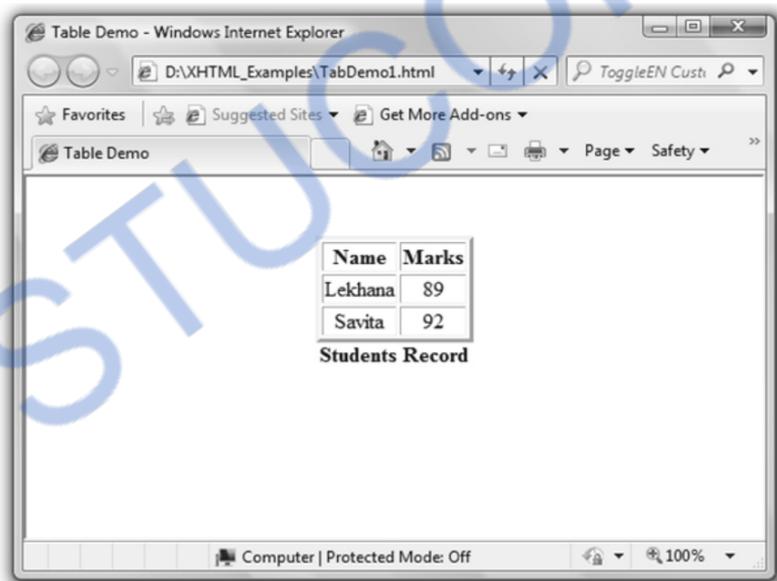
HTML Document [TabDemo1.html]

```
<!DOCTYPE html >
<html >
  <head>
    <title> Table Demo </title>
  </head>
  <body>
    <br/><br/>
    <center>
```

```

<table border="3">
<caption align="bottom">
  <b> Students Record </b>
</caption>
  <th>Name</th>
  <th>Marks</th>
  <tr align=center>
    <td>Lekhana</td>
    <td>89</td>
  </tr>
  <tr align=center>
    <td>Savita</td>
    <td>92</td></tr>
  </tr>
</table>
</center>
</html>

```

Output

The data in each row can be aligned centrally by using `<tr align=center>` Other types of alignment can be left or right by `<tr align=left>` or `<tr align=right>` respectively.

Ex. 2.8.2 : Discuss the process of dividing the table into columns with an example. How would you give headers to the table columns ?

Sol. : The process of dividing the table into column is as follows –

1. Create a table using the tag `<table>`

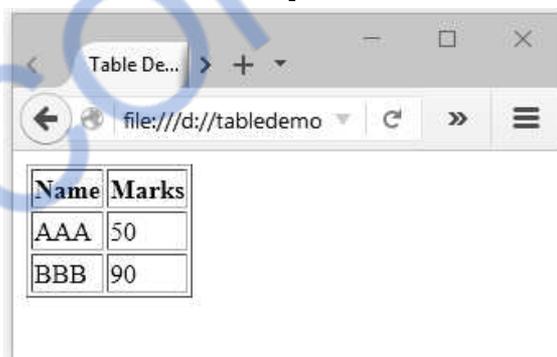
2. Set the border to the table using attribute border="some value"
3. Create rows using the tag <tr> and for each row create the columns using <td> </td> tags.
4. The header for the table column can be given using the tag <th> </th>

Ex. 2.8.3 : Following HTML document divides the table in columns and gives the header to the column.

Sol. :

```
<!DOCTYPE html>
<html>
<head>
<title>Table Demo</title>
</head>
<body>
  <table border ="1">
    <th>Name</th>
    <th>Marks</th>
    <tr>
      <td>AAA</td>
      <td>50</td>
    </tr>
    <tr>
      <td>BBB</td>
      <td>90</td>
    </tr>
  </table>
</body>
</html>
```

Output



2.8.2 Setting Background to Table

We can decorate our tables by setting the background of the table. Either some color can be set as table background or some image can be set as a table background. Here is an illustration -

HTML Document[TabDemoBack.html]

```
<!DOCTYPE html >
<html>
  <head>
    <title> Table Decoration </title>
  </head>
  <body>
    <table border="2">
      <caption align="top">
        <b>Background color</b>
      </caption>
```

```

<th bgcolor=yellow>Name</th>
<th bgcolor=yellow>Marks</th>
<tr align=center>
  <td bgcolor=red>A</td>
  <td bgcolor=green>92</td>
</tr>
<tr align=center>
  <td bgcolor=green>B</td>
  <td bgcolor=red>90</td>
</tr>
</table>

```

Setting the background color to each cell

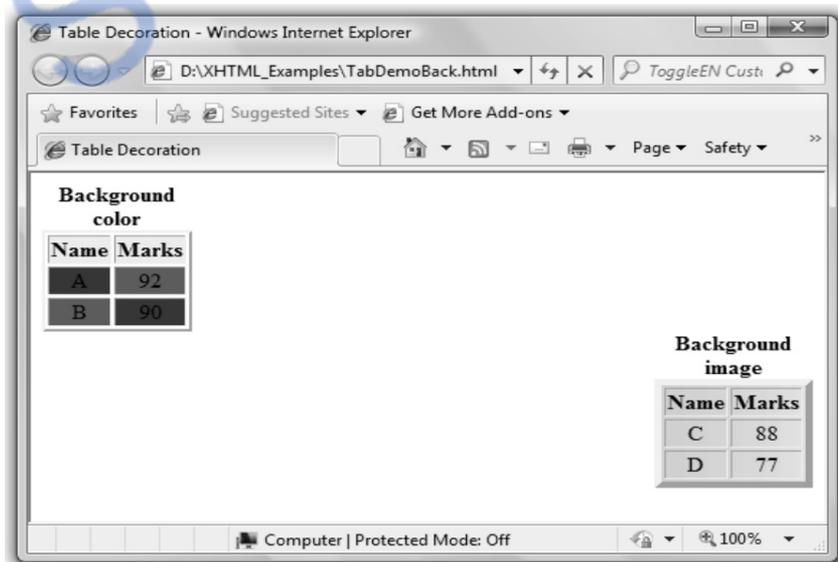
```

<table border="5" background="backgr1.jpg" align="right">
  <caption align="top">
    <b>Background image</b>
  </caption>
  <th>Name</th>
  <th>Marks</th>
  <tr align=center>
    <td>C</td>
    <td>88</td>
  </tr>
  <tr align=center>
    <td>D</td>
    <td>77</td>
  </tr>
</table>
</html>

```

Setting the background image

Output



2.8.3 Rowspan and Colspan

Sometimes we may require adding sub-columns or sub-rows to categorize the information properly. In such a situation colspan and rowspan can be used.

The rowspan is used to extend the row vertically and colspan is used to extend the column horizontally. For example -

When **rowspan=2** then the **row can be extended vertically by two cells.**

rowspan	First	Second	<td rowspan=2>First
		Third	<td>Second</td> <td>Third</td>

When **colspan=2** then the **column can be extended horizontally by two cells.**

First	Second	<td>First <td>Second
Third		<td colspan=2>Third

colspan

Ex. 2.8.4 : Explain the structure of HTML documents, write code to display following table :

Sr. No.	College Name			
	Section A		Section B	
	X	Y	X	Y
1				
2				
3				

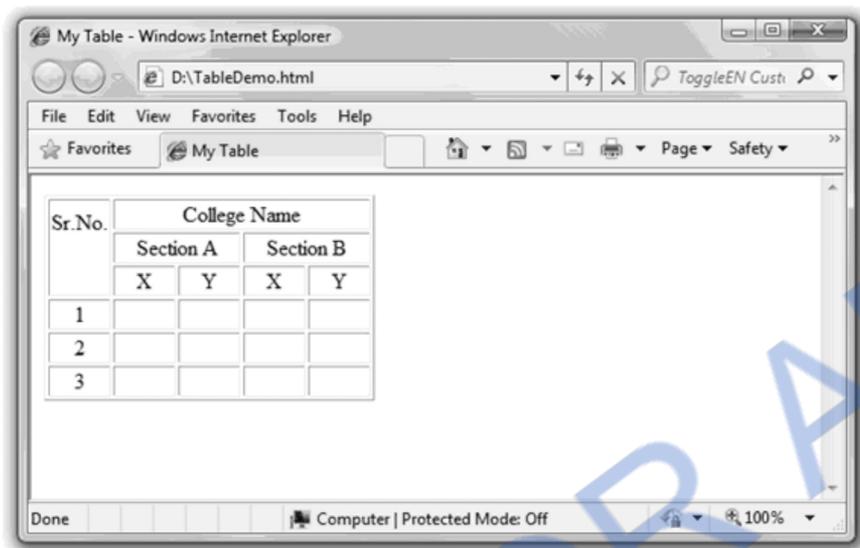
Sol. :

```
<html>
  <head>
    <title>My Table</title>
  </head>
  <body>
    <table border=1>
      <tr>
        <td rowspan="3">
          <p align=center>Sr.No.</p>
          <p>&nbsp;</td>
        </td>
```



```
<td colspan="4">
    <p align="center">College Name</p>
</td>
</tr>
<tr>
<td width="40%" colspan="2" align="center">Section A</td>
<td width="40%" colspan="2" align="center">Section B</td>
</tr>
<tr>
    <td align="center">X</td>
    <td align="center">Y</td>
    <td align="center">X</td>
    <td align="center">Y</td>
</tr>
<tr>
    <td align="center">1</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
</tr>
<tr>
    <td align="center">2</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
</tr>
<tr>
    <td align="center">3</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
    <td align="center">&nbsp;</td>
</tr>
</table>
</body>
</html>
```

Output



Ex. 2.8.5 : Write HTML code to draw table given below :

Image (20%)	Company Name (80%)
	Schedule
1	Type 1
2	
3	Type 2
4	

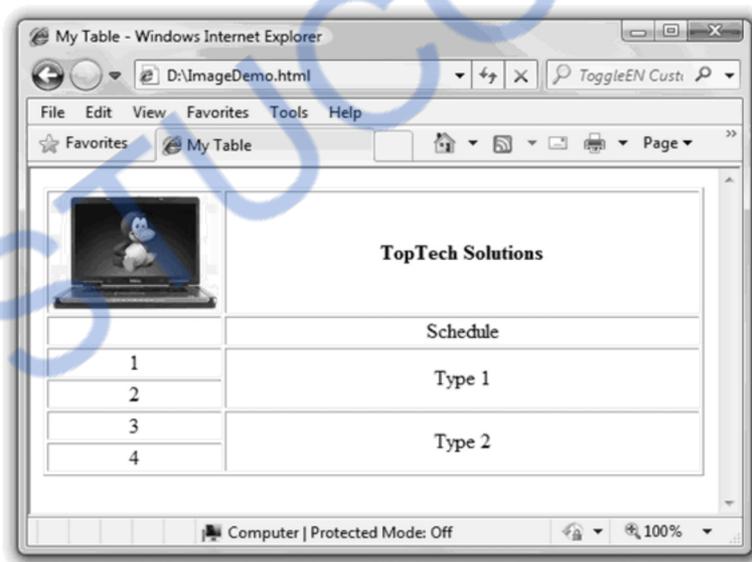
Sol. : ImageDemo.html

```
<html>
  <head>
    <title>My Table</title>
  </head>
  <body>
    <table border=1>
      <tr>
        <td width="20%" align=center></td>
        <th width="80%" align=center>TopTech Solutions</th>
      </tr>
      <tr>
        <td>&nbsp;</td>
        <td align=center>Schedule</td>
      </tr>
```

```

<tr>
  <td align=center>1</td>
  <td rowspan="2" align=center>Type 1</td>
</tr>
<tr>
  <td align=center>2</td>
</tr>
<tr>
  <td align=center>3</td>
  <td rowspan="2" align=center>Type 2</td>
</tr>
<tr>
  <td align=center>4</td>
</tr>
</table>
</body>
</html>

```

Output

Ex. 2.8.6 : Write HTML code to draw table given below.

Items		Price
Shirt	Trouser	Rs. 1000/-
Rs. 400/-	Rs. 600/-	

Sol. :

```
<html>
<head>
<title>Item-Price List</title>
</head>
<body>
<table border=1>
<tr>
<th colspan="2"><center>Items</center></th>
<th>Price</th>
</tr>
<tr>
<th><center>Shirt</center></th>
th><center>Trouser</center></th>
<td rowspan="2"><center>Rs.<br/>1000/-</center></td>
<tr>
<td><center>Rs.400/-</td>
<td><center>Rs.600/-</td>
</tr>
</table>
</body>
</html>
```

Output



Ex. 2.8.7 : Write HTML code which includes table, Hyperlink, character formatting ordered and unordered list to display your resume.

Sol. : resume.html

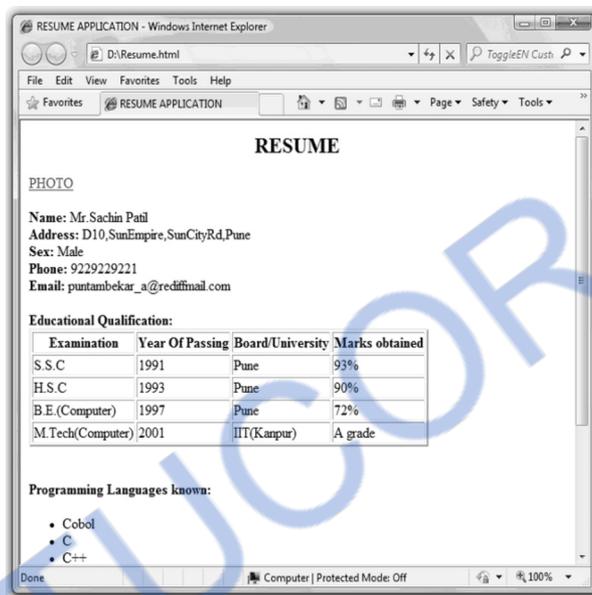
```
<html>
<head>
<title>RESUME APPLICATION</title>
```

```
</head>
<body>
<center><h2>RESUME</h2></center>
<a href="my_photo.jpg">PHOTO</a>
<br/><br/>
<strong>Name: </strong> Mr.Sachin Patil<br/>
<strong>Address: </strong> D10,SunEmpire,SunCityRd,Pune<br/>
<strong>Sex: </strong>Male<br/>
<strong>Phone: </strong>9229229221<br/>
<strong>Email: </strong>puntambekar_a@rediffmail.com
<br/><br/>
<strong>Educational Qualification:</strong>
<table border=2>
<tr>
<th>Examination</th>
<th>Year Of Passing</th>
<th>Board/University</th>
<th>Marks obtained</th>
</tr>
<tr>
<td>S.S.C</td><td>1991</td><td>Pune</td><td>93%</td>
</tr>
<tr>
<td>H.S.C</td><td>1993</td><td>Pune</td><td>90%</td>
</tr>
<tr>
<td>B.E.(Computer)</td><td>1997</td><td>Pune</td><td>72%</td>
</tr>
<tr>
<td>M.Tech(Computer)</td><td>2001</td><td>IIT(Kanpur)</td><td>A grade</td>
</tr>
</table>
<strong>
<br/><br/>
Programming Languages known:
</strong>
<ul>
<li>Cobol
<li>C
<li>C++
<li>Visual C++
<li>Visual Basic
<li>Java
</ul>
```

```

<br/><br/>
<div align=left>Date:</div>
<div align=right>Place:</div>
<br/>
<div align=left>Signature</div>
</body>
</html>

```

Output**Ex. 2.8.8 : Create the following table using HTML tags.**

Name of Train	Place	Destination	Time Arrival	Departure	Fare
Rajdhani	Mumbai	Delhi	07.30	08.45	Rs. 989.00
Madras Mail	Mumbai	Madras	09.00	10.15	Rs. 450.00
Konkan Exp.	Mumbai	Manglore	13.30	14.45	Rs. 756.00
Deccan Exp.	Mumbai	Pune	16.00	17.30	Rs. 345.00

Sol. :

```

<html>
<head>
  <title> Train Time Table </title>
</head>
<body>
  <table border="1">
    <tr>
      <th rowspan=2>Name of Train</th>
      <th rowspan=2>Place</th>

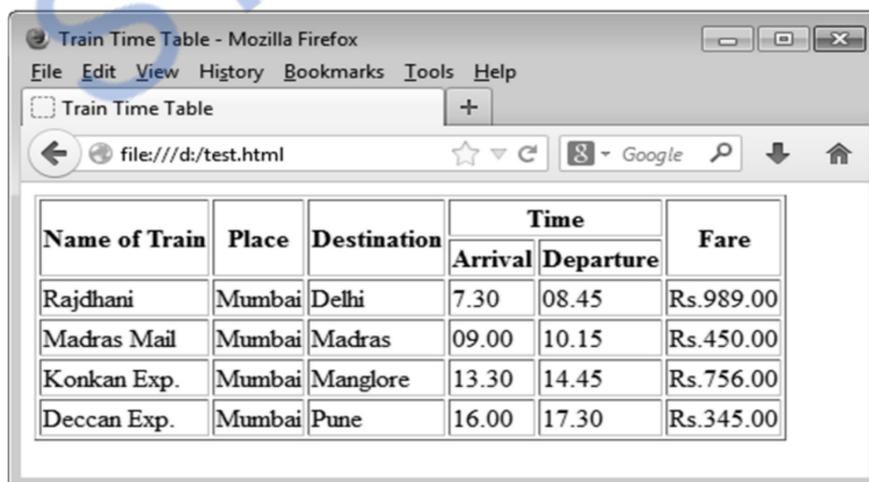
```

```

<th rowspan=2>Destination</th>
<th colspan=2>Time</th>
<th rowspan=2>Fare</th>
</tr>
<tr><th>Arrival</th><th>Departure</th></tr>
<tr>
<td>Rajdhani</td><td>Mumbai</td><td>Delhi</td>
<td>7.30</td><td>08.45</td>
<td>Rs.989.00</td>
</tr>
<tr>
<td>Madras Mail</td><td>Mumbai</td><td>Madras</td>
<td>09.00</td><td>10.15</td>
<td>Rs.450.00</td>
</tr>
<tr>
<td>Konkan Exp.</td><td>Mumbai</td><td>Manglore</td>
<td>13.30</td><td>14.45</td>
<td>Rs.756.00</td>
</tr>
<tr>
<td>Deccan Exp.</td><td>Mumbai</td><td>Pune</td>
<td>16.00</td><td>17.30</td>
<td>Rs.345.00</td>
</tr>
</table>
</body>
</html>

```

Output



Name of Train	Place	Destination	Time		Fare
			Arrival	Departure	
Rajdhani	Mumbai	Delhi	7.30	08.45	Rs.989.00
Madras Mail	Mumbai	Madras	09.00	10.15	Rs.450.00
Konkan Exp.	Mumbai	Manglore	13.30	14.45	Rs.756.00
Deccan Exp.	Mumbai	Pune	16.00	17.30	Rs.345.00

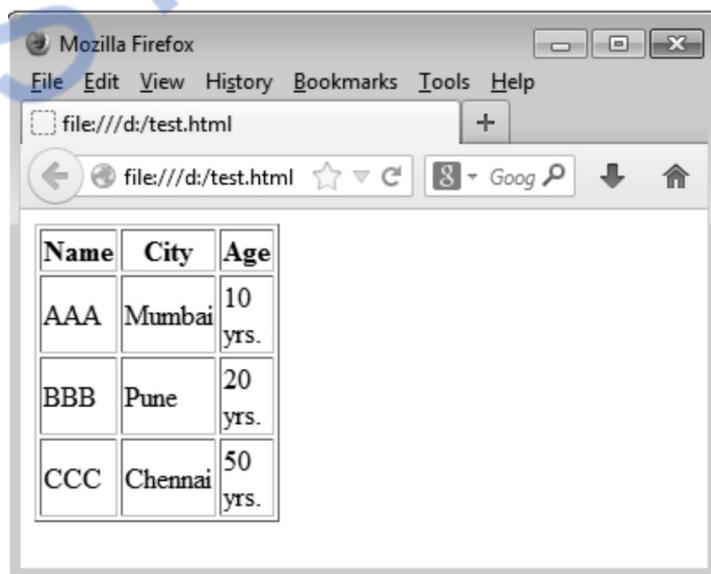
Ex. 2.8.9 : Differentiate between `<table width="400" height="200">` and `<table width="100%" and height="50%">`

Sol. : (i) `<table width="400" height="200">` denotes that the table width is 400 pixels and height is 200 pixels. It denotes the **absolute value**

For example

```
<html>
<head>
</head>
<body>
  <table border="1" width="100" height="50">
    <tr>
      <th>Name</th>
      <th>City</th>
      <th>Age</th>
    </tr>
    <tr>
      <td>AAA</td><td>Mumbai</td><td>10 yrs.</td>
    </tr>
    <tr>
      <td>BBB</td><td>Pune</td><td>20 yrs.</td>
    </tr>
    <tr>
      <td>CCC</td><td>Chennai</td><td>50 yrs.</td>
    </tr>
  </table>
</body>
</html>
```

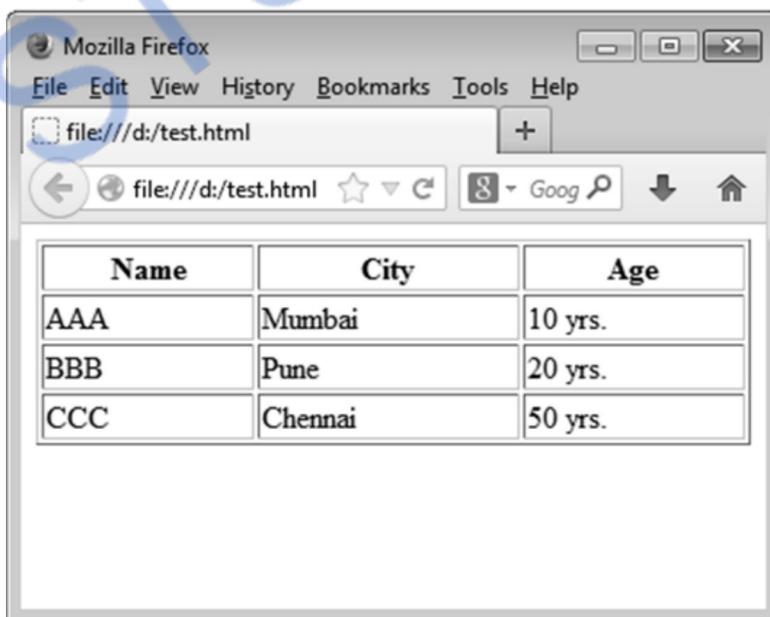
Output



- ii) On the other hand `<table width="100%" and height="50%">` denotes the width of the table is 100% of the screen and height is 50% of the browser screen. It denotes the **relative value**

For example

```
<html>
<head>
</head>
<body>
  <table border="1" width="100%" height="50%">
    <tr>
      <th>Name</th>
      <th>City</th>
      <th>Age</th>
    </tr>
    <tr>
      <td>AAA</td><td>Mumbai</td><td>10 yrs.</td>
    </tr>
      <tr>
        <td>BBB</td><td>Pune</td><td>20 yrs.</td>
      </tr>
      <tr>
        <td>CCC</td><td>Chennai</td><td>50 yrs.</td>
      </tr>
    </table>
</body>
</html>
```

Output

Review Questions

1. Explain how tables can be inserted into HTML document with an example.

AU : May-12, Marks 8

2. Explain the way in which data can be presented in a tabular form using HTML.

AU : Dec.-13, Marks 8

2.9 Frames

AU : May-13, 14, Dec.-09, Marks 16

- HTML frames allow us to present documents in **multiple views**.
- Using multiple views we can keep certain information visible and at the same time other views are scrolled or replaced.
- For example, within the same window, one frame can display a company information, a second frame can be a navigation menu and a third frame may display selected document that can be scrolled through or replaced by navigating in the second frame.
- Various frames can be set in one browser window .
- To set the frames in the browser window we use **frame set**.

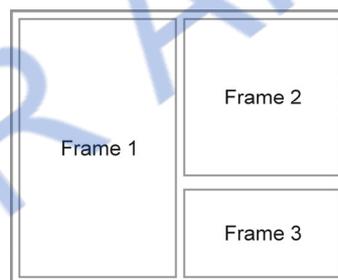


Fig. 2.9.1 Frames

- **For example -**

```
<frameset cols="150,*">
```

will allow us to divide the window into two columns (i.e. in two vertical frames). One frame occupying the size of 150 pixels and the other occupies the remaining portion of the window. Here * means any number of pixels.

- Similarly

```
<frameset rows="*,120">
```

will divide the window into two rows (i.e. in two horizontal frames). The second part of horizontal frame will be of 120 pixels and upper horizontal frame will occupy remaining portion of the window.

- Similarly we can also specify the frameset in percentage form. For example

```
<frameset rows="30%,70%">
```

Using frameset we can divide the rows and columns in any number of frames. For example

```
<frameset rows = "20%,30%,50%" cols = "30%,*">
```

This will create a frames in the browser's window as follows -



Fig. 2.9.2 Frames

- In every layout frame we can load the desired html page by using **frame src**. For example
`<frame src="D:\\html_examples\\bulleted1.html" name="Left_Vertical">`

By this statement we are loading the web page **bulleted1.html** in the specific frame and the frame is named as **Left_Vertical**.

Attributes in frameset tag

Attribute	Value	Purpose
cols	pixels % *	It specifies the number and size of columns in a frameset
rows	pixels % *	It specifies the number and size of rows in a frameset.

Attributes of frame tag

The `<frame>` tag has no end tag. The `<frame>` tag defines one frame within a `<frameset>` tag. Various attributes of frame tag are

Attribute	Value	Purpose
frameborder	0 or 1	Value 1 specifies that the frame is displayed with border and 0 indicates that there is no border.
name	Some name	It specifies name of the frame
Nosize		Due to this attribute we cannot resize the particular frame.
scrolling	yes, no or auto	It specifies whether or not to display the scrollbar along with the frame.
src	URL	It specifies the name of the Document to be displayed within the frame.

Example of Browser containing frame

Step 1 : Create a main HTML document which will display three HTML documents in three vertical frames

FrameSet.html

```
<!DOCTYPE html>
<html>
<frameset cols="25%,*,50%">
  <frame src="frame1.html">
  <frame src="frame2.html">
  <frame src="frame3.html">
</frameset>
</html>
```

Step 2 : Create frame1.html, frame2.html and frame3.html files as follows –

Frame1.html

```
<!DOCTYPE html>
<html>
<body>
  <h1> Frame 1 </h1>
</body>
</html>
```

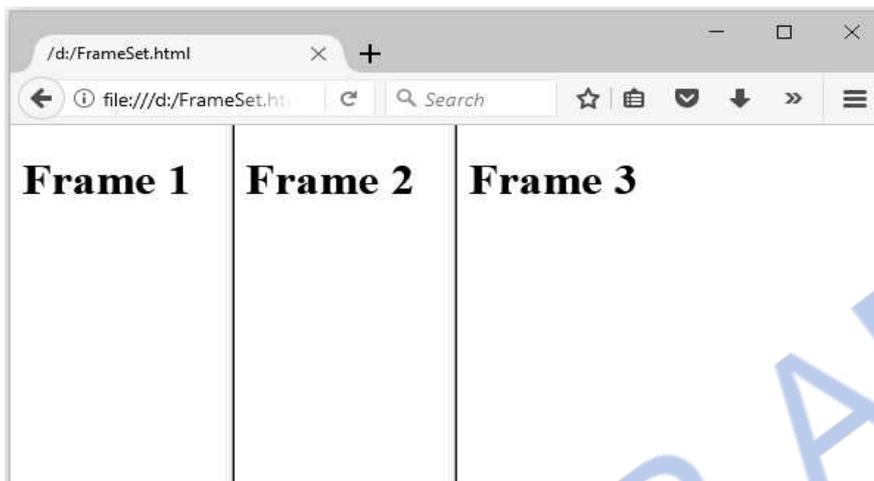
Frame2.html

```
<!DOCTYPE html>
<html>
<body>
  <h1> Frame 2 </h1>
</body>
</html>
```

Frame3.html

```
<!DOCTYPE html>
<html>
<body>
  <h1> Frame 3 </h1>
</body>
</html>
```

Output



Script Explanation :

The above HTML document, the `<frameset>` tag is used to define frameset. The `col` attribute is used to create the three column frames.

2.9.1 Frames with Scrollbars

```
<!DOCTYPE html>
<html>
<frameset cols="25%,*,50%">
  <frame src="frame1.html" scrolling=no>
  <frame src="frame2.html" scrolling=auto>
  <frame src="frame3.html">
</frameset>
</html>
```

Output



To set the scroll bar we should assign the value **auto** to the **scrolling** parameter and if we do not want the scroll bar accompanying the frame then we must assign the value **no** to the scrolling parameter.

Ex. 2.9.1 : Create XHTML document that has two frames. The left frame displays the contents.html and the right frame displays the cars.html where second frame is the target of link from the first frame. [Note : Contents.html is a list of links for the cars description]

Sol. :

Step 1 : We will create the basic html file that contains the left and right frames. The HTML document is as follows :

CarDemo.html

```
<!DOCTYPE html >
<html>
  <head>
    <title>My Frames Page</title>
  </head>
  <frameset cols="50%,50%">
    <frame src="contents.html" name="Left_Vertical" noresize />
    <frame src="Description.html" name="Right_Vertical" scrolling=no/>
  </frameset>
</html>
```

Step 2 : Following is a HTML document that displays the contents that are loaded at first. The left frame HTML document is as follows –

contents.html

```
<!DOCTYPE html >
<html >
  <head>
    <title>My Frames Page</title>
  </head>
  <body>
    <h3>
      <a href="Innova.html" target="Right_Vertical">Toyota Innova</a>
    </h3>
    <h3>
      <a href="Scorpio.html" target="Right_Vertical">Mahindra Scorpio</a>
    </h3>
    <h3>
      <a href="Etios.html" target="Right_Vertical">Toyota Etios Liva</a>
    </h3>
  </body>
</html>
```

The right frame HTML document is as follows-

Description.html

```
<!DOCTYPE html >
<html >
  <head>
    <title>My Frames Page</title>
  </head>
  <body>
    <center>
      <h3> WELCOME </h3>
      This page displays the details of a car you have chosen.
    </center>
  </body>
</html>
```

Step 3 : Following are the three html documents each containing the description of each car.

Innova.html

```
<!DOCTYPE html >
<html>
  <head>
    <title>My Frames Page</title>
  </head>
  <body>
    <h2> TOYOTA INNOVA</h2>
    <ul type="disc">
      <li>Price: Rs.9,35,731-Rs.14,82,852</li>
      <li><td>Mileage: </td><td>10Kmpl(City) and 13 Kmpl(hwyr)</li>
    </ul>
  </body>
</html>
```

Scorpio.html

```
<!DOCTYPE html>
<html >
  <head>
    <title>My Frames Page</title>
  </head>
  <body>
    <h2> MAHINDRA SCORPIO</h2>
    <ul type="disc">
      <li>Price: </td><td>Rs.7,49,583-Rs.12,85,479</li>
      <li>Mileage: </td><td>12Kmpl(City) and 16 Kmpl(hwyr)</li>
    </ul>
  </body>
</html>
```

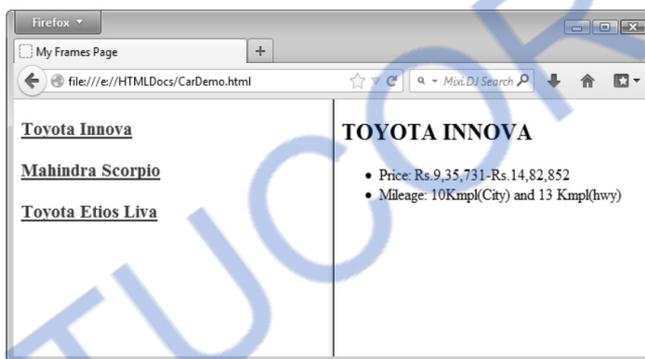
Etios.html

```

<!DOCTYPE html>
<html >
  <head>
    <title>My Frames Page</title>
  </head>
  <body>
    <h2> TOYOTA ETIOS LIVA </h2>
    <ul type="disc">
      <li> Price: </td><td>Rs.4,46,276-Rs.6,86,426</li>
      <li> Mileage: </td><td>17Kmpl(City) and 19 Kmpl(hwy)</li>
    </ul>
  </body>
</html>

```

Step 4 : Open the suitable web browser and invoke the CarDemo.html(created in Step 1), the output will be as follows -



Ex. 2.9.2 : Create a HTML document for a company home page and explain.

AU : May-13, Marks 16

Sol. : For creation of this web page the frames are used.

Step 1 : The main page is created as follows -

Main.html

```

<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <frameset rows="20%,80%">
    <frame src="logo.html">
    <frameset cols="30%,70%">
      <frame src="menu.html">
      <frame src="contents.html">
    </frameset>
  </frameset>
</html>

```

Step 2 : In the main page a layout of the company home page is created. Various html files that are called in this HTML document are as given below -

Menu.html

```
<html>
<head><title>LOGO</title></head>
<body bgcolor="aqua">
<h3><a href="about.html"> About us</a></h3>
<h3><a href="download1.html"> Free Downloads</a></h3>
<h3><a href="download2.html"> Download Catalogue</a></h3>
<h3><a href="download3.html">Download Orderform</a></h3>
<h3><a href="catalogues.html">Catalogues</a></h3>
<h3><a href="distributors.html">Distributors</a></h3>
<h3><a href="Contacts.html">Contact Us</a></h3>
<h3><a href="feedback.html">Feedback</a></h3>
</body>
</html>
```

Logo.html

```
<html>
<head><title>LOGO</title></head>
<body bgcolor="magenta">
<h1> Technical Publications</h1>
</body>
</html>
```

Contents.html

```
<html>
<head><title>LOGO</title></head>
<body bgcolor="khaki">
<center>
  <h2>BOOKS</h2>
</center>
  <p>TECHNICAL PUBLICATIONS is known for commitment to quality and innovation. We are Leaders in our chosen scholarly and educational markets, serving the book Industry & Academic Institutions.
  </p>
  <p>We have been in the industry for the last 18 years and are known for the quality and scholarly publications in Engineering books. We publish more than 1000 titles of text books, for various Universities across the India.
  </p>
  <p>We are specialized in Engineering text books and have been publishing titles for various Engineering branches such as Electrical, Electronics, Computer Science, Information Technology, Mechanical etc. and other management
```

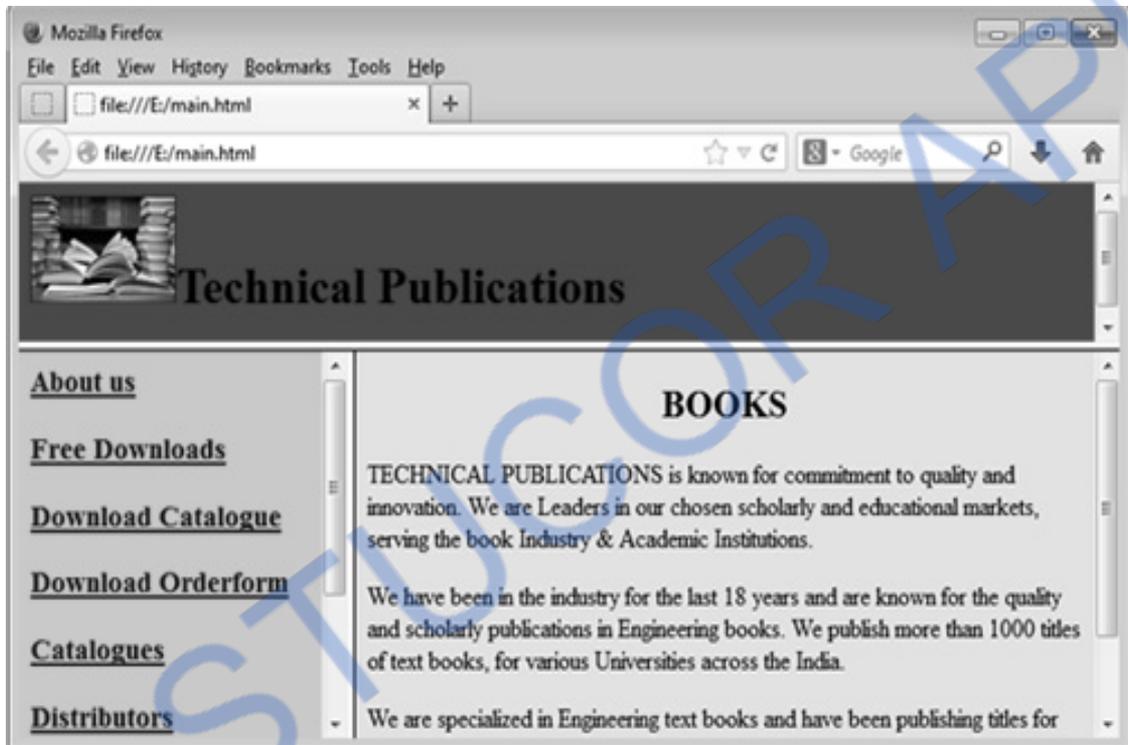
books.

```
</p>
```

```
</body>
```

```
</html>
```

Step 3 : Open the suitable web browser and type the address for **main.html**. The output will be as follows -



University Questions

1. Explain FRAME and IFRAME tags and attributes.
2. Write down HTML tags to explain frame within a frame.

AU : May-14, Marks 8

AU : Dec.-09, Marks 2

2.10 Forms

AU : May-09, Dec.-12, Marks 16

- Form is a typical layout on the web page by which a user can interact with the web page.
- Typical component of forms are **text, text area, checkboxes, radio buttons and push buttons**.
- HTML allows us to place these form components on the web page and send the desired information to the destination server.
- All these form contents appear in the <form> tag. The form has an attribute action which gets executed when user clicks a button on the form.

- Various attributes of form are -

Attribute	Description
action	It specifies the url where the form should be submitted.
method	It specifies the HTTP methods such as GET, POST
name	This attribute denotes the name of the form.
target	It specifies the target of the address in the action attribute.

Let us learn various form components with the help of simple HTML documents.

2.10.1 Text

- **Text** is typically required to place one line text. For example if you want to enter some name then it is always preferred to have Text field on the form.
- The text field can be set using

```
<input type="text" size="30" name="username" value="" >
```

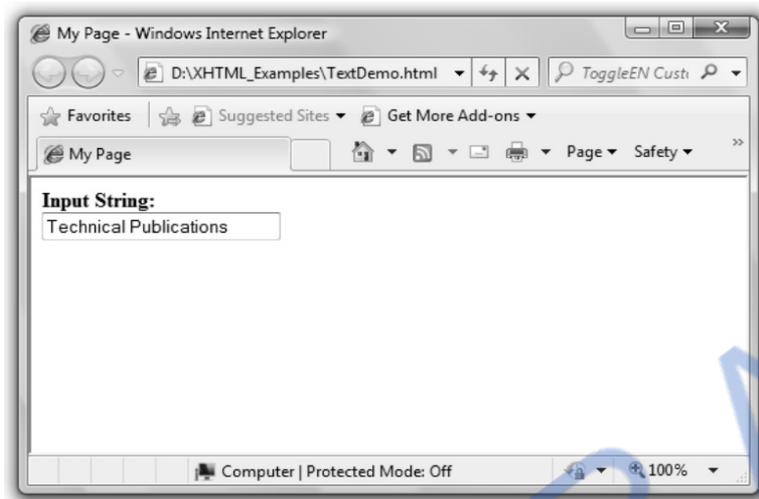
The input type is **text** and the **value** of this text field is “ ” That means the blank text field is displayed initially and we can enter the text of our choice into it. There is **size** parameter which allows us to enter some size of the text field.

- Some other parameters or attributes can be
 - **maxlength** that allows us to enter the text of some maximum length.
 - **name** indicates name of the text field.
 - **align** denotes the alignment of the text in the text field. The alignment can be left, right, bottom and top.

Example Code

HTML Document [TextDemo.html]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>My Page</title>
  </head>
  <body>
    <form>
      <b>Input String:</b><br/><input type="text" size="25" value="">
    </form>
  </body>
</html>
```

Output**Script Explanation :**

In above document

- 1) we have the label "Input String" just before the `<input>` tag. We can also specify the label by using the `<label>` tag as follows -

```
<label>Input String: <br/><input type="text" size="25" value=""></label>
```

- 2) Thus the label gets bound to the text box. This aspect is always beneficial for a web programmer because using label control we can focus on the corresponding text box contents.
- 3) Initially the text box field is blank. We can type some text inside this text box.

Ex. 2.10.1 : How will you create password field in a HTML form ?

Sol. : The password field is typically created on the form. Hence following code can be written to create it -

```
<form name="form1">  
  Password:<input type="password"/>  
</form>
```

2.10.2 Text Area

- Text field is a form component which allows us to enter single line text, what if we want to have multiple line text? Then you must use textarea component.

HTML Document [TextareaDemo.html]

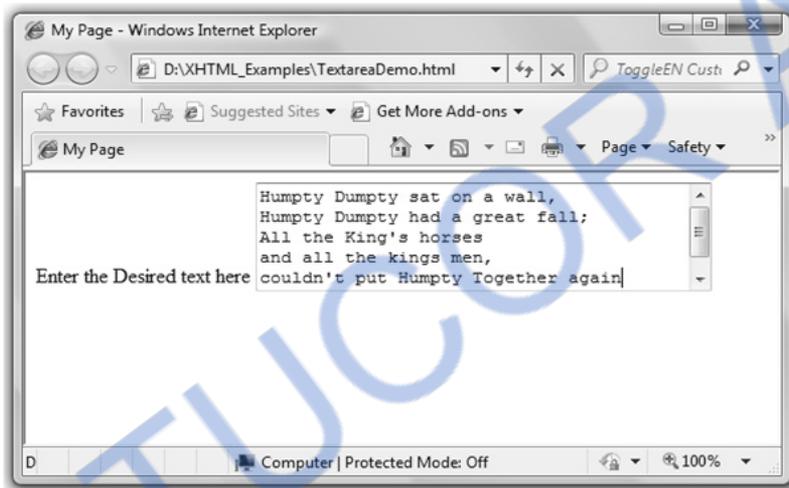
```
<!DOCTYPE html>  
<html>
```

```

<head>
  <title>My Page</title>
</head>
<body>
  <form>
    Enter the Desired text here
    <textarea cols="40" rows="5" name="myname">
  </textarea>
  </form>
</body>
</html>

```

Output



Various parameters that can be set for the text area can be

- **row** denotes total number of rows in the text area.
- **col** specifies total number of columns in the text area.
- **name** denotes the name of the text area which can be utilised for handling that component for some specific purpose.
- **wrap** can be **virtual** or **physical**. If the **wrap** is virtual then the line breaks get disappeared when the text is actually submitted to the server. But if the wrap is assigned to the **physical** then the line breaks (if any) appear as it is in the text.

2.10.3 Checkbox

- It is the simplest component which is used particularly when we want to make some selection from several options.
- For having the checkbox we have to specify the input type as **checkbox**. For example

```
<input type="checkbox" name="option1" value="mango" checked="checked">Mango<br/>
```

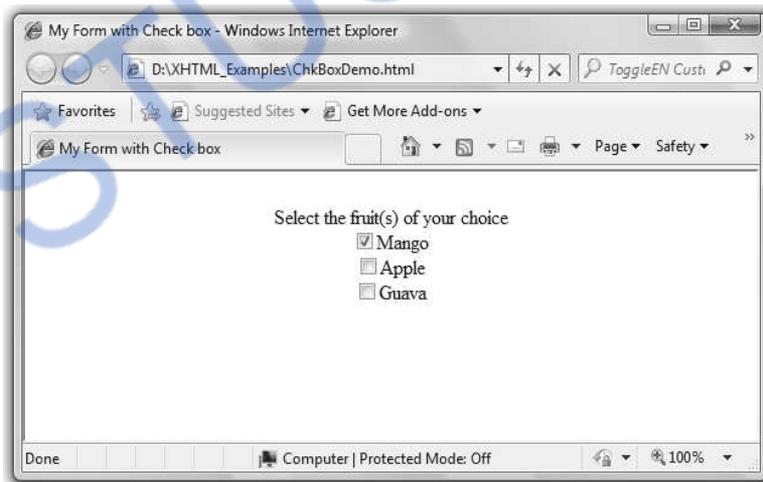
- If we want to get the checkbox displayed as checked then set **checked="checked"**

Example Code

HTML Document[ChkBoxDemo.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Form with Check box</title>
  </head>
  <body>
    <form name ="checkboxForm">
      <div align="center"><br>
      Select the fruit(s) of your choice<br/>
      <input type="checkbox" name="option1" value="mango"
        checked="checked">Mango<br/>
      <input type="checkbox" name="option2" value="apple">Apple<br/>
      <input type="checkbox" name="option3" value="guava">Guava<br/>
      </div>
    </form>
  </body>
</html>
```

Output



Script Explanation

- 1) In the above program to set some checkbox in checked state we can mention the attribute **checked** as **checked**.
- 2) We can set the **value** attribute as "" but this then the checkbox will not get associated with any value. The **Mango**, **Apple** and **Guava** are the labels of the checkboxes.

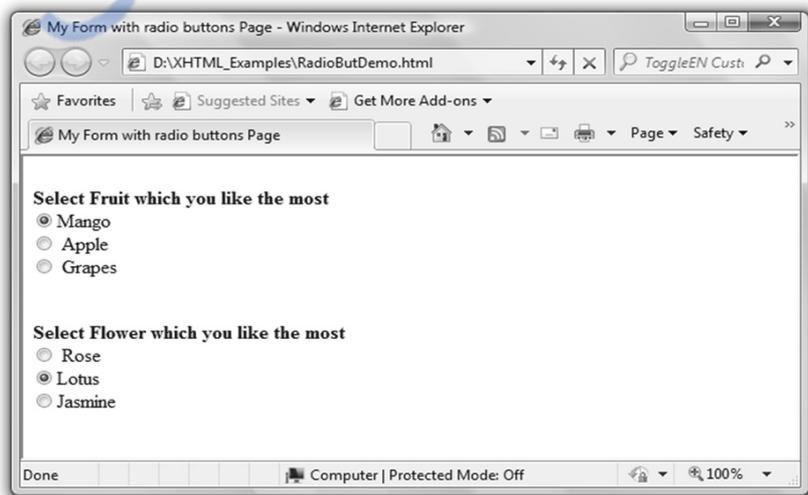
2.10.4 Radio Button

- This form component is also use to indicate the selection from several choices.
- Using **input type="radio"** we can place radio button on the web page.
- This component allows us to make only one selection at a time.
- We can create a group of some radio button component.
- Following HTML document displays the radio buttons for two different groups.

HTML Document[RadioButDemo.html]

```
<!DOCTYPE html>
<html >
  <head>
    <title>My Form with radio buttons Page</title>
  </head>
  <body>
    <form name="myform">
      <div align="left"><br>
        <b>Select Fruit which you like the most</b><br/>
        <input type="radio" name="group1" value="Mango">Mango<br/>
        <input type="radio" name="group1" value="Apple" checked> Apple<br/>
        <input type="radio" name="group1" value="Grapes"> Grapes
        <br/><br/><br/>
        <b>Select Flower which you like the most</b><br/>
        <input type="radio" name="group2" value="Rose"> Rose<br/>
        <input type="radio" name="group2" value="Lotus">Lotus<br/>
        <input type="radio" name="group2" value="Jasmine" checked>Jasmine<br/>
      </div>
    </form>
  </body>
</html>
```

Output



Ex. 2.10.2 : What is the difference between group of checkbox buttons and group of radio buttons ?

Sol. : • The checkbox and radio buttons are used for making the selection from a group of choices.

- When a user selects (checks) a checkbox, its value gets assigned as the current value of the checkbox group's control name.
- A checkbox group's control name may get paired with several current values if the user selects more than one checkbox.
- Radio buttons work just like checkboxes except they are typically set up to be mutually exclusive of one another, i.e. when one is selected, all the others are automatically deselected.

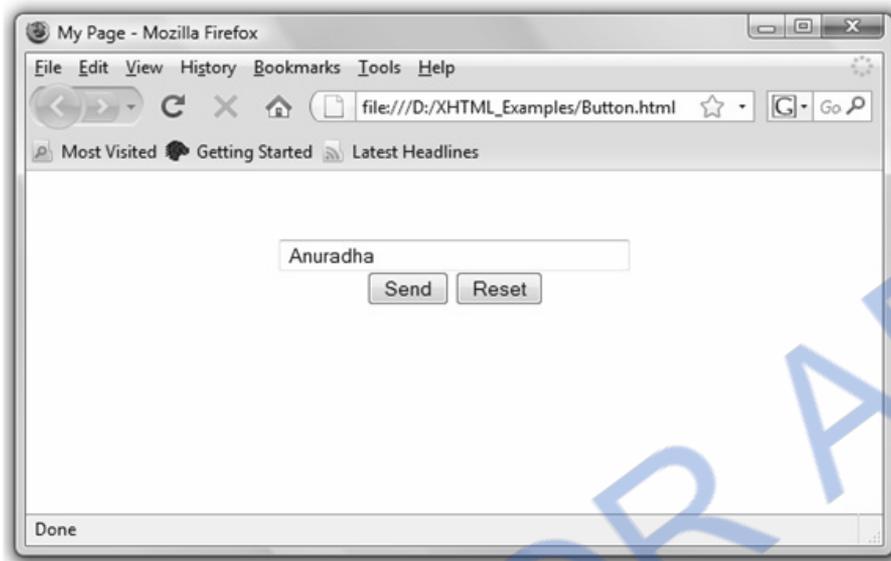
2.10.5 Button

- We can create the button using `<input type =”button”>` tag.
- There are two types of buttons that can be created in HTML. One is called **submit** button and the another one is **reset** button.
- Various parameters of submit button are
 - 1) **name** denotes the name of the submit button.
 - 2) **value** is for writing some text on the text on the button.
 - 3) **align** specifies alignment of the button.

Example Code

HTML Document[Button.html]

```
<!DOCTYPE html>
<html >
  <head>
    <title> My Page </title>
  </head>
  <body>
    <form name="myform" action="http://www.localhost.com/cgi-bin/hello.cgi" method="POST">
      <div align="center">
        <br/><br/>
        <input type="text" size="35" value=" ">
        <br><input type="submit" value="Send">
        <input type="reset" value="Reset"><br>
      </div>
    </form>
  </body>
</html>
```

Output**Script Explanation**

- 1) In above HTML document, we have used the form whose name is “**myform**”.
- 2) There are two attributes associated with the **form** tag and those are **action** and **method**. The **action** parameter indicates the address and the cgi script where the contents should go and **method** parameter is for the methods for submitting the data. The **method** can be **get** or **post**. Thus by specifying the action and method for a form we can send the desired data at desired location.

2.10.6 Menus

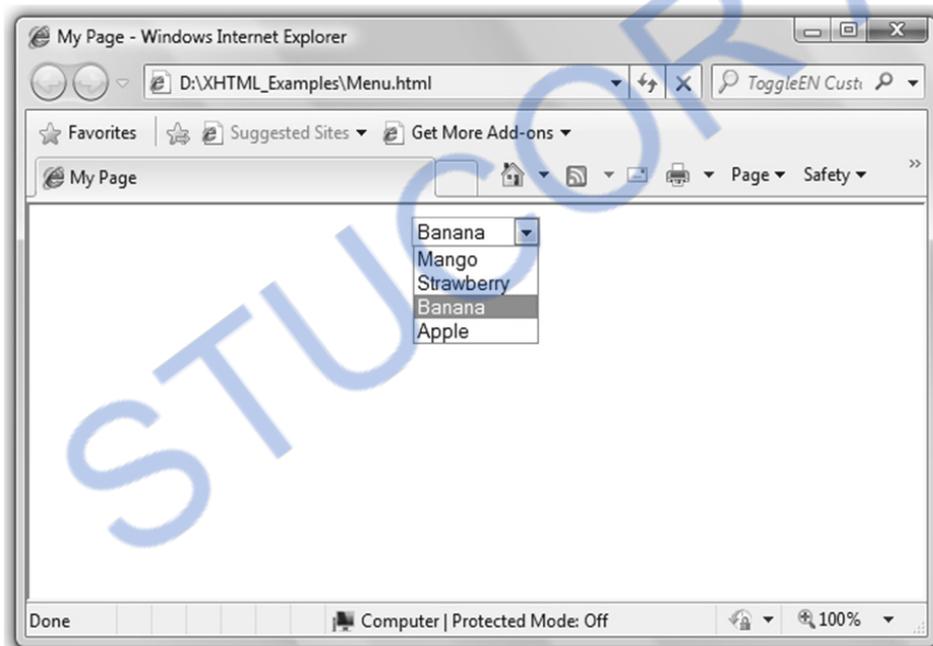
- HTML allows us to have pop down menu on the web page so that the desired selection can be made.
- The parameter **select** is for the menu component and **option** parameter is for setting the values to the options of drop down menu.
- We can make some specific option selected by **selected value =** .
- In the following HTML document we have created one drop down menu in which various fruits are enlisted. By default “Banana” is set as selected.

HTML Document [Menu.html]

```
<!DOCTYPE html>
<html >
  <head>
    <title> My Page </title>
```

```
</head>
<body>
<form name="myform">
<div align="center">
  <select name="My_Menu">
    <option value="Mango">Mango</option>
    <option value="Strawberry">Strawberry</option>
    <option selected value="Banana">Banana </option>
    <option value="Apple">Apple</option>
  </select>
</div>
</form>
</body>
</html>
```

Output



Ex. 2.10.3 : Create a HTML document that has the form with the following controls

- a) A text box to collect the customer name.
- b) Four checkboxes, one for the following items :
 - i. Four HTML textbooks for ₹ 1000
 - ii. Eight XML textbooks for ₹ 2000
 - iii. Four Javabeans books for ₹ 2500
 - iv. Eight UML textbooks for ₹ 1500
- c) A collection of radio buttons that are labelled as follows –
 - i. Cash
 - ii. Cheque/DD
 - iii. Credit card.

Sol. :

```

<!DOCTYPE html>
<html >
  <body>
    <center> <h2>REGISTRATION FORM</h2></center>
    <hr/>
    <form>
      <table>
        <tr>
          <td>Name:</td>
          <td><input type="text" size="25"value="" ></td>
        </tr>
        <tr>
          <td>Select the desired Items:</td>
          <td>
            <select name="f">
              <option value="4 HTML Books(Rs. 1000)">4 HTML Books(Rs. 1000)</option>
              <option value="8 XML Books(Rs. 2000)">8 XML Books(Rs. 2000)</option>
              <option value="4 JavaBeans Books(Rs.2500)">4 JavaBeans Books(Rs.2500)</option>
              <option value="8 UML Books(Rs.1500)">8 UML Books(Rs.1500)</option>
            </select>
          </td>
        </tr>
        <tr>
          <td>
            Mode of Payment:
          </td>
          <td>
            <input type="radio" name="group1" value="Cash">Cash</td>
          </tr>
          <tr>
            <td>
              <input type="radio" name="group1" value="Cheque/DD">Cheque/DD</td>
            </tr>
          <tr>
            <td>
              <input type="radio" name="group1" value="Credit Card">Credit Card</td>
            </tr>
          <tr></tr><tr></tr><tr></tr>
          <td><input type="submit" value="Submit"></td>
          <td><input type="reset" value="Reset"></td>
        </tr>
      </table>
    </form>
  </body>

```

Ex. 2.10.4 : Write a form to collect details of a user such as name, address, radio button to choose subject of book he wants to buy, dropdown to choose favorite author, comments for the last book he read.

Sol. :

```

<!DOCTYPE html >
<html >
<head>
  <title>My Page</title>
</head>
<body>
  <form>
<b>Name:</b> <input type="text" size="20" value=""><br/>
    <b>Address:</b> <input type="text" size="35" value=""><br/>
    <b>Subjects:</b> <br/>
    <input type="radio" name="authors" value="Web Programming">Web Programming<br/>
    <input type="radio" name="authors" value="Computer Network">Computer Network<br/>
    <input type="radio" name="authors" value="Software Engineering">
    Software Engineering<br/>
    <input type="radio" name="authors" value="Data Structures">Data Structures<br/>
    <b>Select favorite Author:</b>
    <select name="MyMenu">
      <option value="AAA">AAA</option>
      <option value="BBB">BBB</option>
      <option value="CCC">CCC</option>
      <option value="DDD">DDD</option>
    </select>
    <br/>
    <b>Comments:</b> <br/>
    <textarea cols="30" rows="10" name="comments">
    </textarea>
    <br/> <br/>
    <input type="submit" value="Submit"/>
    <input type="reset" value="Clear"/>
  </form>
</body>
</html>

```

Output

The screenshot shows a web browser window with the title 'My Page'. The address bar shows 'file:///D:/test.'. The form contains the following elements:

- Name:** A text input field.
- Address:** A text input field.
- Subjects:** A list of radio buttons with labels: Web Programming, Computer Network, Software Engineering, and Data Structures.
- Select favorite Author:** A dropdown menu with 'AAA' selected.
- Comments:** A large text area for entering comments.
- Buttons:** 'Submit' and 'Clear' buttons at the bottom.

Ex. 2.10.5 : Design a simple HTML form for registration of a student.

Sol. : HTML Form for Student Registration

```

<!DOCTYPE html>
<html >
<head>
  <title>My Page</title>
</head>
<body>
  <form>
    <b>Student Name:</b><input type="text" size="20" value=""><br/><br/>
    <b>Address:</b><input type="text" size="35" value=""><br/><br/>
    <b>Email:</b><input type="text" size="20" value=""><br/><br/>
    <b>Password:</b><input type="password" size="10" value=""><br/><br/>
    <b>Select Course:</b><br/><br/>
    <input type="radio" name="courses" value="Computer Engineering">
  
```

```

Computer Engineering<br/><br/>
<input type="radio" name="courses" value="E&TC Engineering">
E&TC Engineering<br/><br/>
<input type="radio" name="courses" value="Mechanical Engineering">
Mechanical Engineering<br/><br/>
<input type="radio" name="courses" value="Civil Engineering">
Civil Engineering<br/><br/>
<b>Select Payment Mode:</b>
<select name="MyMenu">
  <option value="Cheque">Cheque</option>
  <option value="Cash">Cash</option>
  <option value="Card">Card</option>
</select>
<br/><br/><br/>
<input type="submit" value="Submit"/>
<input type="reset" value="Clear"/>
</form>
</body>
</html>

```

Output

My Page x

file:///D:/StudentForm.html ☆

Student Name:

Address:

Email:

Password:

Select Course:

Computer Engineering

E&TC Engineering

Mechanical Engineering

Civil Engineering

Select Payment Mode:

Cheque
Cash
Card

Ex. 2.10.6 : Write and explain any five HTML form objects for a typical online user registration process.

AU : May-09, Marks 8

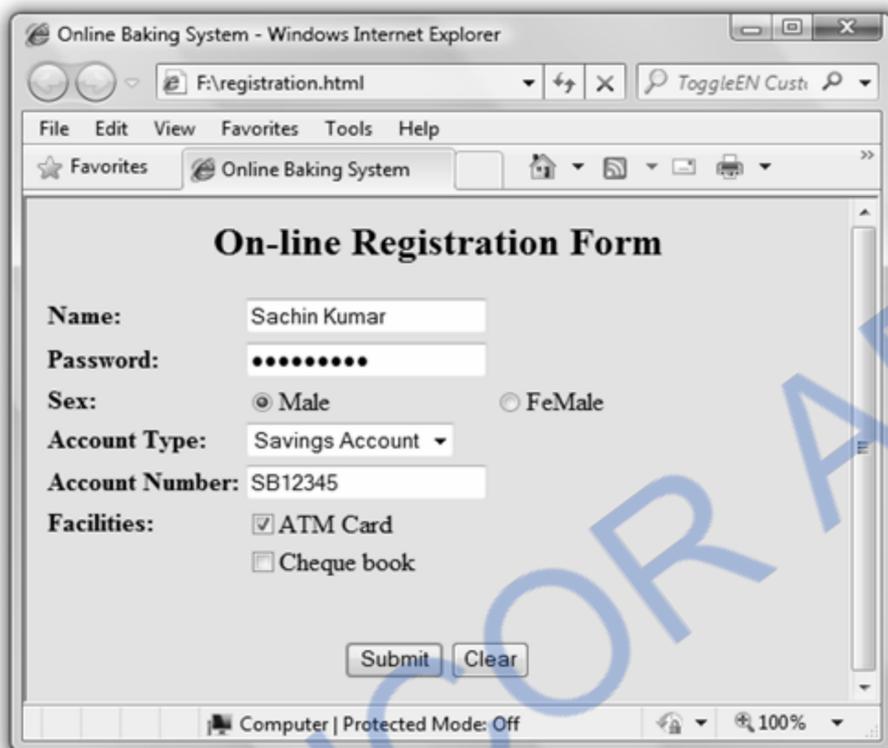
Sol. : The five objects that can be used in the registration form are -

1. Text Field
2. Radio Button
3. Check Box
4. Submit Button
5. Reset Button

A sample application using all these form objects is as given below -

Registration.html

```
<html>
<head>
<title>Online Baking System</title>
</head>
<body bgcolor="khaki">
<center>
<h2>On-line Registration Form</h2>
</center>
<form name="form1">
<table>
<tr><td><b>Name:</b></td><td><input type="text" name="userName"></td>
<tr><td><b>Password:</b></td><td><input type="password" name="pwd"></td>
<tr>
<td><b>Sex:</b></td>
<td><input type="radio" name="group1" value="Male">Male</td>
<td><input type="radio" name="group1" value="Female">Female</td>
</tr>
<tr><td><b>Account Type:</b></td>
<td><select name="MyMenu">
<option value="Savings Account">Savings Account</option>
<option value="Current Account">Current Account</option>
</select>
</td>
</tr>
<tr><td><b>Account Number:</b></td><td><input type="text" name="AccNo"></td>
<tr><td><b>Facilities:</b></td>
<td><input type="checkbox" name="option1" value="ATM card">ATM Card</td></tr>
<tr><td></td><td><input type="checkbox" name="option2" value="Cheque book">
Cheque book</td></tr>
</table>
<br/><br/>
<center>
<input type="submit" value="Submit">
<input type="reset" value="Clear">
</center>
</form>
</body>
</html>
```



Ex. 2.10.7 : Create a registration form for an educational web site with E-learning resources. All form controls should have appropriate name attributes. Use the GET method for form submission and specify an empty string for the action attribute. **AU : Dec.-12, Marks 16**

Sol. :

```
<!DOCTYPE html>
<html>
  <head> <title>My Form </title> </head>
  <body>
    <h3> E-learning Resource Registration Form</h3>
    <form name="myform" action="" method="get" >
      <div align="left"> <br>
      Name:<input type="text" name="name" value="">
      <br/><br/>
      Address:<input type="text" name="add" value="">
      <br/><br/>
      Phone:<input type="text" name="ph" value="">
      <br/><br/>
      Sex:
      <input type="radio" name="group1" value="Male">Male
      <input type="radio" name="group1" value="Female">Female<br/>
```

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

Select the Course of your choice:

```
<select name="My_Menu">
  <option value="Computer">Computer</option>
  <option value="Information Technology">Information Technology</option>
  <option selected value="Mechanical">Mechanical </option>
  <option value="Electronics and Telecommunication">
    Electronics and Telecommunication</option>
  <option value="Electrical">Electrical</option>
</select>
<br/><br/><br/>
<input type="button" value="Submit">
</div>
</form>
</body>
</html>
```

2.11 HTML 5.0

- The HTML 5.0 is the fifth revision of HTML standard of World Wide Consortium(W3C) which is finalized in October 2014.
- The earlier version was HTML 4.0 which was released in 1997.
- The objective of this markup language version was - i) to provide support for latest multimedia and ii) to make the script readable and consistently understood by the developer.
- The HTML5 adds many new syntactic features. It includes the elements such as <video>, <audio> and <canvas> elements. It also has integration of Scalable Vector Graphics(SVG)contents and MathML for mathematical Formula.

2.11.1 Features of HTML 5.0

1. The XHTML doctype is just impossible to memorize (!DOCTYPE html PUBLIC "..."). Hence a new HTML doctype is <!DOCTYPE html>
2. There are new graphic elements such as <canvas> and <svg> in HTML5.0
3. The support for multimedia elements such as <audio> and <video>
4. It has support for <header>, <footer>, <article>, and <section>
5. It has support for new form controls such as number, date, time, calendar, and range.
6. It has a rich set of Application Programming Interface(API) for geolocations, HTML Drag and Drop, Local Storage, Application cache and so on.

2.11.2 Difference between HTML and HTML5

Sr. No.	HTML	HTML5
1.	The DOCTYPE declaration is much longer such as <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">	The DOCTYPE declaration is simple <!DOCTYPE html>
2.	<audio> and <video> tags are not supported.	<audio> and <video> tags are supported for playing audio and video files.
3.	Finding out geographic location is impossible using HTML	The HTML5 supports the API for identifying the geographic location .
4.	It supports the tag such as <applet>, <big>, <center>, , <frame>, <strike>	The tags that are removed from HTML5 are <applet>, <big>, <center>, , <frame>, <strike>
5.	Does not allow JavaScript to run in browser. JavaScript runs in same thread as browser interface.	It allows JavaScript to run in background.
6.	There is no support for <canvas>	The <canvas> tag is for 2D drawing.
7.	It needs external plugins such as flash.	The need for external plugin is reduced .

Two Marks Questions with Answers**Q.1 How to write comment statements in HTML ?**

Ans. : The comments in HTML can be denoted as follows -

<!-- It is a comment statement -->

There should not be a space between angular bracket and exclamation mark. This comment is beginning with <!-- and ending with -->. There should not be any character such as -- inside the comment.

Q.2 What is the use of <pre> tag in HTML ?

Ans. : The pre tag can be used to preserve the white spaces and lines in the text.

Q.3 What is non breaking space character in HTML ?

Ans. : The is a non breaking space character. This entity is useful for defining the space between two strings and informing browser for not performing the word wrapper between the strings.

Q.4 What is cellpadding and cell spacing attributes ?

Ans. : The cellpadding allows to have some space between the contents of each cell and its borders. The distance between each cell is called cell spacing.

Q.5 What is the need of using form in HTML ?

Ans. : Form is a typical layout on the web page by which user can interact with the web page. The components that can be placed on the form are text box, check box, radio buttons, push buttons and so on. Thus form is typically used to create an interactive Graphical User Interface.

Q.6 What is the purpose of using frames in HTML?

Ans. : The XHTML frames allows the web designer to present the web document in multiple views. Using multiple views one can keep the information visible and at the same time other views can be scrolled or replaced. For example, within the same window, one frame can display a company information, the second frame can be a navigation menu and third frame may display selected document that can be scrolled through or replaced by navigating in the second frame.

Q.7 What is the use of markup language in Web technology ? Give examples of some markup languages.

Ans. : The markup language is used to layout the information on the web page. Various markup languages are HTML, XHTML, DHTML, JavaScript.

Q.8 How will you create password field in a HTML form ?

AU : Dec.-11

Ans. : The password field is typically created on the form. Hence following code can be written to create it -

```
<form name="form1">
  Password:<input type="password"/>
</form>
```

Q.9 List and explain any two HTML elements.

AU : May -12

Ans. :

Element	Meaning
<title>	This element is used to specify the title of the web page. This element is defined within the <head> part.
<p>	This element is generally defined in the <body> part. It is used to create a paragraph.

Q.10 List and explain any two special characters in HTML**Ans. :**

Special Character	Entity Reference	Meaning
<	<	To display the less than character.
&	&	To display the ampersand.

Q.11 How would you insert an image file named elephant.jpg at the very top of a web page ?**Ans. :**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<p>

</p>
</body>
</html>
```

Q.12 Is it possible to insert white space in the web page while writing some paragraph ?

Ans. : Yes. We can insert the white space in the paragraph by using the entity reference i.e. non breaking character space.

Q.13 What are the uses of hyperlinks in HTML ?

Ans. : Following are the uses of hyperlinks for the web document -

- i) To logically link one page with another
- ii) Use of link to enhance readability of the web document.
- iii) The navigation from one page to another is possible.

Q.14 What is the advantage of using tables on the web document ?

Ans. : Using tables the information can be arranged systematically in row-wise and column-wise manner.

Q.15 Write HTML code to display an image.**AU : Dec.-12****Ans. :**

```
<html>
<head>
<title> Image Demo </title>
</head>
<body>

```

</body>

</html>

Q.16 List and explain the three flavors of HTML

AU : May -13

Ans. : The three flavors of HTML are as follows -

1. **XHTML 1.0 Strict :** When we want a clean markup code or markup then this type is used. To get the font, color, layout effects this flavor can be used along with the CSS.
2. **XHTML 1.0 Transitional :** By this type we can use the deprecated or presentational elements but frames are not allowed.
3. **XHTML 1.0 Frameset :** We can use this type when we want to use Frames to partition the browser window into two or more frames.

Q.17 How XHTML is more advantageous than HTML ? Specify.

AU : May-14

Ans. : The XHTML is convenient to use if some one needs to reprocess the document using XML. This reprocessing is always needed when we want to send the XHTML document on some other devices say PDA. XML's stricter syntax rules make automatic processing of XHTML much easier and cheaper than ordinary HTML.

Q.18 Enlist any five features of HTML5.0.

Ans. :

1. It has a rich set of Application Programming Interface(API) for geolocations, HTML Drag and Drop, Local Storage, Application cache and so on.
2. There are new graphic elements such as <canvas> and <svg> in HTML5.0
3. The support for multimedia elements such as <audio> and <video>
4. It has support for <header>, <footer>, <article>, and <section>
5. The XHTML doctype is just impossible to memorize(!DOCTYPE html PUBLIC "...). Hence a new HTML doctype is <!DOCTYPE html>

Q.19 How do the hyperlinks appear by default ?

Ans. : By default the hyperlinks are unvisited, underlined and blue.

Q.20 Mention the HTML tag that can be used to make a picture a link.

Ans. : The href tag around **img** can be used to make a picture a link. For example -

```
<a href = "http://www.myhome.com" > <img src =
http://myhome.com/gallery/sampleflat.jpg> </a>
```

Q.21 What are HTML forms ?

Ans. : HTML form is an element used to allow a user to input data on the web page. The element used in <form>. The main attributes are **action** and **method**. For example

```
<form action =http://www.mywebpage.com/ method=get></form>
```

Q.22 What do you mean by row spanning and column spanning ?

Ans. : The row spanning is used to combine two or more rows and columns spanning is used to combine two or more columns.

Q.23 Explain the format of HTML document.

Ans. : The format of HTML document is as follows -

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
    ... HTML statements
  </body>
</html>
```

Q.24 What are the uses of HTML form ?

Ans. : The HTML forms can be used in variety of applications such as

1. For creating registration forms
2. For creating the login forms
3. For gathering user information
4. For conducting surveys.

Q.25 How you use Form's Action Attribute and Submit Button in HTML ?

Ans. : The action attribute specifies where to send the form-data when a form is submitted.

For example - On clicking the submit button, the **mypage.php** page will be invoked for further processing.

```
<form action="mypage.php" method="get">
  First name: <input type="text" name="name"> <br>
  <input type="submit" value="Submit">
</form>
```

Q.26 How to use Line Break and Horizontal Line tags in HTML ?

Ans. : • The line break tag is denoted by

- The horizontal line can be inserted in the HTML document by <hr> tag.

Q.27 How you define href, target and name Attributes ?

Ans. : The href tag is used to have hyperlink in the HTML document. The **target** attribute specifies where to open the linked document. For example - Following created hyperlink helps the user to open the web document in another page.

```
<a href="/" target="_blank">The home page will open in another tab.</a>
```

Q.28 Why we use XHTML ?**AU : Dec.-17**

Ans. : The XHTML is used for following reasons -

- i) It creates strict standard for making web pages.
- ii) It reduces the incompatibility between various web browsers, and hence is compatible for almost all the web browsers.
- iii) It creates a standard that can be used on variety of different devices without change.

Q.29 What is HTML canvas and what are the uses of it ?**AU : Dec.-18**

Ans. :

- The < canvas > is an element in HTML 5.0 which creates rectangular area on HTML page on which we can draw graphs.
- The uses of it are
 - i) For drawing graphics such as boxes, circles, text, adding images.
 - ii) Online, offline games.
 - iii) Animations
 - iv) Interactive video and audio.



Unit II

3	CSS
----------	------------

Syllabus

Style Sheets : CSS-Introduction to Cascading Style Sheets-Features-Core Syntax-Style Sheets and HTML- Style Rule Cascading and Inheritance-Text Properties-Box Model Normal Flow Box Layout-Beyond the Normal Flow-CSS3.0.

Contents

3.1	Introduction to Cascading Stylesheet		
3.2	Features	May-11,14,	Marks 8
3.3	Core Syntax		
3.4	Selectors		
3.5	Style Sheet and HTML		
3.6	Style Rule Cascading and Inheritance		
3.7	Text Properties.....	Dec.11,	Marks 8
3.8	Box Model		
3.9	Color		
3.10	Background Image	May-09, 13,	Marks 8
3.11	Normal Flow Box Layout		
3.12	Beyond Normal Flow		
3.13	CSS 3.0		
	<i>Two Marks Questions with Answers</i>		

3.1 Introduction to Cascading Stylesheet

- The CSS stands for Cascading Style Sheet.
- The Cascading Style Sheet is a markup language used in the web document for presentation purpose.
- The **primary intension of CSS** was to separate out the web content from the web presentation.
- Various elements such as text, font and color are used in CSS for presentation purpose. Thus CSS specification can be applied to bring the styles in the web document.

3.2 Features

- By combining CSS with the html document, considerable amount of **flexibility** into the content submission can be achieved.
- Similarly, separating out the style from actual contents help in **managing** large-scale complex sites. Thus CSS facilitates publication of content in multiple presentation formats.
- If CSS is used, effectively then **global style sheet** can be applied to a web document. This helps in maintaining the **consistency in the web document**.
- If a **small change** needs to be done in the style of web content, then CSS makes it more convenient.
- All above mentioned advantages of CSS show the need for it in web development

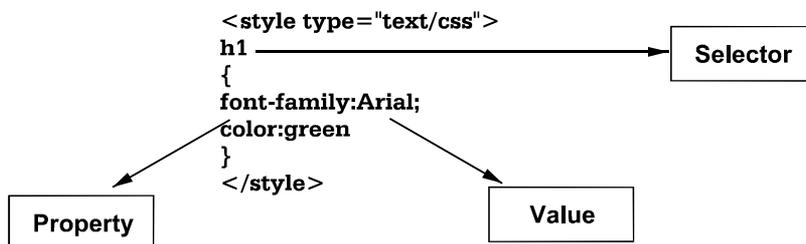
University Question

1. Explain the features of cascading style sheets.

AU : May-11,14, Marks 8

3.3 Core Syntax

- The style specification is specified differently for each different level. For instance :
- For inline cascading style sheets the **style** appears in side the tag for defining the value.
`<p style="font-size: 30pt ;font-family:Script">`
- For the document cascading style sheet the **style** specification appear as the content of a style element within the header of a document.



The **type** attribute tells the browser that what it is reading is text which is affected by the Cascading Style Sheet. The type specification is necessary because there is one more specification used in JavaScript.

- The External style sheet makes use of **style** specification in the same manner as in document cascading style sheet.

The most commonly used CSS properties are enlisted in the following table -

Property Type	Property
Fonts	font font-family font-size font-style font-weight @font-face
Text	letter-spacing line-height text-align text-decoration text-indent
Color and background	background background-color background-image background-position background-repeat color
Borders	border border-color border-width border-style border-top border-top-color border-top-width

Spacing	padding padding-bottom, padding-left, padding-right, padding-top margin margin-bottom, margin-left, margin-right, margin-top
Sizing	height max-height max-width min-height min-width width
Layout	bottom, left, right, top clear display float overflow position visibility z-index
Lists	list-style list-style-image list-style-type

3.4 Selectors

3.4.1 Simple Selector Form

- The simple selector form is a single element to which the property and value is applied.
- We can also apply property and value to group of elements.
- Following is an illustration for simple selector form.

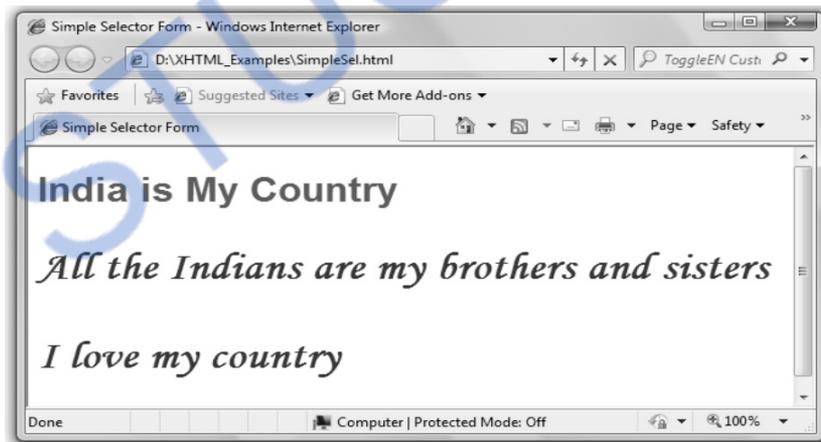
HTML Document[SimpleSel.html]

```
<!DOCTYPE html>
<html >
```

```
<head>
<title>Simple Selector Form</title>
<style type="text/css">
h1
{
font-family:Arial;
color:green;
}
h2,h3
{
font-family:Monotype Corsiva;
color:red;
font-size: 28pt;
}
</style>
</head>
<body>
<h1>India is My Country</h1>
<h2>All the Indians are my brothers and sisters</h2>
<h3>I love my country</h3>
</body>
</html>
```

We can apply style to more than one selector

Output



Similarly the style can also be applied to the elements at specific positions.

For example :

```
body b p {font-size:18pt;}
```

Note that there are more than one element to which the style is applied and these elements are separated by the white spaces.

3.4.2 Class Selectors

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

For example

HTML Document[ClassSel.html]

```
<!DOCTYPE html>
<html>
<head>
<title>Class Selector Form</title>
<style type="text/css">
  h1.RedText
  {
  font-family:Monotype Corsiva;
  color:red;
  font-size: 14pt;
  }
  h1.BlueText
  {
  font-family:Arial;
  color:blue;
  font-size: 10pt;
  }
</style>
</head>
<body>
  <h1 class ="RedText">India is My Country</h1>
  <h1 class="BlueText">All the Indians are my brothers and sisters</h1>
  <h3>I love my country</h3>
</body>
</html>
```

Output



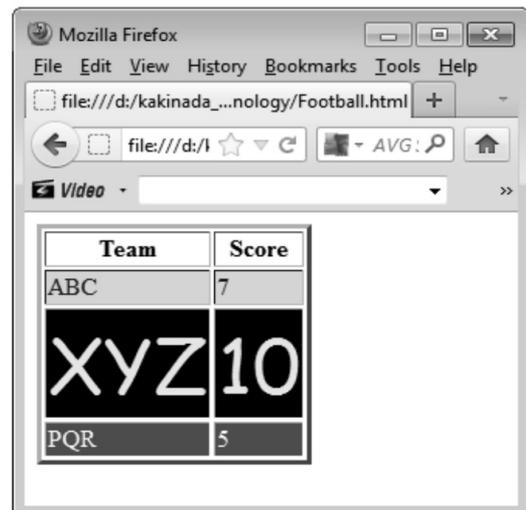
Script Explanation :

Note that in above given script we have used two different classes for the element h1. One class selector is for displaying the string in red color and another is for displaying the string in blue color. This definition is given in the head section. In the body section these class names appear in double quotes inside the h1 tag.

Ex. 3.4.1 : Create a HTML document that displays a table of basketball scores at national games in which the team names have their respective team colors. The score of the leading/wining team should appear larger and in different font than the losing team. Use CSS.

Sol. : Football.html

```
<!DOCTYPE html>
<html >
<head>
<style type="text/css">
tr.one
{
background-color:pink;
color:blue;
}
tr.two
{
font-family:cursive;
background-color:black;
color:yellow;
font-size:50px;
}
tr.three
{
background-color:green;
color:white;
}
</style>
</head>
<body>
<table border="3">
<th>Team</th>
<th>Score</th>
<tr class="one"><td>ABC</td><td>7</td>
</tr>
<tr class="two"><td>XYZ</td><td>10</td>
</tr>
<tr class="three"><td>PQR</td><td>5</td>
```

Output

```
</tr>
```

```
</body>
```

```
</html>
```

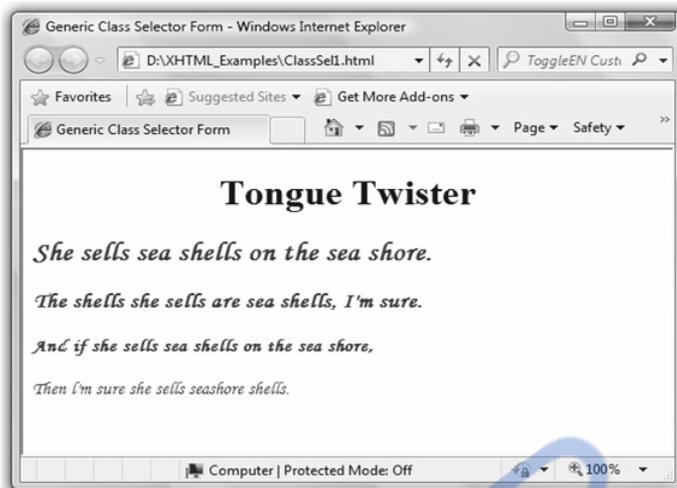
3.4.3 Generic Selectors

- We define the class in generalised form.
- In the sense, that particular class can be applied to any tag.
- Here is the HTML document which makes use of such generic selector.

HTML Document [ClassSel1.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Generic Class Selector Form</title>
    <style type="text/css">
      .RedText
      {
        font-family:Monotype Corsiva;
        color:red;
      }
    </style>

  </head>
  <body>
    <center>
      <h1> Tongue Twister</h1>
    </center>
    <h2 class="RedText">
      She sells sea shells on the sea shore.
    </h2>
    <h3 class="RedText">
      The shells she sells are sea shells, I'm sure.
    </h3>
    <h4 class="RedText">
      And if she sells sea shells on the sea shore,
    </h4>
    <p class="RedText">
      Then I'm sure she sells seashore shells.
    </p>
  </body>
</html>
```

Output

Note that the class selector must be preceded by a dot operator.

3.4.4 Id Selectors

- The id selector is similar to the class selector but the only difference between the two is that class selector can be applied to more than one elements where as using the id selector the style can be applied to the one specific element.
- The syntax of using id selector is as follows -
`#name_of_id {property:value list;}`
- Following HTML document makes use of id selector

HTML Document[IdSel.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>id Selector</title>
    <style type="text/css">
      #top
      {
        font-family:Monotype Corsiva;
        color:blue;
        font-size:16pt;
      }
    </style>
  </head>
  <body>
    <div id="top">
```

It is the mark of an educated mind to be able to entertain a thought without accepting it.

</div>

<p>

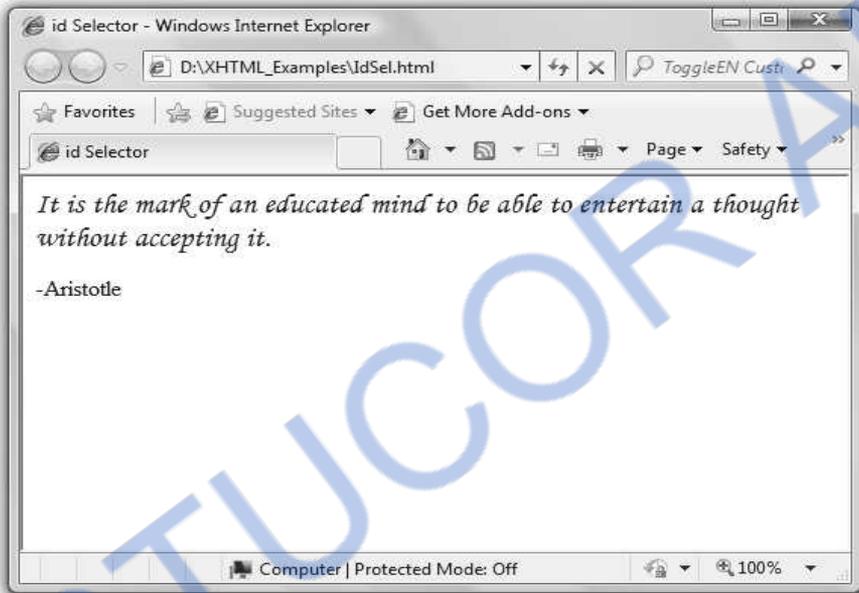
-Aristotle

</p>

</body>

</html>

Output



3.4.5 Universal Selectors

This selector is denoted by * (asterisk). This selector can be applied to all the elements in the document.

HTML Document[UniverSel.html]

```
<!DOCTYPE html>
```

```
<html >
```

```
  <head>
```

```
    <title>Universal Selector</title>
```

```
    <style type="text/css">
```

```
      * {
```

```
        color:green;
```

```
      }
```

```
    </style>
```

```
  </head>
```

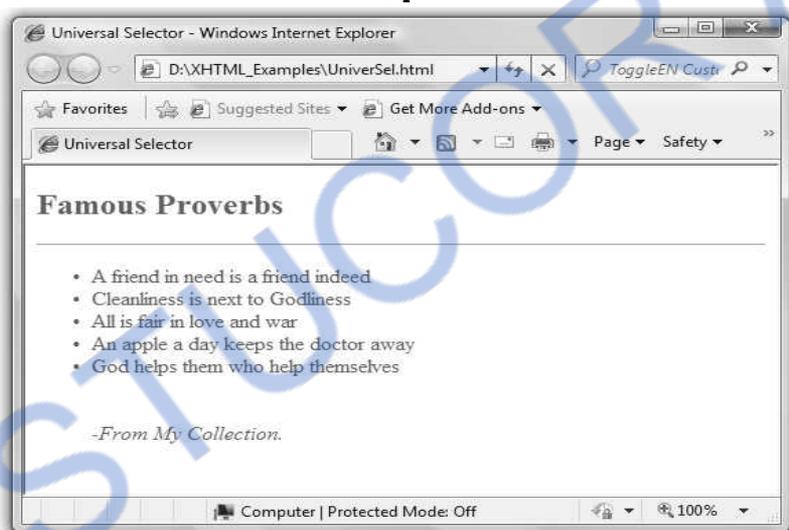
```
</body>
```

```

<h2> Famous Proverbs</h2>
<hr/>
<ul type="disc">
<li>A friend in need is a friend indeed</li>
<li>Cleanliness is next to Godliness</li>
<li>All is fair in love and war</li>
<li>An apple a day keeps the doctor away</li>
<li>God helps them who help themselves</li>
<br/><br/>
<em>
-From My Collection.
</em>
</body>
</html>

```

Output



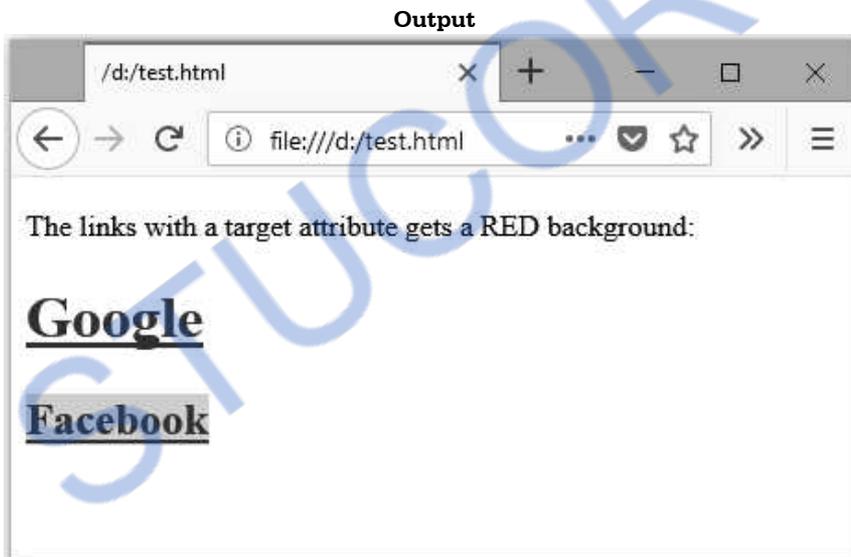
As we have defined the universal selector that sets the green color. Hence the text that is appearing on the above web page is in green color.

3.4.6 Attribute Selector

- CSS allows authors to specify rules that match elements which have certain attributes defined in the source document.
- The syntax is
 - [att] - Match when the element sets the "att" attribute, whatever the value of the attribute.
 - [att=val] - Match when the element's "att" attribute value is exactly "val".
- Example : In the following example all the <a> elements get selected with a target attribute.


```
<!DOCTYPE html>
```

```
<html>
<head>
<style>
  a[target] {
    background-color: red;
  }
</style>
</head>
<body>
<p>The links with a target attribute gets a RED background:</p>
<h1><a href="https://www.google.com">Google</a> </h1>
<h2><a href="http://www.facebook.com" target="_blank">Facebook</a></h2>
</body>
</html>
```



3.4.7 Pseudo Classes

Using pseudo classes we can give special effects on the selectors. There some pseudo classes which are more commonly used and those are -

- Focus
- Hover
- Hyperlink

In the following XHTML documents we have used these pseudo classes.

HTML Document[PsedoCls.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Pseudo classes</title>
    <style type="text/css">
      input:focus
    {
      background-color:skyblue;
      color:white;
    }
  </style>
</head>

<body>
  <form>
    Enter some string:<input type="text" name="my_txt"/>
    <br/>
    <input type="submit" value="Enter"/>
  </form>
</body>
</html>
```

Output

Let us now see one more illustration in which the pseudo class is used.

HTML Document[PsedoCls1.html]

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Pseudo classes for hyperlinks</title>
<style type="text/css">
a.ColoredTxt:link {color: red}
a.ColoredTxt:visited {color: blue}
a.ColoredTxt:hover {color: green}
```

```
a.BigTxt:link {color: red}
a.BigTxt:visited {color: blue}
```

```
a.BigTxt:hover {font-size: 200%}
```

```
</style>
</head>
<body>
<h3>Move Mouse over following text</h3>
```

```
<p>
<em>
```

```
  <a class="ColoredTxt" href="mypage.html" target="_blank">The text
  changes color
  </a>
```

```
</em>
```

```
</p>
```

```
<p>
```

```
<em>
```

```
  <a class="BigTxt" href="mypage.html" target="_blank">The text is getting
  Big in size
  </a>
```

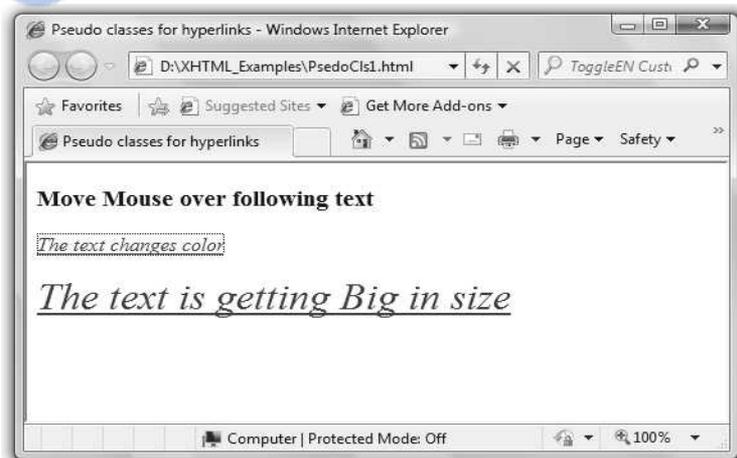
```
</em>
```

```
</p>
```

```
</body>
```

We have defined two pseudo classes named ColoredTxt and BigTxt

Output



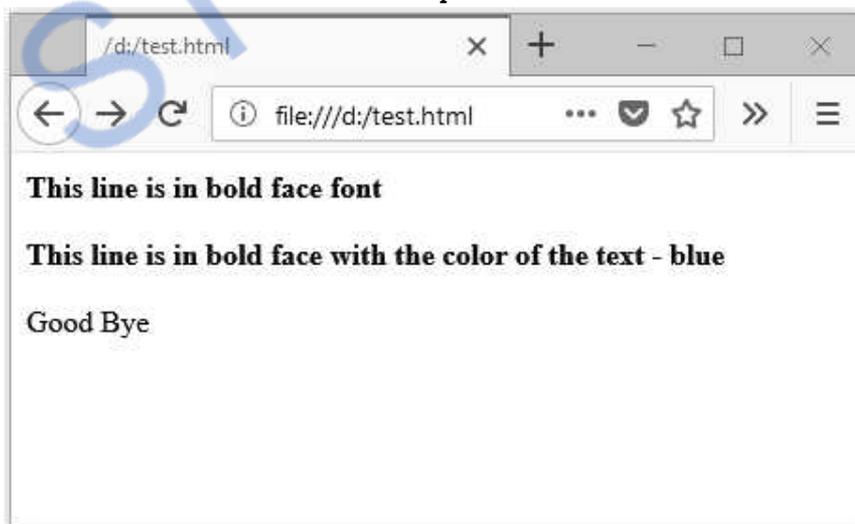
3.4.8 Contextual Selector

- "Contextual selectors" in CSS allow you to specify different styles for different parts of your document.
- You can assign styles directly to specific HTML tags, or you can create independent classes and assign them to tags in the HTML.

For example –

```
<!DOCTYPE html>
<html>
<head>
<style>
  p b {
    font-family: Times, serif; /* Font family */
    font-weight: bold; /* heavy faced type*/
    color: blue; /* blue colour of the text */
  }
</style>
</head>
<body>
<div><b>This line is in bold face font</b></div>
<p><b>This line is in bold face with the color of the text - blue </b></p>
<p> Good Bye</p>
</body>
</html>
```

Output



3.4.9 At Rules

The @import rule is used to import one style sheet into another. The syntax for using @import is

```
@import url("mystyles.css")
```

3.5 Style Sheet and HTML

There are three levels of cascading style sheets -

- Inline style sheet
- Document level style sheet
- External level style sheet

3.5.1 Inline Style Sheet

The inline cascading style sheet is a kind of style sheet in which the styles can be applied to HTML tags. This tag can be applied using following rule -

```
Tag
{
property: value
}
```

For example :

```
<p style="font-family: Arial;color:red" >
```

Here for the tag **p** two properties are used such as **font-family** and **color** and those are associated with the values such as Arial and red respectively.

Note that if we want to use **more than one property then** we have to use separator such as **semicolon**. In the following HTML document we have used cascading style sheet-

HTML Document [InlineStyle.html]

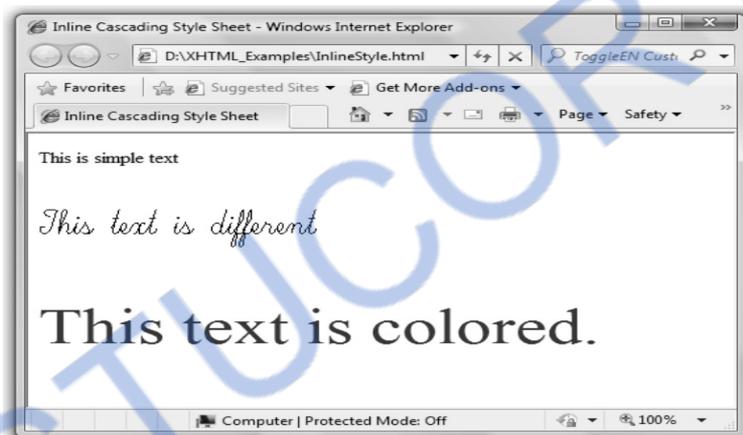
```
<!DOCTYPE html >
<html>
  <head>
    <title>Inline Cascading Style Sheet</title>
  </head>
  <body>
    <p>This is simple text</p>
    <p style="font-size: 30pt ;font-family:Script">This text is different </p>
    <p style="font-size: 40pt ;color:#ff0000">This text is colored.</p>
  </body>
</html>
```

Script Explanation

- 1) In this document, in the body section the style sheets are created.
- 2) In this section first of all we have displayed a simple sentence This is simple text. There is no style for this sentence.
- 3) In the next line, we have applied style in which **font-size** is set to the size of 30 point and **font-family** is set by the font name "Script" .
- 4) Then colored text will be displayed by **color:#ff0000**". The first two digits ff denote the red color, there is no green and blue color as the values next to ff are 00. Hence the text will be displayed in red.

This can be very well illustrated in output.

Output



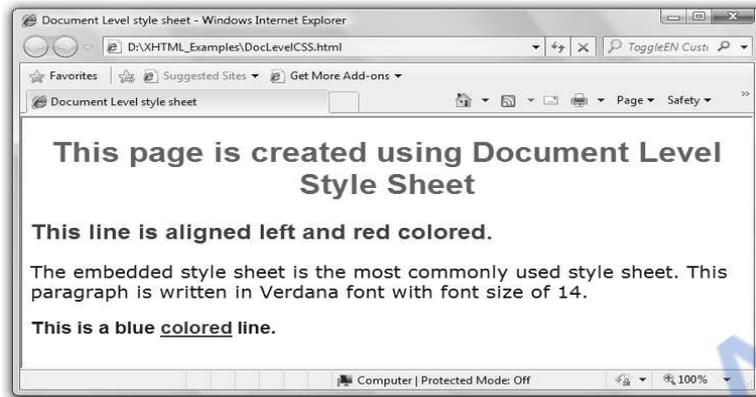
3.5.2 Document Level Style Sheet

- This type style sheet appears only in the head section and in the body section newly defined Selector tags are used with the actual contents.
- For example : In the following HTML script we have defined **h1**, **h2**, **h3** and **p** selectors. For each of these **selectors** different property and values are set. Such setting will help us to represent our web page in some decorative form.
- The most important thing while writing document level style sheet is that we should mention the **style type="text/css"** in the head section. By this the browser will come to know that the program is making use of cascading style sheet.

HTML Document [DocLevelCSS.html]

```
<!DOCTYPE html>  
<html >
```

```
<head>
<title>Document Level style sheet</title>
<style type="text/css">
h1
{
font-family:Arial;
color:green
}
h2
{
font-family:Arial;
color:red;
left:20px
}
h3
{
font-family:arial;
color:blue;
}
p
{
font-size:14pt;
font-family:verdana
}
</style>
</head>
<body>
<h1>
<center>
This page is created using Document Level Style Sheet
</center>
</h1>
<h2>
This line is aligned left and red colored.
</h2>
<p>
The embedded style sheet is the most commonly used style sheet.
This paragraph is written in Verdana font with font size of 14.
</p>
<h3>
This is a blue <a href="colormame.html">colored</a> line.
</h3>
</body>
</html>
```

Output**Script Explanation**

- 1) In above program, we have defined all the selectors in the head sections only. And these HTML elements are then used along with the web page contents.
- 2) The setting defined in the selectors will affect the web page contents. For example we have defined the selector h2 as



and then in the body section we have written -

```
<h2> This line is aligned left and red colored.</h2>
```

- 3) Now as h2 defines font to be "Arial" with color as "red" having left alignment of 20 pixels, the sentence "This line is aligned left and red colored." will be displayed in Arial font, which is red colored and aligned from left by 20 pixels. Surely we can see this effect on our web browser.

3.5.3 External Stylesheet

- Sometimes we need to apply particular style to more than one web documents in such cases external style sheets can be used.
- The central idea in this type of style sheet is that the desired style is stored in one .css file. And the name of that file has to be mentioned in our web pages.
- Then the styles defined in .css file will be applied to all these web pages.

- Here is a sample program in which external style sheet is used.

Step 1 : Create an HTML document

HTML Document[ExtCSS.html]

```
<!DOCTYPE html>
<html >
  <head>
    <link rel="stylesheet" type="text/css" href="ex1.css" />
  </head>
  <body>
    <h1 class="special" > <center> This page is created using External Style
    Sheet</center> </h1>
    <h2>
      This line is aligned left and red colored.
    </h2>
    <p>
      The External style sheet is the compact representation of Cascading Style Sheets.
      This paragraph is written in Monotype Corsiva font with font size of 14.
    </p>
    <h3>
      This is a blue <a href="colorname.html">colored</a> line.
    </h3>
  </body>
</html>
```

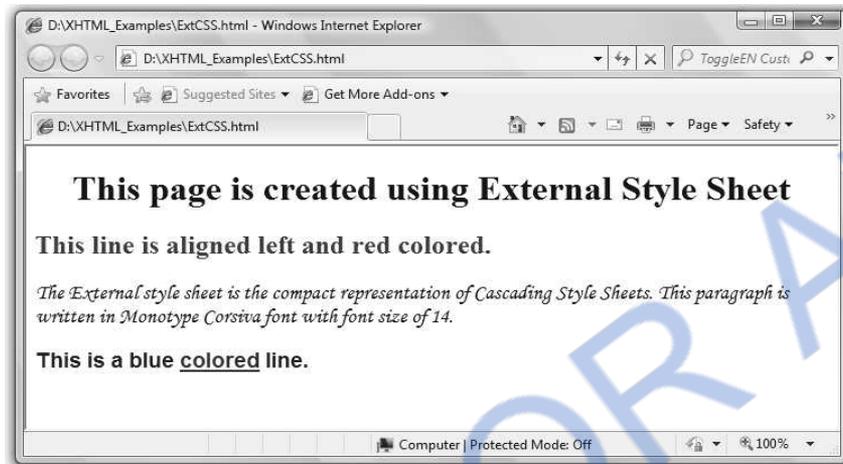
Step 2 : Create a css file which contains the styles that can be applied to different HTML elements present in the above HTML document.

The cascading style sheet ex1.css can be

<!-- The file name ex1.css and can be opened in notepad.-->

```
h1
{
font-family:Arial
}
h2
{
font-family:times new roman;
color:red;
left:20px
}
h3
{
font-family:arial;
color:blue;
}
```

```
p
{
font-size:14pt;
font-family:Monotype Corsiva
}
```



Script Explanation

When we want to link the external style sheet then we have to use `<link>` tag which is to be written in the head section.

- **link** tells the browser some file must be linked to the page.
- **rel=stylesheet** tells the browser that this hyperlink is a style sheet.
- **href=" "** denotes the path name of style sheet file.
- **type="text/css"** tells the browser that what it is reading is text which is affected by the cascading style sheet.

3.6 Style Rule Cascading and Inheritance

3.6.1 Style Rule Cascading

- When there are two or more values for the same property then conflict occurs. There are various levels of style sheets such as embedded CSS and external CSS. Out of which **embedded CSS** have **more precedence over the external CSS**.
- There are some sorting order rules that are used to decide the precedence. These are based on two things, importance (important or normal) of the property and origin (author origin, user origin and user agent origin). Let us first understand the terms important and origins.
- The keyword **!important** can be applied to specify that particular property is important. For example :

```
p.myclass {font-family:Arial !important ;font-style:bold;}
```

specifies that the font being Arial is one property but font-style being bold is an important property.

- When the specification is marked by !important then it actually specifies the weight. The weight can be either important or normal.
- The origin can be author origin, user origin and user agent origin. These terms can be defined as follows -
 - **Author** is a person who writes the XHTML document and associated style sheets.
 - **User** is a person who interacts with user agent and use the web document along with the associated style sheets.
 - **User agent** is a program that interprets the web document written in scripting language and applies the style sheet according to the specification. For example, HTML is an user agent.
- The sorting order can be determined by applying following rules with highest precedence to lowest -
 1. Important declaration having user origin.
 2. Important declaration having author origin.
 3. Normal declaration having author origin.
 4. Normal declaration having user origin.
 5. Any declaration (either important or normal) with user agent (i.e. browser) origin.
- Following rules are also be used to decide the sorting order of the elements from highest precedence to lowest -
 1. **id** selectors have the highest precedence.
 2. **class** and **pseudo class** selectors
 3. Contextual selectors
 4. Universal selectors
- If the conflict still occurs then it can be resolved writing the properties in the specification. The rules that are defined later in the document tree have the highest precedence than those defined earlier.
- **Style rule cascading** can be defined as the process which is used to resolve the conflicts in the style specification. Fig. 3.6.1 shows the steps in CSS cascade.

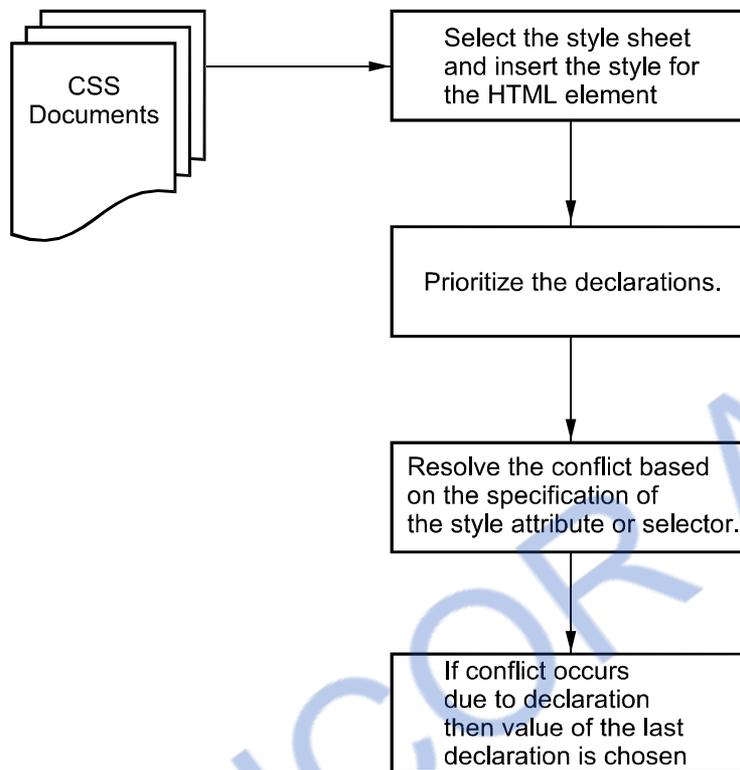


Fig. 3.6.1 Steps in CSS cascade

3.6.2 Inheritance

- In CSS, if a property of an element has no associated declarations then using inheritance technique the value for this property can be obtained.
- In an HTML document, the tags have parent-child relationships. For example, <title> tag is always inside the <head> tag, so the <head> tag is the parent of the <title> tag. Similarly <p> tag is defined inside the <body> tag. Then <body> becomes parent of <p> tag.
- In cascading style sheets, inheritance is based on **tree structure** of the document itself.
- An element inherits the value for its properties by checking the parent element. If the parent has the value for the property demanded by the child then this property value can be inherited. If the parent does not have this value, then its parent (i.e. grandparent) is checked. If this parent has no such property value then its parent is checked. This process is continued until a root element is obtained.
- Thus the search will be made in upward direction from the parent of the element to the root of the tree or to the <html> tag of the document.

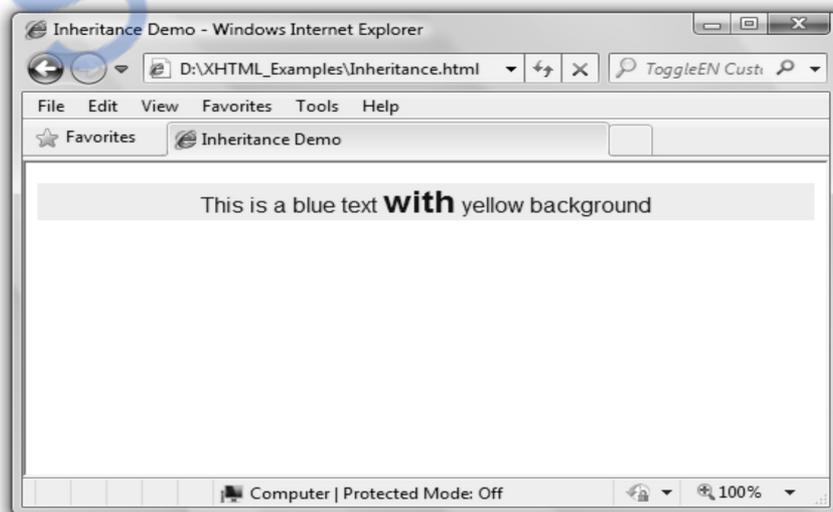
Inheritance can be defined as a concept in which child always inherits the properties of parent tag along with its own properties.

Following is an HML document in which inheritance is demonstrated.

HTML Document

```
<!DOCTYPE html>
<html>
<head>
  <title>Inheritance Demo</title>
  <style type="text/css">
    p
    {
      font-family:Arial;
      color:blue;
      background-color:yellow;
    }
    strong
    {
      font-size:24px;
    }
  </style>
</head>
<body>
  <center>
    <p>This is a blue text <strong>with</strong> yellow background</p>
  </center>
</body>
</html>
```

Output



Script Explanation :

In above, script `<p>` tag has some properties as **Arial** font, blue font color and **yellow** background. The `` tag has only one property can that is font-size which is assigned by the value **24px**. But in the body section, `` tag is the child of `<p>` tag. And note that `` inherits the properties of `<p>` tag. Hence the text defined under`` tag is also blue colored with yellow background.

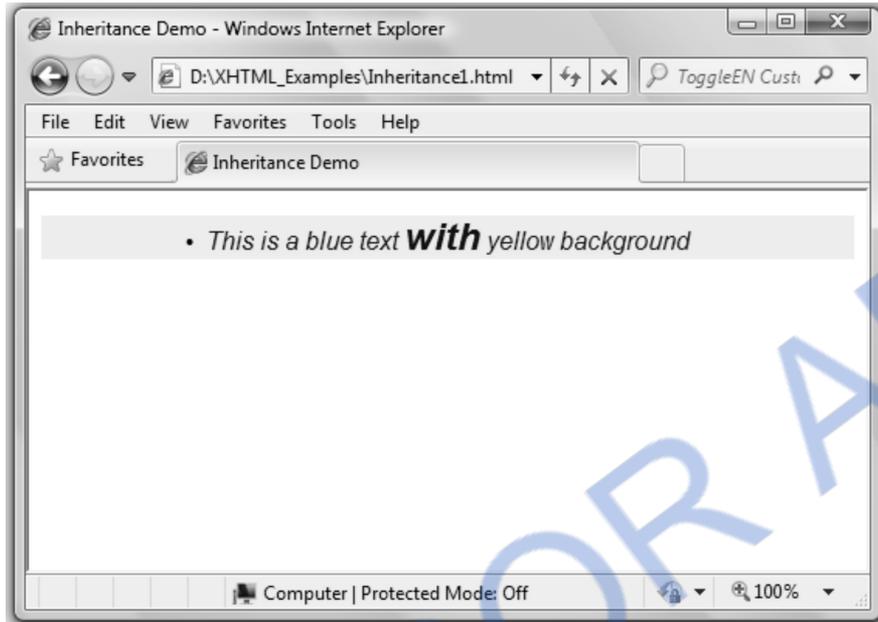
Now just modify by the above script as follows and observe the sample output.

HTML Document [inheritance1.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Inheritance Demo</title>
  <style type="text/css">
    li
    {
      font-style:italic
    }
    p
    {
      font-family:Arial;
      color:blue;
      background-color:yellow
    }
    strong
    {
      font-size:24px
    }
  </style>
</head>
```

```
<body>
  <center>
    <li>
      <p>This is a blue text <strong>with</strong> yellow background</p>
    </li>
  </center>
</body>
</html>
```

Added `` tag as grand parent for `` tag.

Output

Note that the above style sheet contains the nested tags -

```
<li>
.....<p>
.....<strong>
.....</strong>
.....</p>
</li>
```

The `` tag contains the italic property. Hence the text inside the `` tag becomes italic. That means `` tag has inherited the italic property from its grandchild.

3.7 Text Properties

The font properties can be setting different types of fonts, styles and sizes.

3.7.1 Font Families

The **font-family** denotes different types of fonts such as Arial, Times New Roman, Script, monospace and so on. Following are some examples of generic fonts

Generic Font Name	Example
sans-serif	Arial, Helvetica, Futura
cursive	Zapf-chancery

fantasy	Critter, Cottonwood
monospace	Courier, Prestige
serif	Times New Roman, Garamond

Let us see usage of **font-family** property in HTML document.

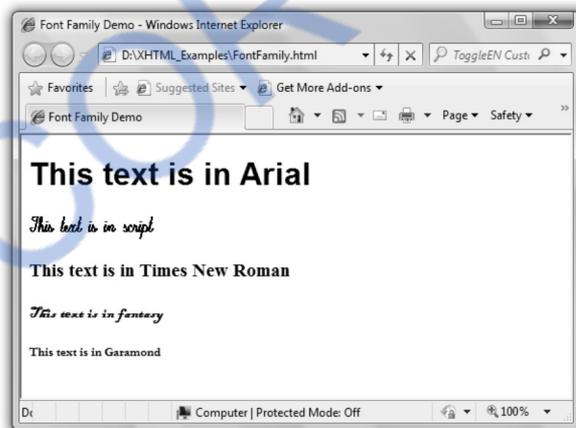
HTML Document[FontFamily.html]

```

<!DOCTYPE html>
<html>
<head>
<title>Font Family Demo</title>
<style type="text/css">
  h1
  {
  font-family:Arial;
  }
  h2
  {
  font-family:script;
  }
  h3
  {
  font-family:'Times New Roman';
  }
  h4
  {
  font-family:fantasy;
  }
  h5
  {
  font-family:Garamond;
  }
</style>
</head>
<body>
  <h1>This text is in Arial </h1>
  <h2>This text is in script </h2>
  <h3>This text is in Times New Roman</h3>
  <h4> This text is in fantasy</h4>
  <h5> This text is in Garamond</h5>

```

Output



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

If the font name has more than one word then the name should be enclosed by the single quotes. Although quotes are not mandatory, it is a good practice to use quotes for specifying the font name.

3.7.2 Font Sizes

This property is used to specify the size of the font. One can specify the font size in points (pts), pixels (px) or in percentage (%). We can also specify the font size using the relative values such as **small**, **medium**, **large**. Use of font size in such a way is relative. And the disadvantage of using such relative sizes is that one cannot have the strict control over the font-size. Different browsers may have different font size values.

In the following HTML document we will use the font-size property.

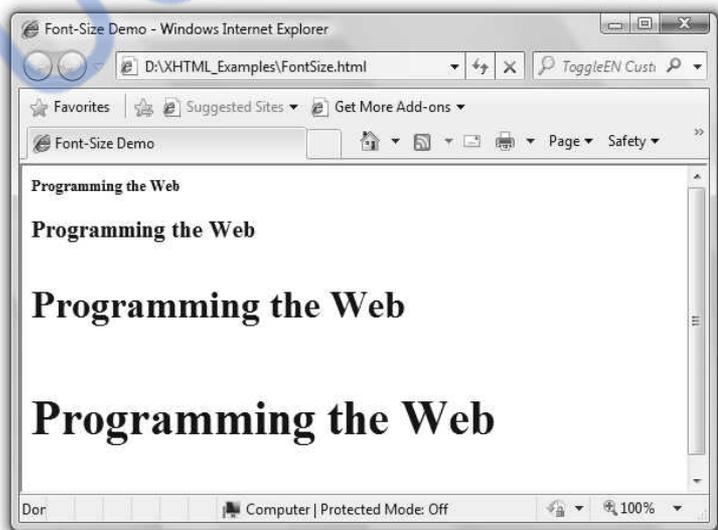
HTML Document[FontSize.html]

```
<!DOCTYPE html>
<html>
  <head>

<title>Font-Size Demo</title>
  <style type="text/css">
    h1
    {
      font-size:10pt;
    }
    h2
    {
      font-size:20px;
    }
    h3
    {
      font-size:xx-large;
    }
    h4
    {
      font-size:250%;
    }

  </style>
</head>
<body>
```

Output



```

<h1>Programming the Web</h1>
<h2>Programming the Web</h2>
<h3>Programming the Web</h3>
<h4>Programming the Web</h4>
</body>
</html>

```

3.7.3 Font Variants

The font variation can be achieved by making setting the font in upper case or in lower case.

We use **font-variant** property for this purpose.

Value	Meaning
normal	The font can be displayed in normal form.
small-caps	The font can be displayed in small capital letters.

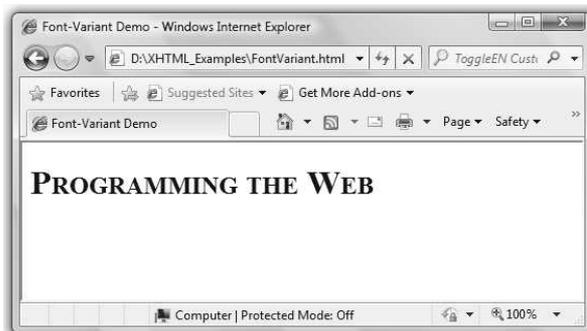
HTML Document[FontVariant.html]

```

<!DOCTYPE html>
<html>
  <head>
    <title>Font-Variant Demo</title>
    <style type="text/css">
      h1
    {
      font-variant:small-caps;
    }
    </style>
  </head>
  <body>
    <h1>Programming the Web</h1>
  </body>
</html>

```

Output



3.7.4 Font Styles

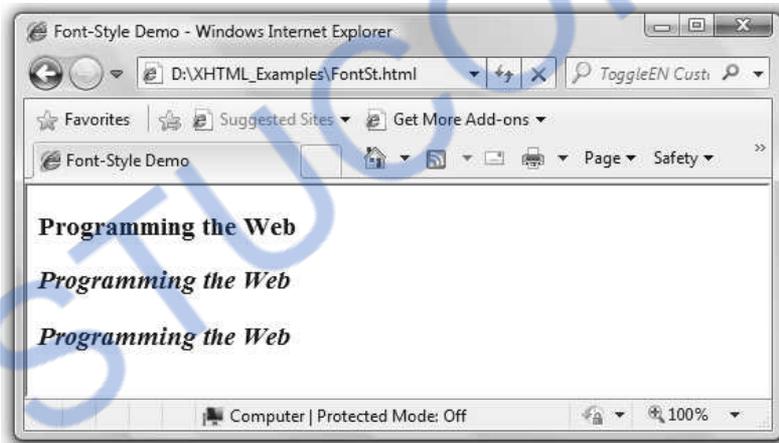
Various font styles are -

- normal
- italic
- oblique

The illustration is as given below -

HTML Document[FontSt.html]

```
<!DOCTYPE html>
<html
  <head>
  <title>Font-Style Demo</title>
  <style type="text/css">
  h3.normal { font-style:normal;}
  h3.italic { font-style:italic;}
  h3.oblique { font-style:oblique;}
  </style>
  </head>
  <body>
  <h3 class="normal">Programming the Web</h3>
  <h3 class="italic">Programming the Web</h3>
  <h3 class="oblique">Programming the Web</h3>
  </body>
</html>
```

Output**3.7.5 Font Weights**

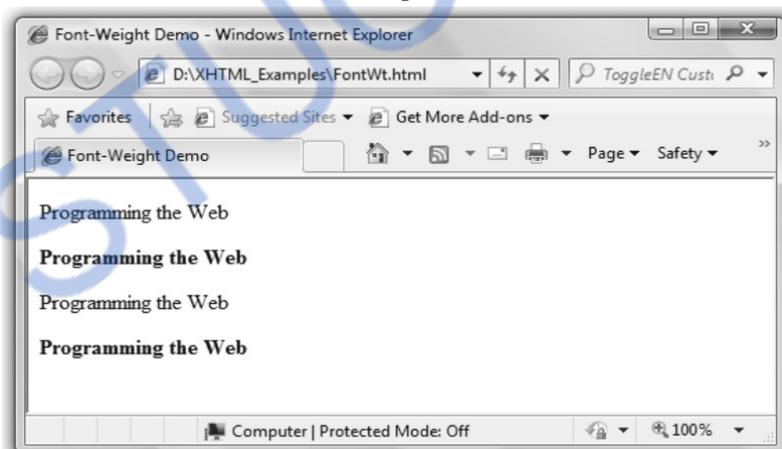
Various font styles are -

- Normal (by default)
- Bold
- Bolder
- Lighter

The illustration is as given below -

HTML Document[FontWt.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Font-Weight Demo</title>
    <style type="text/css">
      p.normal { font-weight:normal;}
      p.bold { font-weight:bold;}
      p.bolder { font-weight:bolder;}
      p.lighter { font-weight:lighter;}
    </style>
  </head>
  <body>
    <p class="normal">Programming the Web</p>
    <p class="bold">Programming the Web</p>
    <p class="lighter">Programming the Web</p>
    <p class="bolder">Programming the Web</p>
  </body>
</html>
```

Output**3.7.6 Font Shorthands**

We can specify more than one font properties in a list

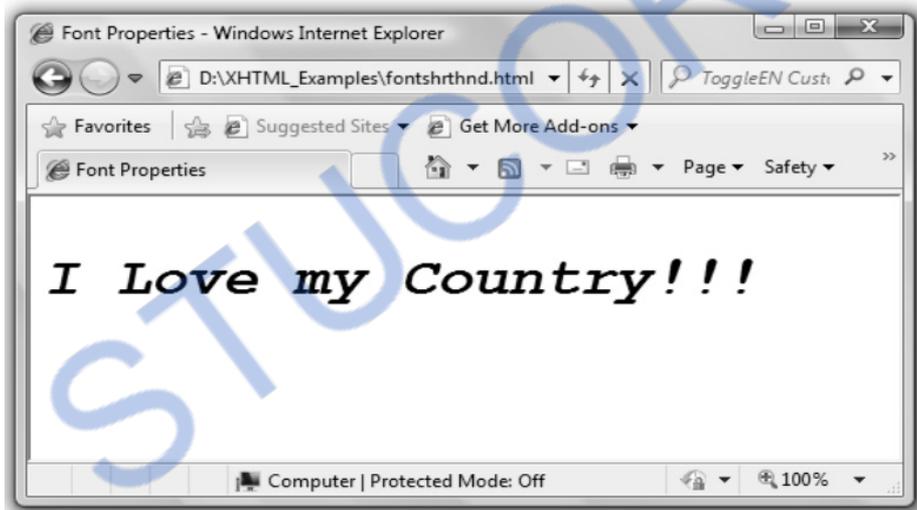
For example

HTML Document[Fontshrthnd.html]

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Font Properties</title>
<style type="text/css">
  p.myfont {font-family:Courier New;
    font-style:italic;
    font-weight:bold;
    font-size:28pt;
  }
</style>
</head>
<body>
  <p class="myfont">I Love my Country!!!</p>
</body>
</html>
```

Output



3.7.7 Text Decoration

Using text decoration property we can include special features in the text. Various properties of text decoration are

- underline
- overline
- line-through

HTML Document[TxtDecor.html]

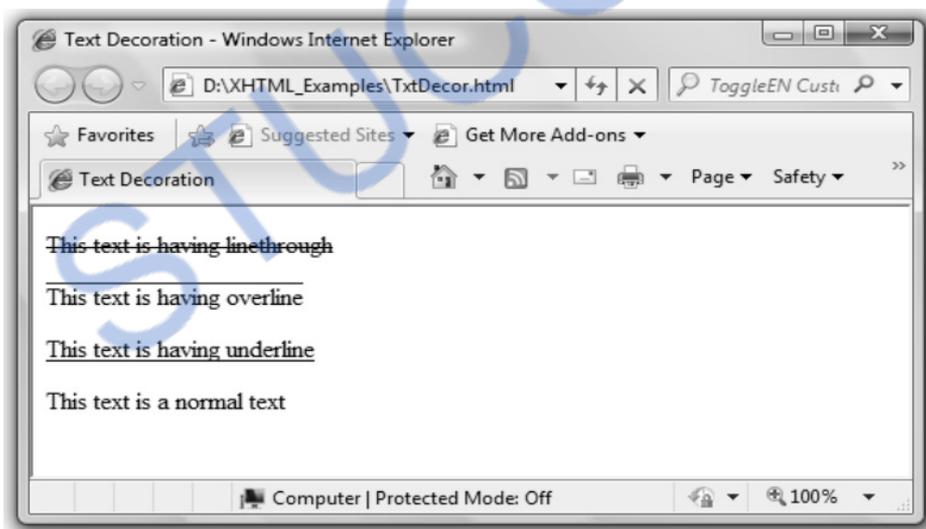
```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Text Decoration</title>
<style type="text/css">
p.linethrough {text-decoration:line-through;}
p.underline {text-decoration:underline;}
p.overline {text-decoration:overline;}
p.normal {text-decoration:normal;}

</style>
</head>
<body>
  <p class="linethrough"> This text is having linethrough </p>
  <p class="overline"> This text is having overline </p>
  <p class="underline"> This text is having underline </p>
  <p class="normal"> This text is a normal text </p>

</body>
</html>
```

Output



3.7.8 Alignment of Text

Using cascading style sheets we can align the text. This alignment can be done using following properties such as -

Property	Value	Meaning
text-align	left	Aligning the text to the left.

text-align	right	Aligning the text to the right
text-align	center	Alignment of text at the centre.
text-indent	value in inches	Desired indentation of the text can be applied.

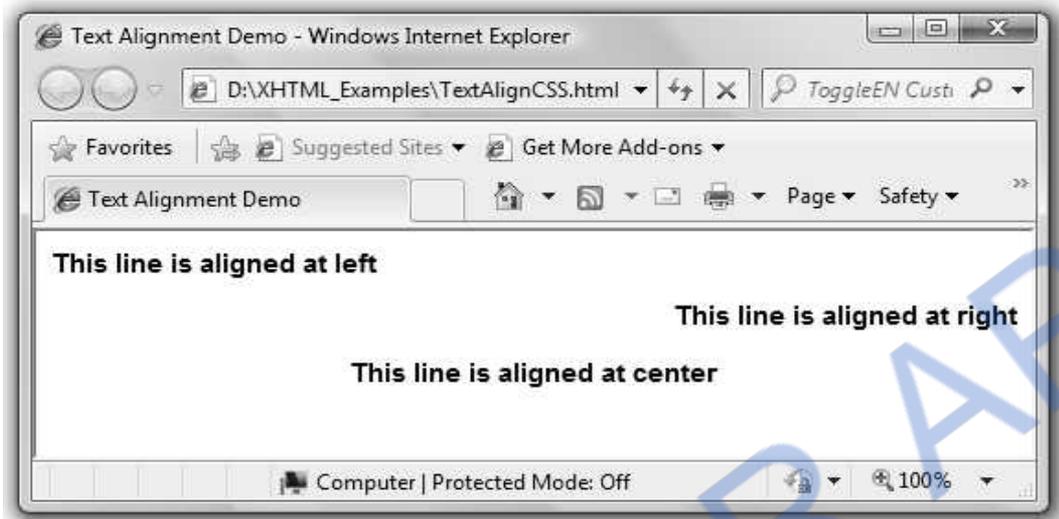
Following is script which uses various text alignment properties.

HTML Document[TextAlignCSS.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Text Alignment Demo</title>
    <style type="text/css">
      h1
      {
        font-family:Arial;
        font-size:15px;
        text-align:left;
      }
      h2
      {
        font-family:Arial;
        font-size:15px;
        text-align:right;
      }
      h3
      {
        font-family:Arial;
        font-size:15px;
        text-align:center;
      }
    </style>
  </head>

  <body>
    <h1>This line is aligned at left</h1>
    <h2>This line is aligned at right</h2>
    <h3>This line is aligned at center</h3>
  </body>
</html>
```

Output



University Question

1. Explain any eight CSS text properties.

AU : Dec.11, Marks 8

3.8 Box Model

The box comes in picture when we put the contents within some rectangle. The box model represents the contents, inner edge, outer edge, margins and padding. It is as shown in Fig. 3.8.1.

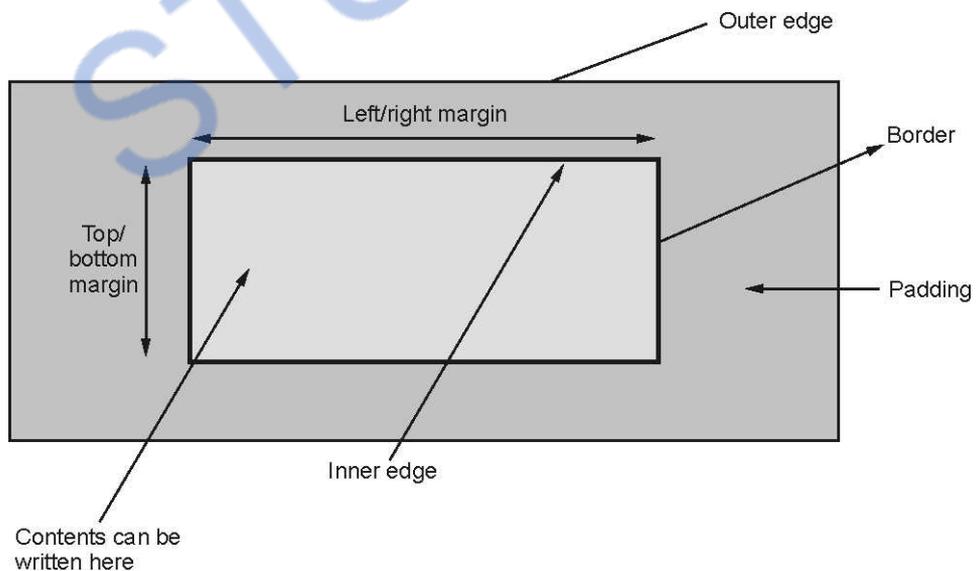


Fig. 3.8.1 The box model

Let us understand each of these terms along with suitable illustrations

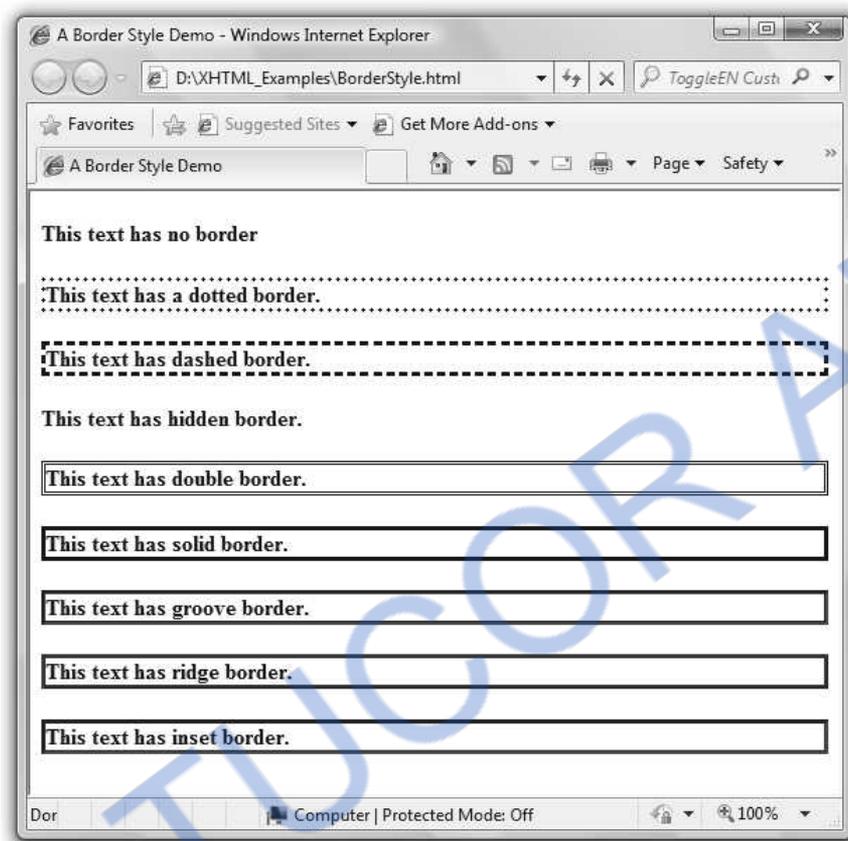
3.8.1 Borders

There is a property called border-style which controls the element's content border.

HTML Document[BorderStyle.html]

```
<!DOCTYPE html >
<html >
  <head>
    <title>A Border Style Demo</title>
    <style type="text/css">
      h4.none {border-style:none}
      h4.dotted {border-style:dotted}
      h4.dashed {border-style:dashed}
      h4.solid {border-style:solid}
      h4.double {border-style:double}
      h4.groove {border-style:groove}
      h4.ridge {border-style:ridge}
      h4.inset {border-style:inset}
      h4.outset {border-style:outset}
      h4.hidden {border-style:hidden}
    </style>
  </head>
  <body>
    <h4 class="none">This text has no border</h4>
    <h4 class="dotted">This text has a dotted border.</h4>
    <h4 class="dashed">This text has dashed border.</h4>
    <h4 class="hidden">This text has hidden border.</h4>
    <h4 class="double">This text has double border.</h4>
    <h4 class="solid">This text has solid border.</h4>
    <h4 class="groove">This text has groove border.</h4>
    <h4 class="ridge">This text has ridge border.</h4>
    <h4 class="inset">This text has inset border.</h4>
  </body>
</html>
```

Output



There is another property and that is **border-width** which defined the width of the border. The width can be specified in terms of pixels. The border-width property is illustrated in the following example.

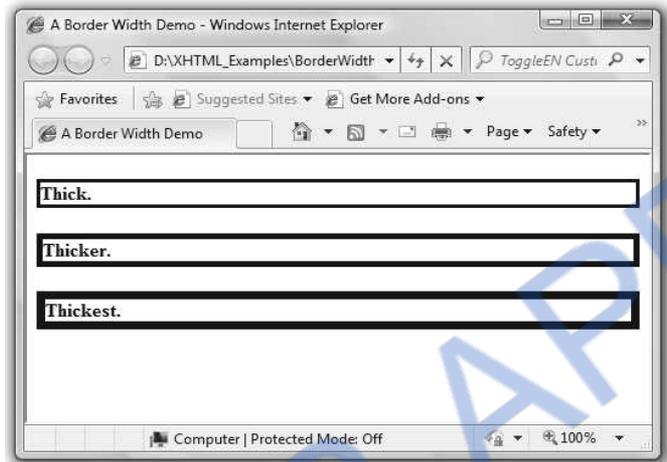
HTML Document[BorderWidth.html]

```
<!DOCTYPE html >
<html >
<head>
<title>A Border Width Demo</title>
<style type="text/css">
h4.thick
{
border-style:solid;
border-width:3px;
}
```

```

h4.thicker
{
border-style:solid;
border-width:5px;
}
h4.thickest
{
border-style:solid;
border-width:7px;
}
</style>
</head>
<body>
<h4 class="thick">Thick.</h4>
<h4 class="thicker">Thicker.</h4>
<h4 class="thickest">Thickest.</h4>
</body>
</html>

```

Output

We can make the borders colourful using **border-color** property. Following script displays the colored borders.

HTML Document[BorderColor.html]

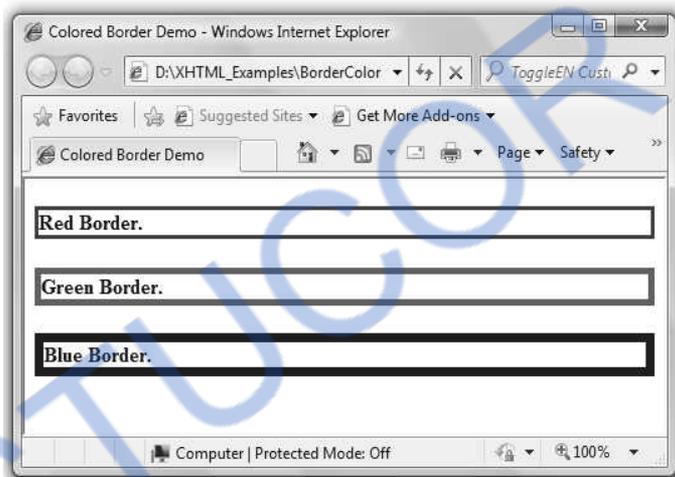
```

<!DOCTYPE html>
<html>
<head>
<title>Colored Border Demo</title>
<style type="text/css">
h4.thickred
{
border-style:solid;
border-width:3px;
border-color:red;
}
h4.thickergreen
{
border-style:solid;
border-width:5px;
border-color:green;
}
h4.thickestblue
{
border-style:solid;

```

```
border-width:7px;
border-color:blue;
}
</style>
</head>
<body>
<h4 class="thickred">Red Border.</h4>
<h4 class="thickergreen">Green Border.</h4>
<h4 class="thickestblue">Blue Border.</h4>
</body>
</html>
```

Output



We can also specify the individual side of the box by some particular style. For example we can display a box with the solid border on the top and right, dashed border on the bottom and dotted border on the left. Thus any possible combinations with all four sides can be made using the individual border style property.

Hence we can use -

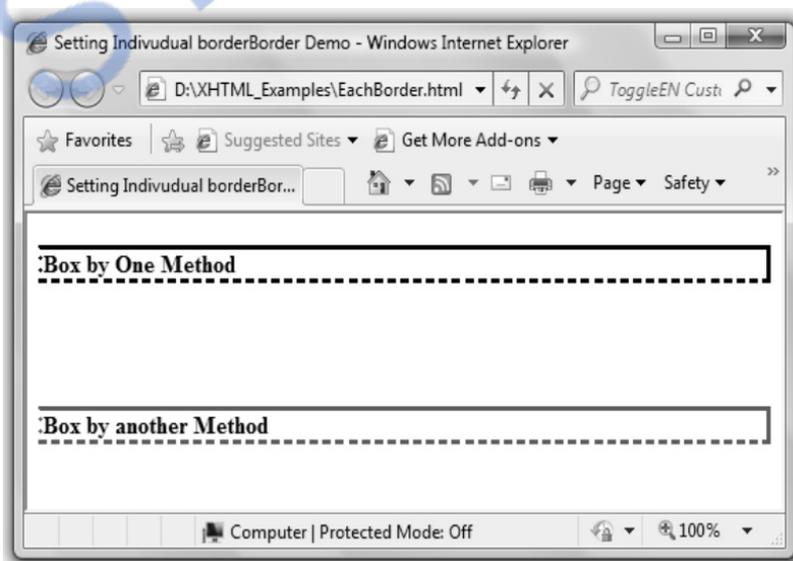
- border-top-style:solid;
- border-right-style:solid;
- border-bottom-style:dashed;
- border-left-style:dotted;

OR

- border-style:solid solid dashed dotted

HTML Document[EachBorder.html]

```
<!DOCTYPE html>
<html>
<head>
<title>Setting Individual borderBorder Demo</title>
<style type="text/css">
h4.method1
{
border-top-style:solid;
border-right-style:solid;
border-bottom-style:dashed;
border-left-style:dotted;
}
h4.method2
{
border-style:solid solid dashed dotted;
border-color:green;
}
</style>
</head>
<body>
<h4 class="method1">Box by One Method</h4>
<br/><br/>
<h4 class="method2">Box by another Method</h4>
</body>
</html>
```

Output

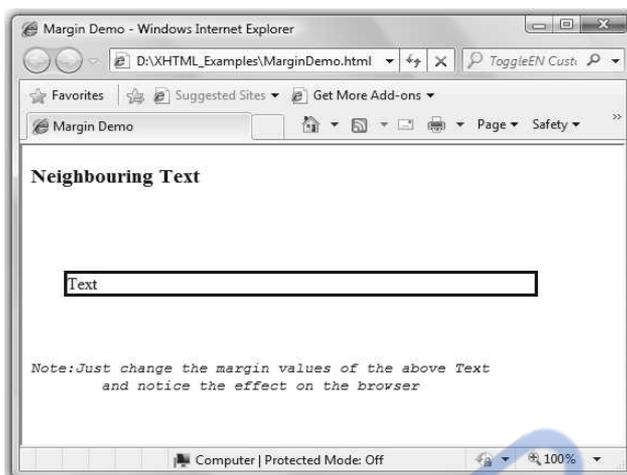
3.8.2 Margins and Padding

Margin means the space between the content and its neighbouring content. There can be top margin, bottom margin, left margin and right margin. Hence various properties of margins are margin-top, margin-bottom, margin-left and margin-right. The values of these properties can be given in **px** and **in**.

HTML Document[MarginDemo.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Margin Demo</title>
    <style type="text/css">
      p.method1
      {
        border-style:solid;
        margin-top:80px;
        margin-bottom:60px;
        margin-left:30px;
        margin-right:70px;
      }
    </style>
  </head>
  <body>
    <h3>Neighbouring Text</h3>
    <p class="method1">Text</p>
  </em>
  <pre>Note:Just change the margin values of the above Text
    and notice the effect on the browser
  </pre>
  </em>
</body>
</html>
```

Output



Just change the values of the properties such as margin-top, margin-bottom, margin-left and margin-right of the above given XHTML document and observe the change in the web browser. Thus by experimenting the above script in this way, you can understand the actual meaning of the margin.

Padding means the space between the contents and its border. Various properties of padding are padding-left, padding-right, padding-top and padding bottom. These values can also be given in **px** or in **in**.

HTML Document[PaddingDemo.html]

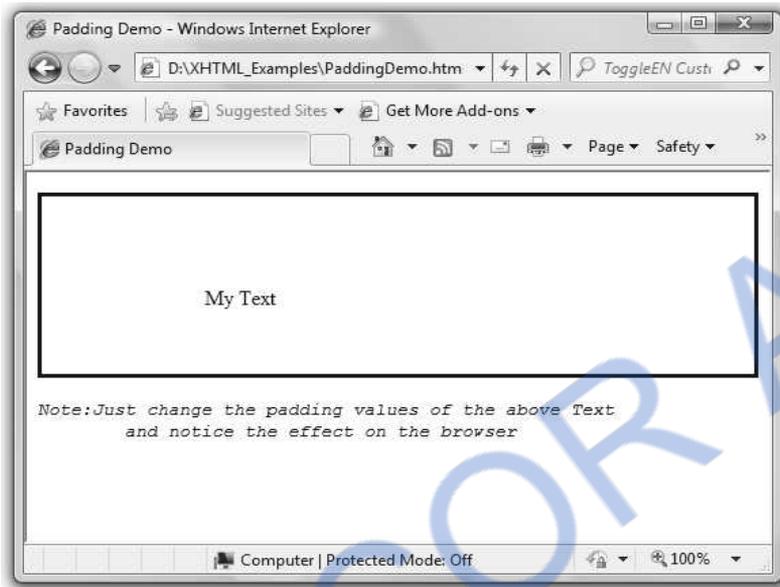
```

<!DOCTYPE html>
<html>
  <head>
    <title>Padding Demo</title>
    <style type="text/css">
      p.method1
      {
        border-style:solid;
        padding-top:70px;
        padding-bottom:50px;
        padding-left:120px;
        padding-right:60px;
      }
    </style>
  </head>
  <body>
    <p class="method1">My Text</p>
  </em>
  <pre>Note:Just change the padding values of the above Text
    and notice the effect on the browser
  </pre>
</em>

```

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

Output



Just change the values of the properties such as padding-top, padding-bottom, padding-left and padding-right of the above given XHTML document and observe the change in the web browser. Thus by experimenting the above script in this way, you can understand the actual meaning of the padding.

Review Questions

1. Explain the box modeling of CSS

3.9 Color

3.9.1 Color Groups

There are three groups of colors. Those are as given below -

1. This is the smallest group of colors. In this group there are 16 colors. These are named colors. The colors in this group are enlisted as follows.

Sr. No.	Color Name	Hexadecimal Value
1	black	000000
2	red	FF0000
3	lime	00FF00

4	blue	0000FF
5	yellow	FFFF00
6	fuchsia	FF00FF
7	aqua	00FFFF
8	white	FFFFFF
9	silver	C0C0C0
10	gray	808080
11	maroon	800000
12	purple	800080
13	green	008000
14	olive	808000
15	navy	000080
16	teal	008080

These colors get displayed as it is on the web browsers. And all the web browsers support these colors.

2. There is a set of **216** colors which is called **web palette**. The elements of such colors are red, green and blue. The values of these elements can be 00, 33, 66, 99, CC and FF.
3. These are 16 millions colors. These are 24-bit colors.

3.9.2 Color Properties

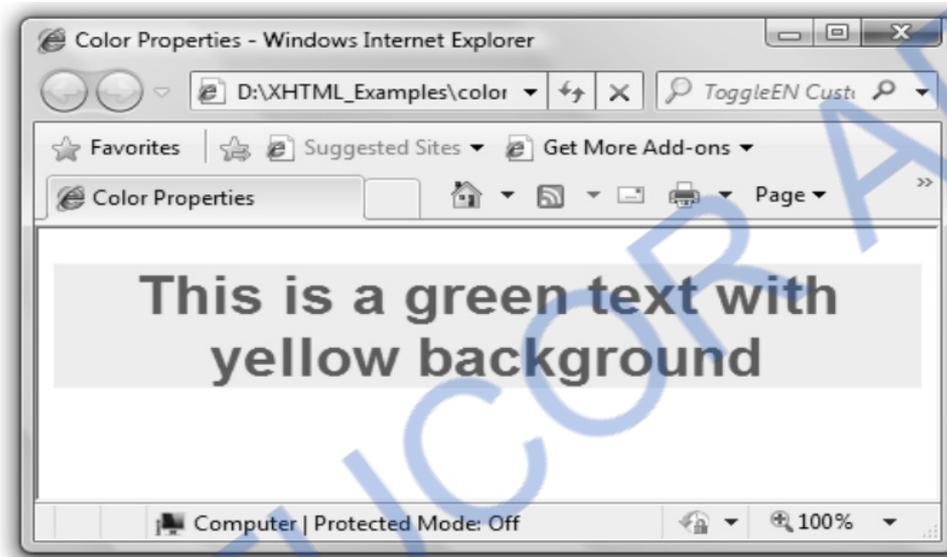
Using **color** property we can set the foreground color and using **background-color** the background color can be set. The use of these properties is shown in XHTML document.

HTML Document [colorProperties.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Color Properties</title>
    <style type="text/css">
      h1
      {
        font-family:Arial;
        color:green;
        background-color:yellow;
      }
    </style>
  </head>
```

```
<body>
  <center>
    <h1>This is a green text with yellow background</h1>
  </center>
</body>
</html>
```

Output



3.10 Background Image

Using **background-image** property an image can be set as background.

HTML Document[Backimg.html]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Background Image</title>
    <style type="text/css">
      body {background-image:url(victoriafalls.jpg);
        background-repeat:no-repeat; }
      p
      {
        font-family:Arial;
        font-size:20px;
      }
    </style>
  </head>
```

```
<body>
<center>
<h2>Wonder of the World</h2>
</center>
<p>
```

The Vicorial waterfall is one of the greatest waterfalls in the world.

It lies on the Zimbezi river which is between the border of Zambia and Zimbabwe.

The Victoria fall can be seen from 25 to 40 miles away.

The roaring of the water can be heard from a long distance.

The native people call it as “mosi-oa-tunya”, that means “smoke that thunders”.

Victoria falls was discovered by David Livingstone in 1855.

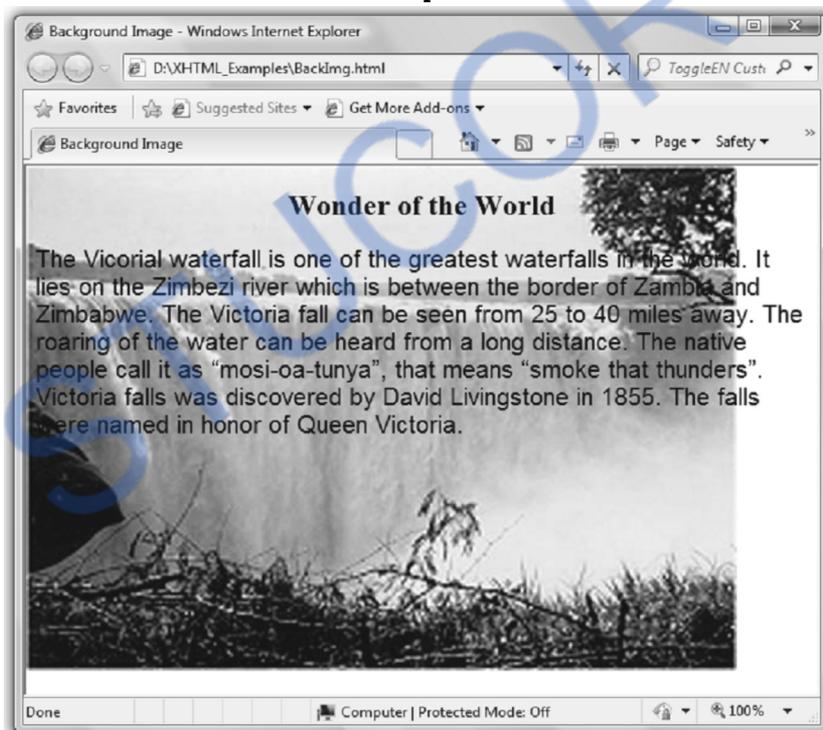
The falls were named in honor of Queen Victoria.

```
</p>
```

```
</body>
```

```
</html>
```

Output



It is expected that the foreground text over the image should be visible. That means the color of the background image must not be the same as the color of the text written over it.

- **Background-repeat property**

The **background-image** property sets the background image with repetition. If we want to control the repetition of the image then we must set the **background-repeat** property. The

values of this property could be no-repeat, repeat-x, repeat-y or repeat(default). The repeat-x means the background image gets repeated over the x-axis(horizontally) and repeat-y means the background image gets repeated over the y-axis(vertically).

- **Background-position property**

Various values for back-ground position property could be **top**, **bottom**, **left** and **right**

Ex. 3.10.1 : Write a CSS rule that places a background image halfway down the page, tilting it horizontally. The image should remain in place when the use scrolls up or down.

Sol. :<!DOCTYPE html>

```
<html>
<head>
<style type="text/css">
p
{
font-size:40px;
color:magenta;

}
body
{

background-image:url(baby.jpg);
background-repeat:repeat-x;
background-attachment:fixed;
}
</style>
</head>
<body>
<p>
Twinkle, twinkle, little star, How I wonder what you are.Up above the world so high, Like a
diamond in the sky.Twinkle, twinkle, little star, How I wonder what you are!
When the blazing sun is gone, When there's nothing he shines upon,Then you show your little
light, Twinkle, twinkle, through the night.
Twinkle, twinkle, little star, How I wonder what you are!
In the dark blue sky so deep Through my curtains often peep For you never close your eyes
Til the morning sun does rise
Twinkle, twinkle, little star How I wonder what you are Twinkle, twinkle, little star How I
wonder what you are
</p>
</body>
</html>
```

Ex. 3.10.2 : Create a web page to create a fixed background image.

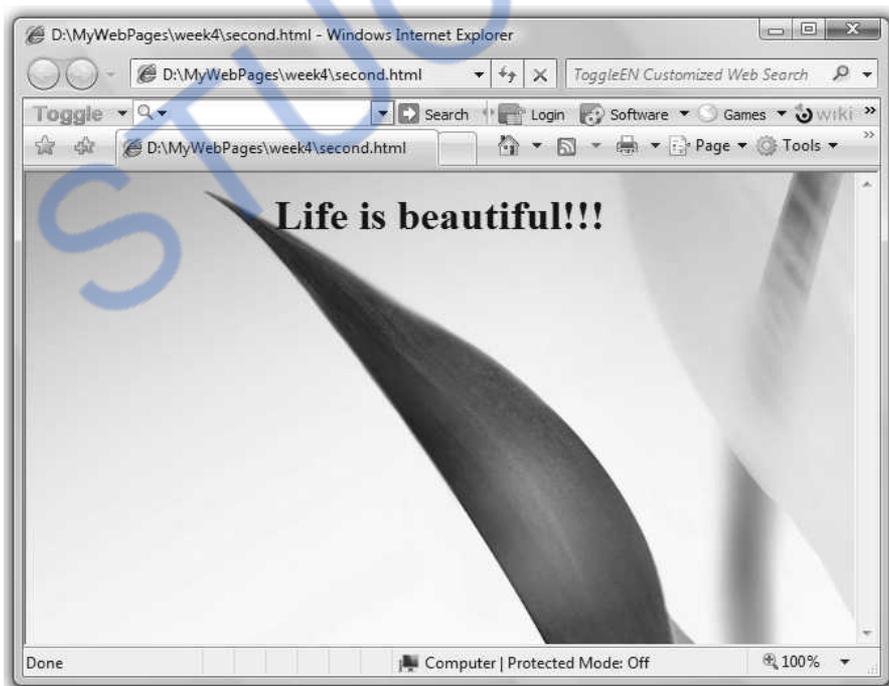
AU : May-09, Marks 2

Sol. :

```
<html>
  <head>
    <style type="text/css">
      body
      {
        background-image:url("Myimg.jpg");
        background-repeat:no-repeat
      }
    </style>
  </head>
  <body>
    <center>
      <h1>Life is beautiful!!!</h1>
    </center>
  </body>
</html>
```

No repeat creates fixed background image

Output



Ex. 3.10.3 : Write a CSS which adds background images and indentation.

AU : May-13, Marks 8

Sol. :

```
<html>
  <head>
    <style type="text/css">
      body
      {
        background-image:url("Rose.jpg");
        background-repeat:no-repeat
      }
      p {text-indent:50px;}
    </style>
  </head>
  <body>
    <center>
      <p>Roses are beautiful.</p>
    </center>
  </body>
</html>
```

3.11 Normal Flow Box Layout

- Normal flow is the default scheme used for positioning. It applies to any element of the web document.
- In this scheme, the two boxes are used -
 1. Block box
 2. Inline box.
- The block boxes flow vertically starting at the top of their containing block with each placed directly below the preceding one. Inline boxes flow horizontally from left to right.

3.11.1 Basic Box Layout

- In any XHTML document, the `<html>` is the root element. Corresponding to this element a browser generates a box called **initial containing block**. If browser's horizontal and vertical scroll bars are not active then the **client area** and the **initial containing block** become one and the same.
- It is assumed that the web document is placed on an imaginary area called **canvas**. The browser client area acts as a **viewport** to view the portion of the web document.
- All the relative CSS boxes will be added to the initial containing block. For instance the box corresponding to the body element will be placed inside the **initial containing block**. Thus if B element is inside the element A then the box corresponding to B will be placed inside the box corresponding to A which in turn will be inside the **initial containing block box**. Thus the

normal flow processing of the blocks occurs from outer block to inner. This behaviour is known as **normal flow processing of boxes**.

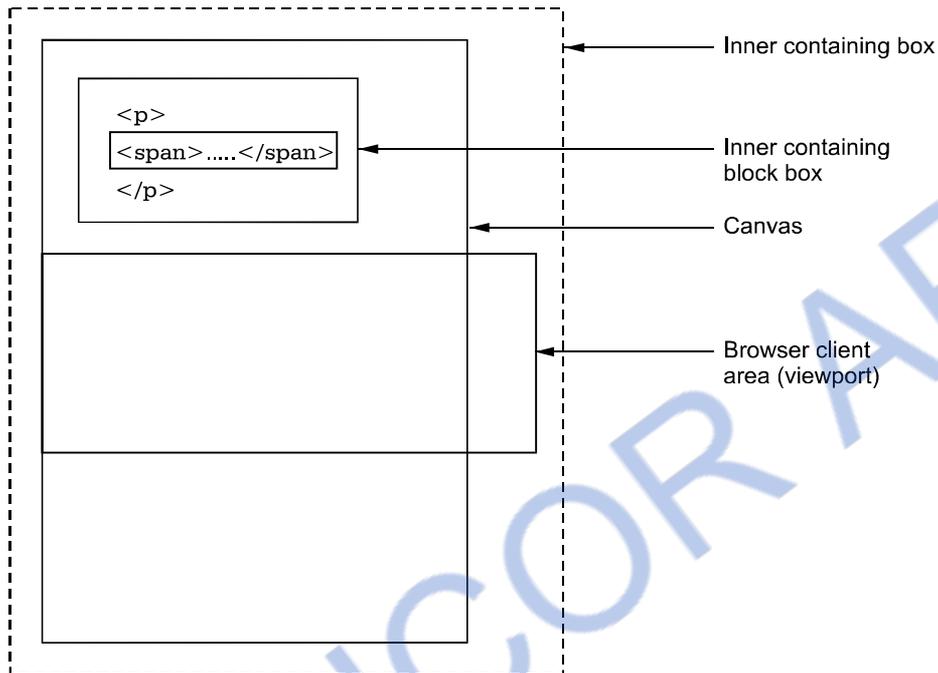


Fig. 3.11.1 Normal flow processing

- If the browser window is changed then the **width** of the block box of the body element gets changed and accordingly the **width** of the other block elements gets changed. The **height** of the block box varies according to the contents of the elements.

3.11.2 The Display Property

- The CSS defines the following HTML elements as the block elements - body, div, dt, dd, fieldset, form, frame, frameset, hr, html, h1, h2, h3, h4, h5, h6, ol, p, pre and ul.
- Examples of **display** property are -

```
p {display:block}
em{display:inline}
pre{display:block}
table{display:table}
```

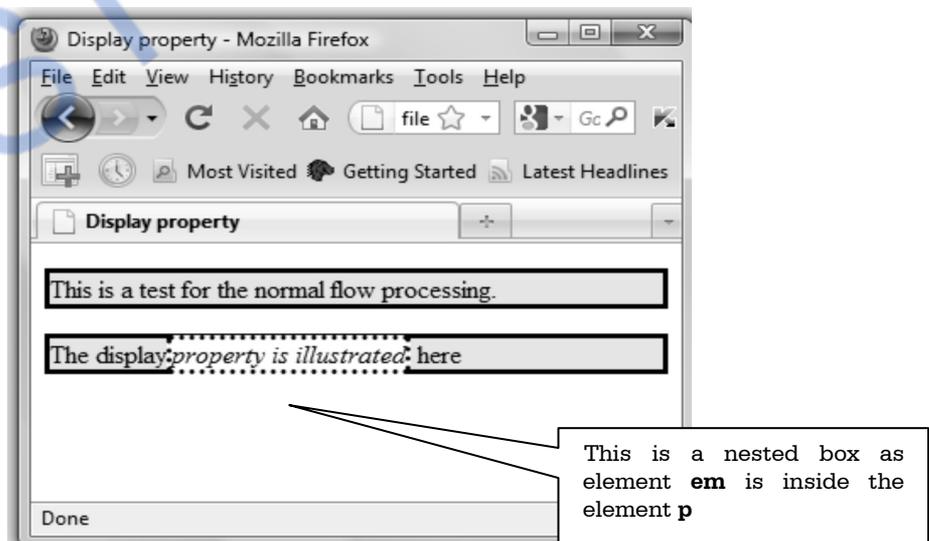
- Following script illustrates the use of display property.

XHTML document[DisplayDemo.html]

```
<!DOCTYPE html >
<html>
<head>
```

```
<title>Display property</title>
<style type="text/css">
  p
  {
    background-color:yellow;
    display:block;
    border:solid
  }
  em
  {
    background-color:white;
    display:inline;
    border:dotted
  }
</style>
</head>
<body>
  <p>
    This is a test for the normal flow processing.
  </p>
  <p>
    The display<em>property is illustrated</em> here
  </p>
</body>
</html>
```

Output



3.11.3 Margin Collapse

When two consecutive **block boxes** are displayed on the browser window then the separation between these block boxes is defined by a special rule called **margin collapse**. According to this rule, the bottom margin of the upper box and the top margin of the lower box are collapsed in a single margin.

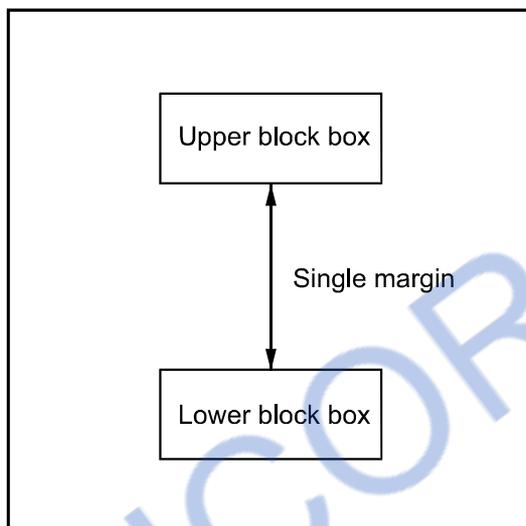


Fig. 3.11.2 Margin collapse

3.11.4 Inline Box

The HTML elements **span** and **strong** are treated as the inline elements. Hence the boxes corresponding to these elements are called **inline boxes**. Using the display property one can define the inline box. The **inline elements** are those elements that do not form new blocks of content; the content is distributed in lines. The inline boxes may contain some text or some images.

XHTML Document[inlineblock.html]

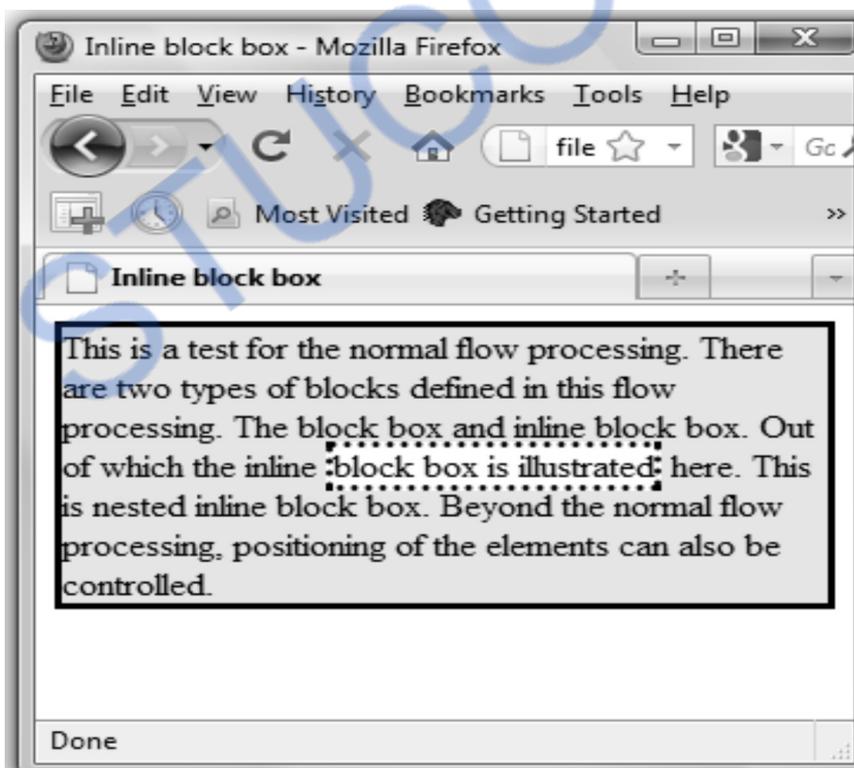
```
<!DOCTYPE html >
<html>
<head>
<title>Inline block box</title>
<style type="text/css">
div
{
background-color:yellow;
border:solid
}
}
```

```
span
{
background-color:white;
display:inline;
border:dotted
}
```

The inline block box corresponds to the element **span**

```
</style>
</head>
<body>
<div>
This is a test for the normal flow processing.
There are two types of blocks defined in this flow processing.
The block box and inline block box.
Out of which the inline <span>block box is illustrated</span> here.
This is nested inline block box.
Beyond the normal flow processing, positioning of the elements can also be controlled. </div>
</body>
</html>
```

Output

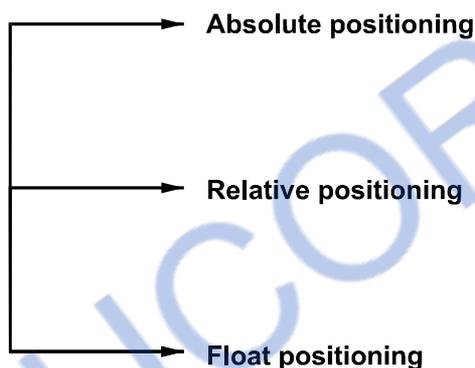


Review Questions

1. What is normal flow processing of boxes ?
2. Write an XHTML document that illustrates the display property.
3. What do you understand by the term margin collapse ?
4. What is inline element ?

3.12 Beyond Normal Flow

Using CSS a web designer can have a full control over positioning of the elements. For that purpose a **position** property is used. CSS allows to control the position of the element using three methods -



Let us discuss each of them with illustrative examples.

3.12.1 Absolute Positioning

Following is an illustration positioning the element using absolute positioning.

XHTML document [Position_Demo1.html]

```
<!DOCTYPE html >
<html >
<head>
  <title> </title>
  <style type="text/css">
    .text_pos
    {
    position:absolute;
    left:150px;
    top:-10px;
    font-family:script;
    font-size:50px;
    width:100px;
    }
  </style>
</head>
</html >
```

```
</style>
</head>
<body>
<p>Taj Mahal is one of the wonders of the World. It is located in
```

Agra,India.Taj Mahal is built by Shah Jahan in memory of his wife

Mumtaz Mahal.

```
</p>
<p>It is considered to be the excellent example of Mughal
```

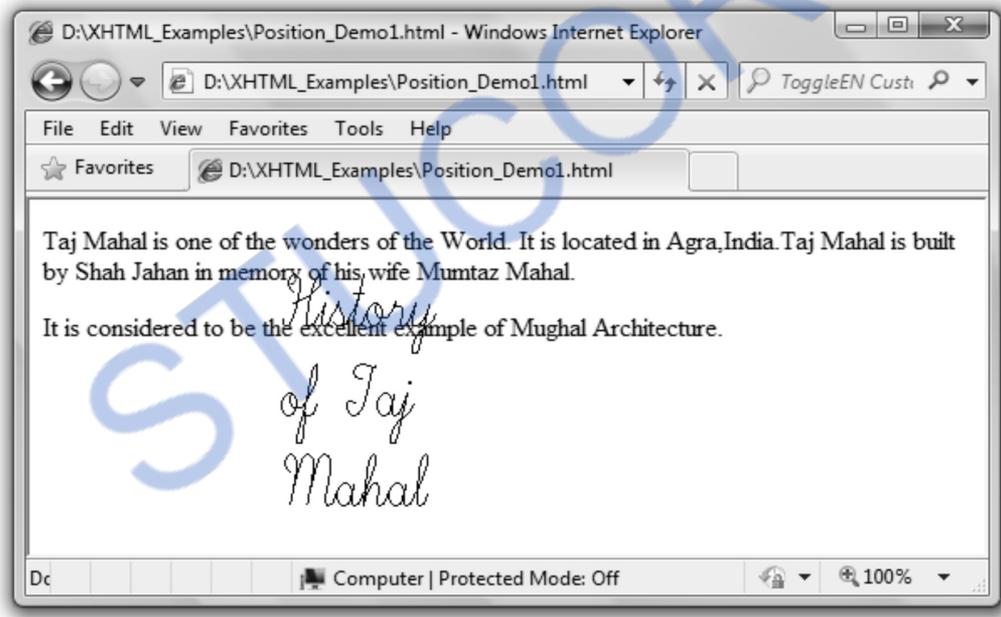
Architecture.</p>

```
<p class="text_pos">History of Taj Mahal</p>
```

```
</body>
```

```
</html>
```

Output



Note that we have used a class named “text_pos” for positioning the text “History of Taj Mahal”. In this class we have used **left** and **top** values for denoting the space from left and top within the browser window. The absolute positioning defines a new co-ordinate system for positioning of child elements. This co-ordinate system has its origin at the top, left corner of the element. The position is measured from the upper left corner of the browser window. The element can be absolutely positioned inside another positioned element.

The width property is for uniformly embedding the text inside another element. If the width is not specified, the element will extend to immediately inside the right outer edge of the parent element. The height, if not specified, will be just large enough for the contents of the element. For understanding the absolute positioning just change the left, top and width values from above script and notice the changes in the browser window.

3.12.2 Relative Positioning

If we assign the value to the position attribute as **relative** then the corresponding element gets placed at the relative position to the previous contents. For instance if there is a paragraph which is starting from leftmost and topmost position in the browser's window and if we want to place the element <h1> with top:70px then <h1> will be displayed 70 pixels from paragraph. Thus relative positions get decided with respect to the previous contents.

Following is an example in which relative positioning is used.

XHTML Document[Position_Demo2.html]

```
<!DOCTYPE html >
<html >
<head>
  <title>Relative and Absolute positioning</title>
  <style type="text/css">
    .text_pos1
    {
    position:relative;
    top:10px;
    font-family:script;
    font-size:30px;
    width:400px;
    }
    .text_pos2
    {
    position:absolute;
    top:10px;
    font-family:Arial;
    font-size:30px;
    width:400px;
    }

  </style>
</head>
<body>
```

```
<p>Taj Mahal is one of the wonders of the World. It is located in Agra,India.Taj Mahal is built by Shah Jahan in memory of his wife Mumtaz Mahal.
```

```
</p>
```

```
<p>It is considered to be the excellent example of Mughal Architecture.</p>
```

```
<p class="text_pos1">This is relative position</p>
```

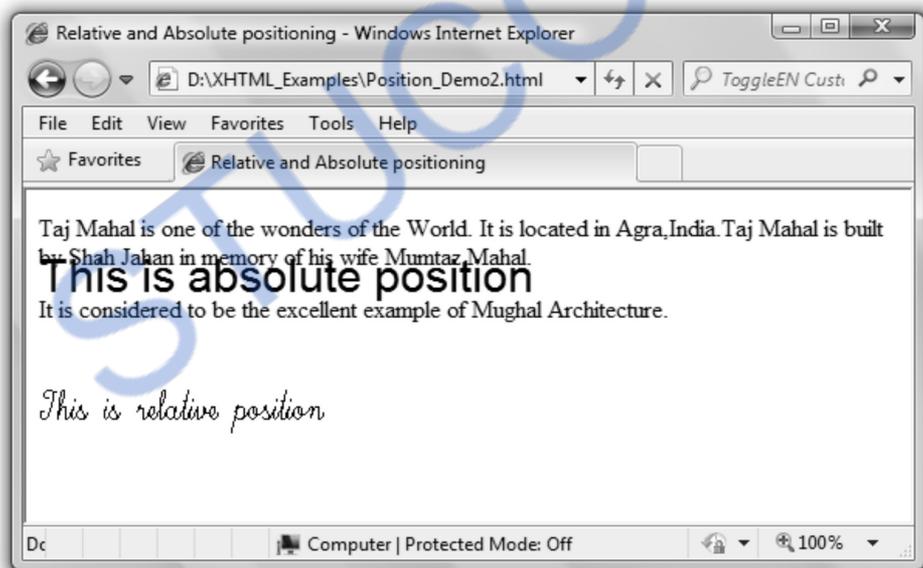
```
<p class="text_pos2">This is absolute position</p>
```

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

Script Explanation

In above script, we have defined two classes **text_pos1** and **text_pos2**. Both the classes have the same positional values i.e. left and top. Both are having same font-size and width parameter. Only font-family is changed to differ the look of both the elements. And the **text_pos1** has the position property set to relative position and **text_pos2** has the position property which is set to absolute position. Hence text_pos1 (relative positioning) will be placed at the relative position from the previous text which is denoted by `<p>` tag. And absolute positioning will be the default positioning which does not consider the position of previous text. This difference between relative and absolute positioning can be well understood by following output.

Output



3.12.3 Float Position

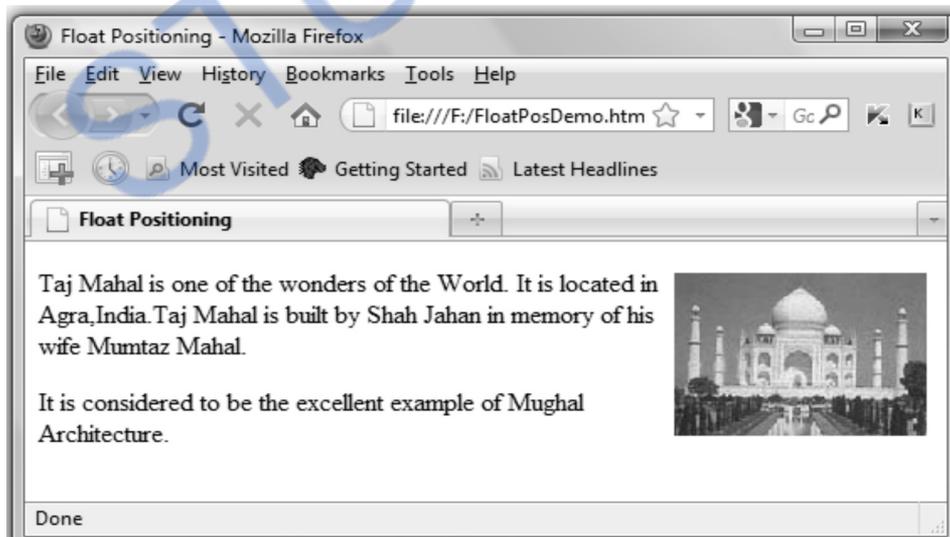
The float position is used to shift an image to one side and wrap the text around it.

In the following example, the image is shifted to the right and the text is wrap around it. We have used **float:right** for this pupose. If one wants to shift the image to the left he can simply declare the style as **float:left**.

XHTML Document[FloatPosDemo.html]

```
<!DOCTYPE html >
<html >
<head>
<title>Float Positioning</title>
<style type="text/css">
img.Rightfloat
{
float:right;
margin: 4px;
}
</style>
</head>
<body>

<p>Taj Mahal is one of the wonders of the World. It is located in
Agra,India.Taj Mahal is built by Shah Jahan in memory of his wife
Mumtaz Mahal.
</p>
<p>It is considered to be the excellent example of Mughal
Architecture.</p>
</body>
</html>
```

Output

Review Questions

1. What is CSS positioning? Explain it with illustrative examples.
2. List and explain the various positioning schemes in detail.

3.13 CSS 3.0

- The CSS3 is a latest version of CSS.
- It is completely backward compatible with older version of CSS.
- It support for selectors, box model, Background and borders, Text effects, animations, Multiple column layout, 3D transformations and so on.

Let us discuss some remarkable features introduced in CSS3.

3.13.1 CSS Border

CSS3 allows to create following types border

1. border-radius
2. box-shadow
3. border-image

Let us discuss them along with the necessary illustrations.

1. The border-radius

The **border-radius** property allows the developer to create round corners in the design. If we set a single value to border-radius then that means all the round corners will be of same value. For example - Following line represents that all the round corners are of 10px radius.

```
border-radius: 10px;
```

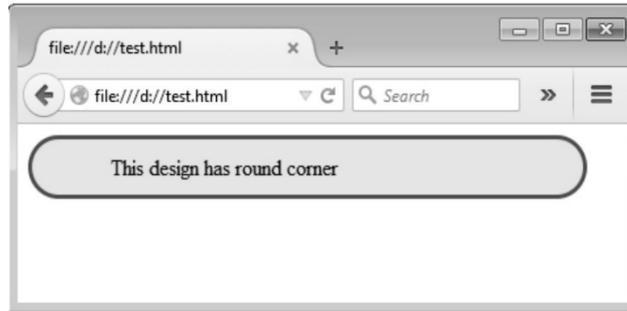
Following script represents this property

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
border: 3px solid red;
padding: 10px 60px;
background: yellow;
width: 300px;
border-radius: 30px;
}
</style>
</head>
<body>
<div>This design has round corner</div>
```

All four corners are of radius of 30pixel.

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

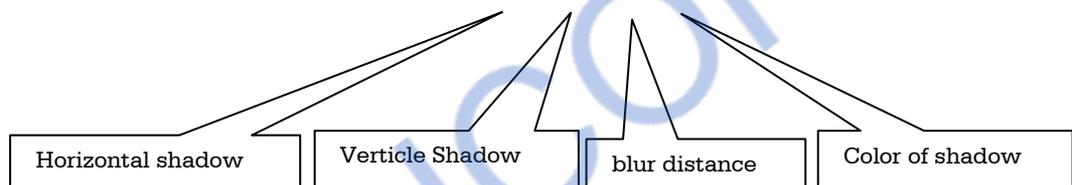
Output



2. The border shadow

The **border-shadow** property represents the shadows around the border. For example-

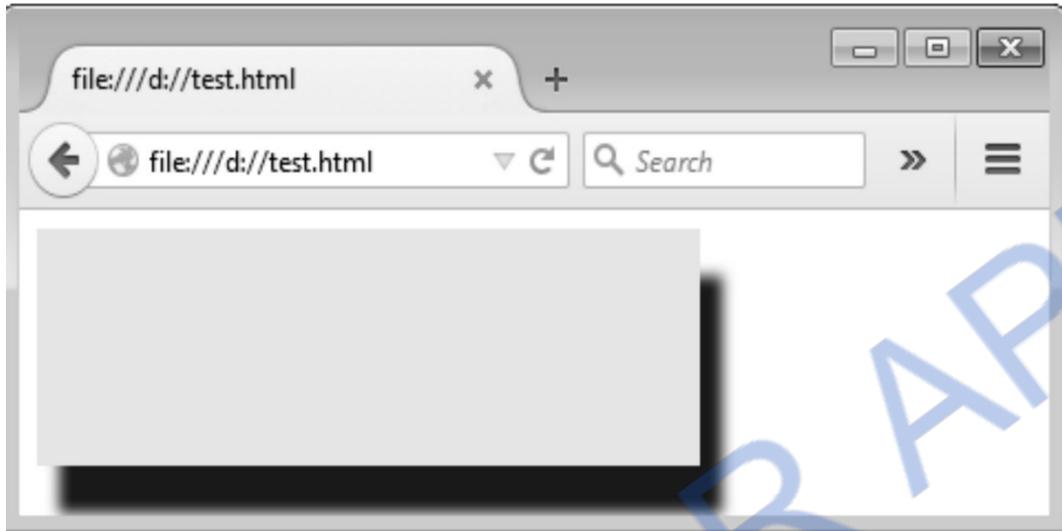
border-shadow : 10x 20x 5x black



Following script represents this property

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
width: 300px;
height: 100px;
background-color: yellow;
box-shadow: 10px 20px 5px blue;
}
</style>
</head>
<body>
<div> </div>
</body>
</html>
```

Output

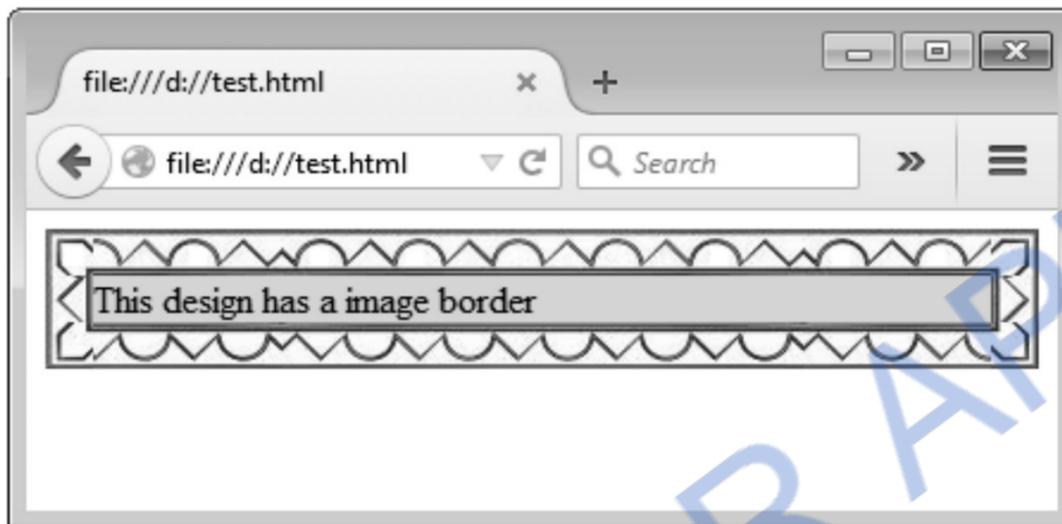


3. The border-image

The **border-image** property is for setting the image as a border. Following example illustrates the idea

```
<!DOCTYPE html>
<html>
<head>
<style>
  div {
    background-color: pink;
    border: 20px solid transparent;
    border-image-source: url('flower.png'); //image as a border
    // The inward offsets of each side of image-border is specified by image-slice
    border-image-slice: 20;
    border-image-repeat: repeat; //repeat the image border
  }
</style>
</head>
<body>
  <div>
    This design has a image border
  </div>
</body>
</html>
```

Output



Explanation

The image file can be provided to the **border-image-source** style using the **url**. The **border-image-slice** specifies the inward offset of each side of the border image. It can be denoted by a value or percentage. The syntax for border-image-slice is given by following example.

```
border-image-slice: 20% 30% 30% 20%; /* top right bottom left*/
```

If a single value is provided then that means all the four corners are sliced using the same number.

By the **border-image-repeat** property the slice gets repeated from the center of the side to fill the space.

3.13.2 CSS3 Text Effects

In this section we will discuss various text effects mainly used in CSS3.

1. Text Shadow Effect

The CSS3 allows to incorporate many advanced text effects in the web design. Out of which the text shadow effect is very commonly used.

The **text-shadow** property can be as given below -

```
text-shadow: horizontal_shadow vertical_shadow blur_distance color_of_shadow
```

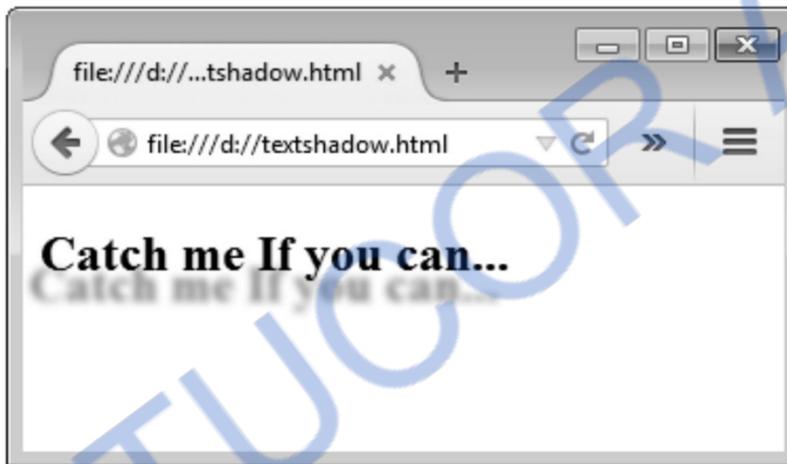
Let us now understand how to apply and experience this text style.

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
  h2 {
    text-shadow: -5px 15px 3px gray;
  }
</style>
</head>
<body>

<h2>Catch me If you can...</h2>
</body>
</html>
```

Output



2. Text Wrap Effect

Another useful effect is text wrap effect. By this effect the text in a paragraph gets wrapped on the next line. This effect can be achieved by breaking the word -

`word-wrap: break-word`

Following is a simple CSS script in which the text wrap effect is used.

```
<!DOCTYPE html>
<html>
<head>
<style>
  div.wrapdemo
  {
    width: 130px;
    border: 1px solid red;
    word-wrap: break-word;
  }
</style>
```

```
</head>
```

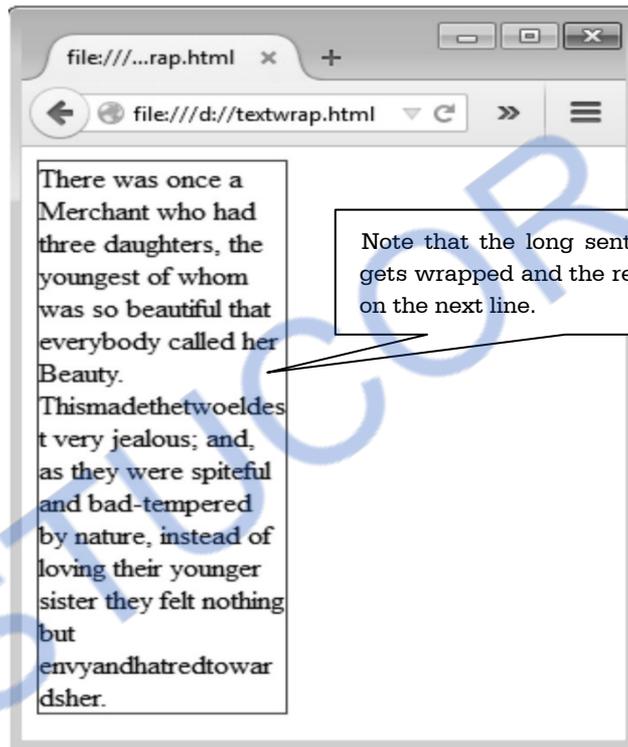
```
<body>
```

```
<div class="wrapdemo"> There was once a Merchant who had three daughters, the youngest of whom was so beautiful that everybody called her Beauty. This made the two eldest very jealous; and, as they were spiteful and bad-tempered by nature, instead of loving their younger sister they felt nothing but envyandhatredtowardsher.</div>
```

```
</body>
```

```
</html>
```

Output



Two Marks Questions with Answers

Q.1 What is CSS file ?

Ans. : The cascading style is many times defined in a separate file. This file is called CSS file. In CSS file one or more style rules are given.

Q.2 Give the syntax of CSS rule.

AU : Dec.-11, May-12

Ans. :

```
<style type="text/css">
  h1
```

```
{
  font-family:Arial;
  color:green
}
```

</style>

Q.3 What are style sheets ?

Ans. : The style sheets are the collection of styles that can be either embedded within the HTML documents or can be externally applied. The Cascading style sheet is a markup language used to apply the styles to HTML elements.

Q.4 What is selector string ? Specify any three forms of selectors.

Ans. : The ruleset in CSS consists of selector string which is basically an HTML element. These selectors can be defined with the help of properties and values.

Q.5 How do you provide uniform access to structured documents in diverse applications ?**AU : Dec.-08**

Ans. : Style sheets allow the user to decide the style of presentation. This presentation can be defined in a separate file. And simply by including this file in various web documents the designated style can be applied to the corresponding tag. For example - If we define

```
h1
{
  font_size : 20 pt
  color : red
}
```

Then every line defined by h1 tag will be of size 20 pt and red in color.

Q.6 Mention the need for cascading style sheet.**AU : May-11**

Ans. : The cascading style sheet allows separation between the information contained in a document and its presentation. Hence any change in presentation can be made without disturbing the information of the document. The cascading style sheet allows the developer to give the consistent appearance to all elements of the web page.

Q.7 What do you mean by the term ruleset ?

Ans. : The styles are defined in separate file. This file is called CSS file. The collection of rules in a CSS file is called **ruleset**.

Q.8 Enlist various categories of properties used in CSS

Ans. : The various categories of properties in CSS are -

- 1) Fonts
- 2) List
- 3) Alignment of Text
- 4) Color
- 5) Margins
- 6) Background
- 7) Borders.

Q.9 What is the use of universal selector ?

Ans. : Using the universal selector the values can be defined for all the elements in the document. It is denoted by *.

Q.10 What is generic class selector ?

Ans. : The generic class applied to any tag in the HTML document. And thus the values defined within that generic selector can be applied to the corresponding tag. The class selector must be preceded by the dot operator.

Q.11 Enlist the three commonly used pseudo classes.

Ans. : The three commonly used pseudo classes are -

1. focus
2. hover
3. hyperlink

Q.12 Define the hover element.

Ans. : The hover is associated with anchor element and the selector for this element is specified by a:hover. When the mouse is moved over the element then the pseudo class get activated.

Q.13 What are the advantages of external style sheet ?

Ans. : When we use external style sheet then the style is defined in one file and actual contents of the web page are defined in another file. Hence if we want to change the style of presentation of web page then we can simply modify the file in which the style is defined.

Q.14 What is the difference between the external style sheet and embedded style sheet ?

Ans. : The external style sheet is a kind of style sheet in which the styles are defined in a separate .css file and this file is mentioned at the beginning of the HTML document. When we need to apply the particular style to more than one web documents then the external style sheet is used. The embedded style sheet is a method in which the style is specified (or embedded) within the HTML document itself. It is not defined in separate file. Due to embedded style sheet unique style can be applied to all the elements.

Q.15 What is the significance of Box Model in CSS ?

Ans. : The box model in CSS represents the contents, inner edge, outer edge, margins and padding.

Q.16 What do you mean by margin and padding ?

Ans. : Margin means the space between the content and its neighbouring contents. Padding means the space between the contents and its border.

Q.17 What is the use of float position ?

Ans. : The float position is used to shift an image to one side and wrap the text around it.

Q.18 What is CSS padding ?

AU : Dec.-09

Ans. : CSS padding is a property that allows to have space between the border and actual content. This space can be at the top, bottom, left and right. For example - Following is a CSS script that creates padding around the sentence.

```
<html>
<head>
<style type="text/css">
h3
{
background-color:green;
}
h3.padding
{
padding-top:50px;
padding-bottom:50px;
padding-right:50px;
padding-left:50px;
}
</style>
</head>
<body>
<h3>This is a simple text without padding</h3>
<h3 class="padding">This text have padding around it</h3>
</body>
</html>
```

This text coil have padding at top bottom left and right

Q.19 If you do not want repeated background images then how will you apply the style ?

Ans. : For not repeated background images the following style can be set :

```
body
{
background-image:url("Myimg.jpg");
background-repeat:no-repeat
}
```

Q.20 What do you mean by the term inline element ?

Ans. : The inline elements are those elements that do not form new blocks of content. The content is distributed in lines.

Q.21 List two forms of style rules with an example.

AU : Dec.-12

Ans. : 1. Universal Selector

This selector is denoted by * (asterisk). This selector can be applied to all the elements in the document.

For example -

```
<style type="text/css">
```

```
* {
  color:green;
}
</style>
```

2. Class Selector

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

For example :

```
<style type="text/css">
h1.RedText
{
  font-family:Monotype Corsiva;
  color:red;
  font-size: 14pt;
}
h1.BlueText
{
  font-family:Arial;
  color:blue;
  font-size: 10pt;
}
</style>
```

Using the **dot Operator** particular class is defined

Q.22 List the ways of positioning an element within a browser window.

AU : Dec.-12

Ans. : There are three ways of positioning the element within the browser window. These are -

1. Absolute positioning
2. Relative positioning
3. Float positioning

Q.23 Give example for inline style sheet.

AU : May-13

Ans. : Using the style tag we can embed the css style within the web document. Such type of style sheet is called inline style sheet. For example -

```
<html>
<head>
  <style type="text/css">
    h1
    {
      color:green
    }
  </style>
</head>
<body>
  <h1> This page contains an inline style sheet</h1>
</body>
```

</html>

Q.24 Define cascading.

Ans. : Cascading is the collection of rules that the browsers use to determine how to use the style information.

Q.25 What are different variations of CSS ?

Ans. : 1) CSS 1 2) CSS 2 3) CSS 2.1 4) CSS 3 5) CSS 4

Q.26 What are the features of CSS3 ?

Ans. : Various features of CSS3 are -

1. It allows simple arithmetic calculations using calc() function.
2. There are many powerful selectors available using which the style sheet can be fancy.
3. There is introduction of new pseudo classes such as only-child, empty, root, first-child, after, before.
4. It supports 2D/3D transformations.
5. It has a support for multiple backgrounds. The background-size property is also introduced in CSS3. That means designer can specify the size of background image using either length or percentage.

Q.27 Write appropriate inline CSS to show a section of the HTML document with a font size 20.

AU : May-16

Ans. : The inline CSS can be written with the tag **style**.

```
<!DOCTYPE html>
<html>
<head>
  <title>Display Section with Specific Font Size</title>
</head>
<body>
  <div style="font-size:20px;">
    India is my country.All Indians are my brothers and sisters.I love my country, and I am proud
    of its rich and varied heritage.I shall always strive to be worthy of it.I shall respect my parents,
    teachers and all elders and treat everyone with courtesy.To my country and my people, I pledge
    my devotion.In their well being and prosperity alone lies my happiness.
  </div>
</body>
```

Q.28 List out the limitations of CSS

AU : May-18

Ans. :

- 1) CSS work differently on different browser.
- 2) One can not read files using CSS.
- 3) It is unable to interact with CSS.
- 4) The CSS can not invoke a web page.



STUCOR APP

Unit II

<h1 style="font-size: 48px; margin: 0;">4</h1>	<h2 style="font-size: 24px; margin: 0;">Client Side Scripting</h2>
--	--

Syllabus

The JavaScript Language-History and Versions Introduction JavaScript in Perspective - Syntax - Variables and Data Types - Statements - Operators - Literals-Functions-Objects-Arrays-Built-in Objects-JavaScript Debuggers.

Contents

4.1	Introduction to JavaScript Language	May-14, Marks 8
4.2	Writing First JavaScript		
4.3	Identifier, Keywords and Comments		
4.4	Data Types		
4.5	Variable		
4.6	Operators		
4.7	Literals		
4.8	Input and Output		
4.9	Control Structures	Dec.-08, May- 10, 11, Marks 8
4.10	Functions.....	Dec.-07, 08, May-11,12,13, Marks 16
4.11	Arrays.....	May- 08,12, Dec.-13, Marks 8
4.12	Standard Objects	Dec.-09, 11, 13, Marks 16
4.13	Form Processing in JavaScript.....	Dec.-09, May-08, 14, Marks 8
4.14	JavaScript Debuggers.....	May-12, Marks 8
4.15	Examples.....	May-08, 09, Dec.-08, 09, Marks 16
	<i>Two Marks Questions with Answers</i>		

4.1 Introduction to JavaScript Language

AU : May-14, Marks 8

4.1.1 History and Versions

The JavaScript is a programming language which is implemented by Netscape Communications in 1995.

Various client side versions of JavaScript are enlisted in the following table

Browser	Year	JavaScript Version
Netscape Navigator 2.0	1995	JavaScript 1.0
Microsoft Internet Explorer 3.0	1996	JavaScript 1.0 (JScript 1.0)
Netscape Navigator 3.0	1996	JavaScript 1.1
Netscape Navigator 4.0	1997	JavaScript 1.2
Microsoft Internet Explorer 4.0	1997	JavaScript 1.2 (JScript 3.0)
Netscape Navigator 4.5	1998	JavaScript 1.3
Microsoft Internet Explorer 5.0	1999	JavaScript 1.3 (JScript 5.0)

4.1.2 Features of JavaScript

Following are some features of JavaScript

- 1. Browser Support :** For running the JavaScript in the browser there is no need to use some plug-in. Almost all the popular browsers support JavaScripting.
- 2. Structure Programming Syntax:** The Javascript supports much commonly the structured language type syntax. Similar to C-style the programming blocks can be written.
3. It automatically inserts the semicolon at the end of the statement, hence there is no need to write semicolon at the end of the statement in JavaScript.
- 4. Dynamic Typing :** It supports dynamic typing, that means the data type is bound to the value and not to the variable. For example one can assign integer value to a variable say 'a' and later on can assign some strg the ing value to the same variable in JavaScript.
- 5. Run Time Evaluation :** Using the **eval** function the expression can be evaluated at run time.
- 6. Support for Object :** JavaScript is object oriented scripting language. However handling of objects in JavaScript is somewhat different that the conventional object oriented programming languages. JavaScript has a small number of in-built objects.
- 7. Regular Expression :** JavaScript supports use of regular expressions using which the text-pattern matching can be done. This feature can be used to validate the data on the web page before submitting it to the server.

8. Function Programming : In JavaScript functions are used. One function can accept another function as a parameter. Even, one function can be assigned to a variable just like some data type. The function can be run without giving the name.

4.1.3 Difference between JavaScript and Java

Sr. No.	Java	JavaScript
1.	Java is a programming language.	JavaScript is a scripting language.
2.	Java is an object oriented programming language.	Javascript is not a object oriented programming language. Its object model is far different than a typical object oriented programming language such as C++ or Java.
3.	Java is strongly typed language and the type checking is done at compile time.	In JavaScript variables need not be declared before their use. Checking the compatibility of type can be done dynamically.
4.	Objects in Java are static. That means the number of data members and method are fixed at compile time.	In JavaScript the objects are dynamic. That means we can change the total data members and method of an object during execution.

University Question

1. Elaborate the language history of JavaScript and its versions.

AU : May-14, Marks 8

4.2 Writing First JavaScript

The JavaScript can be directly embedded within the HTML document or it can be stored as external file.

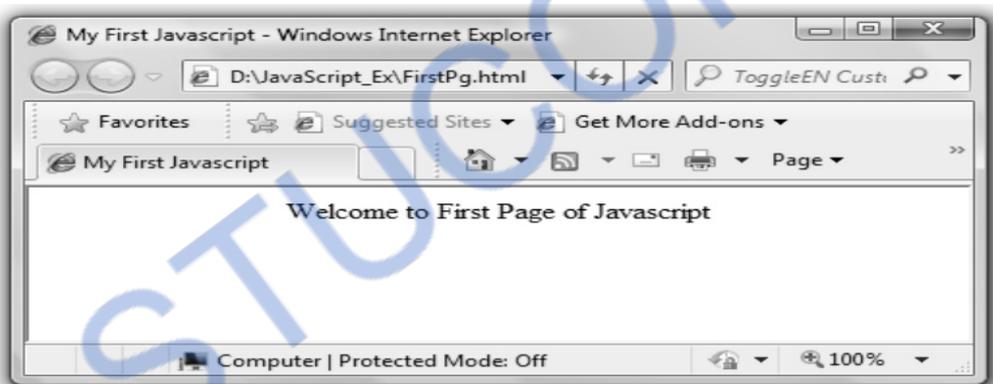
Directly embedded JavaScript

The syntax of directly embedding the JavaScript in the HTML is

```
<script type="text/javascript">
...
...
...
</script>
```

Ex. 4.2.1 : Write a JavaScript to display Welcome message in JavaScript**Sol. :**

```
<!DOCTYPE html >
<html >
  <head>
    <title> My First Javascript </title>
  </head>
  <body>
    <center>
      <script type="text/javascript">
        /*This is the First JavaScript*/
        document.write(" Welcome to First Page of Javascript");
      </script>
    </center>
  </body>
</html>
```

Output**Script Explanation :**

In above script

(1) we have embedded the javascript within

```
<script type="text/javascript">
```

```
...
```

```
</script>
```

(2) a comment statement using /* and */. Note that this type of comment will be recognized only within the <script> tag. Because, JavaScript supports this kind of comment statement and not the XHTML document.

(3) Then we have written document.write statement, using which we can display the desired message on the web browser.

Indirectly Embedding JavaScript

If we want to embed the JavaScript indirectly, that means if the script is written in some another file and we want to embed that script in the HTML document then we must write the script tag as follows -

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

JavaScript is which is to be embedded is in the file **MyPage.js**

We will follow the following steps to use the external JavaScript file.

Step 1 : Create an XHTML document as follows -

XHTML Document[FirstPg.html]

```
<!DOCTYPE html >
<html>
  <head>
    <title> My First Javascript </title>
  </head>
  <body>
    <center>
      <script type="text/javascript" src="my_script.js">
      </script>
    </center>
  </body>
</html>
```

This is an external javascript file, it can be specified with the attribute **src**

Step 2 :

JavaScript[my_script.js]

```
/*This is the First JavaScript*/
document.write(" Welcome to First Page of Javascript");
```

Step 3 : Open the HTML document in Internet Explorer and same above mentioned output can be obtained.

Advantages of indirectly embedding of JavaScript

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

Disadvantages of indirectly embedding of JavaScript

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

4.3 Identifier, Keywords and Comments

1. Identifiers

Identifiers are the names given to the variables. These variables hold the data value. Following are some conventions used in JavaScript for handling the identifiers -

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

2. Reserved words

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

break	Continue	delete	for	in	return	throw	var	with
case	default	else	function	instanceof	switch	try	void	
catch	do	finally	if	new	this	typeof	while	

3. Comments

JavaScript supports following comments

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

4. Semicolon

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

4.4 Data Types

JavaScript defines two entities primitives and objects. The primitives are for storing the values whereas the object is for storing the reference to the actual value.

There are following primitive types used in JavaScript

1. Number
2. String
3. Boolean
4. Undefined
5. Null

There are three type of predefined objects in JavaScript

1. Number
2. String
3. Boolean

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



Fig. 4.4.1 (a) Representation of primitive type



Fig. 4.4.1 (b) Representation of object

4.5 Variable

In JavaScript we can declare the variable using the reserved word var. The value of this variable can be any thing; it can be numeric or it can be string or it can be a Boolean value.

JavaScript[VarDemo.html]

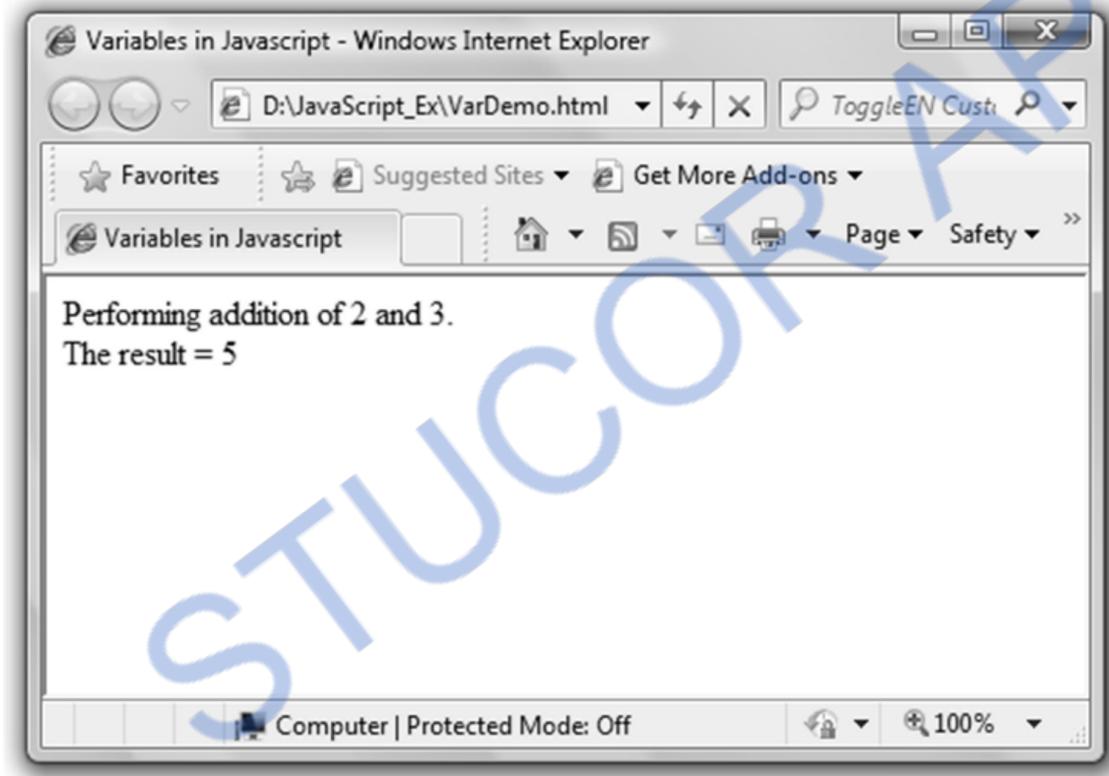
```

<!DOCTYPE html >
<html >
<head>
<title> Variables in Javascript </title>
</head>
<body>
<script type="text/javascript">
var a,b,c;
var string;
a=2;
b=3;
c=a+b;
    
```

Variable declaration is done using **var**. Note that there is no data type required for handling variables.

```
string = "The result = ";
document.write("Performing addition of 2 and 3. "+"<br/>");
document.write(string);
document.write(c);
</script>
</body>
</html>
```

Output



Note that using **var** we can define the variable which is of type numbers(2 , 3 or 5) as well as the string “The result”.

4.6 Operators

Various operators used by JavaScript are as shown in following table -

Type	Operator	Meaning	Example
Arithmetic	+	Addition or unary plus	c = a+b
	-	Subtraction or unary minus	d = -a

	*	Multiplication	c=a*b
	/	Division	c=a/b
	%	Mod	c=a%b
Relational	<	Less than	a<4
	>	Greater than	b>10
	<=	Less than equal to	b<=10
	>=	Greater than equal to	a>=5
	==	Equal to	x==100
	!=	Not equal to	m!=8
Logical	&&	And operator	0&&1
		Or operator	0 1
Assignment	=	Is assigned to	a=5
Increment	++	Increment by one	++i or i++
Decrement	--	Decrement by one	-- k or k--

The conditional operator is ? The syntax of conditional operator is
Condition?expression1:expression 2

Where expression1 denotes the true condition and expression2 denotes false condition.

For example :

a>b?true:false

This means that if a is greater than b then the expression will return the value true otherwise it will return false.

4.6.1 String Concatenation Operator

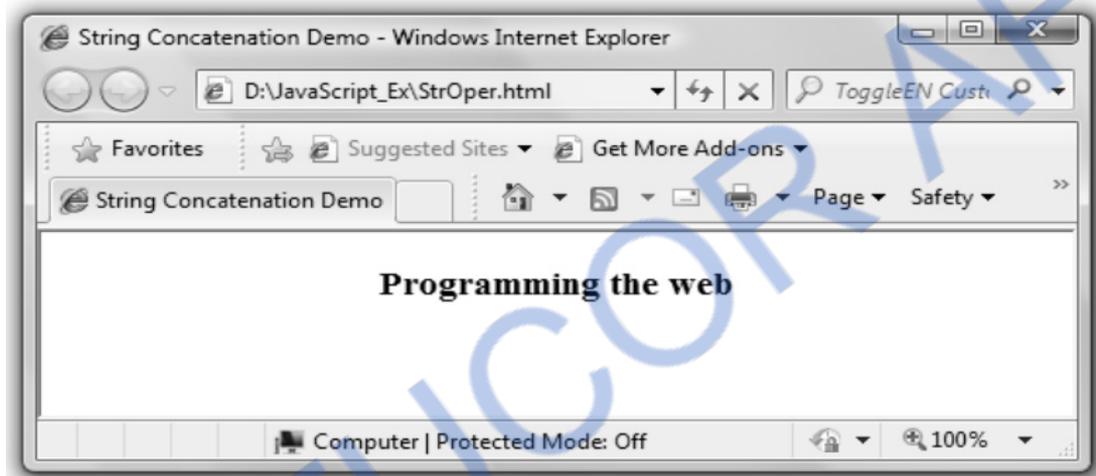
Two string can be concatenated using the + operator. A variable can be concatenated with the string using + operator also.

JavaScript[StrOper.html]

```
<!DOCTYPE html >
<html >
<head>
  <title>String Concatenation Demo</title>
</head>
<body>
```

```
<center>
<script type="text/javascript">
  var first_string;
  first_string="Programming";
  document.write("<h3>" +first_string+" the web"+"</h3>");
</script>
</center>
</body>
</html>
```

Output



4.7 Literals

There are two types of literals used in JavaScript and those are numeric literals and string literals. The numeric literals are called **numbers**. These numbers can include integer values, floating point or double precision values. For example -

10
10.3
10.0
10.
10E3
10.2E4
10.e2
10e-3

are all valid numeric literals.

The string literals are the sequence of characters. It can be written in double quotes “ ” or in single quotes ‘ ’. For example

```
‘Rain Rain come soon ‘
```

4.8 Input and Output

4.8.1 The document.write

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

- **For example**

```
document.write("Great India");
```

- We can pass the variable instead of a string as a parameter to the document.write.

For example

```
var my_msg="Great India";  
document.write(my_msg);
```

Note that the variable my_msg should not be passed in double quotes to document.write otherwise the result the string “my_msg” will be printed on the browser instead of the string “Great India” .

- We can combine some HTML tags with the variable names in document.write so that the formatted display can be possible on the web browser.

For example – if we want to print the message “Great India” on the web browser in bold font then we can write following statements-

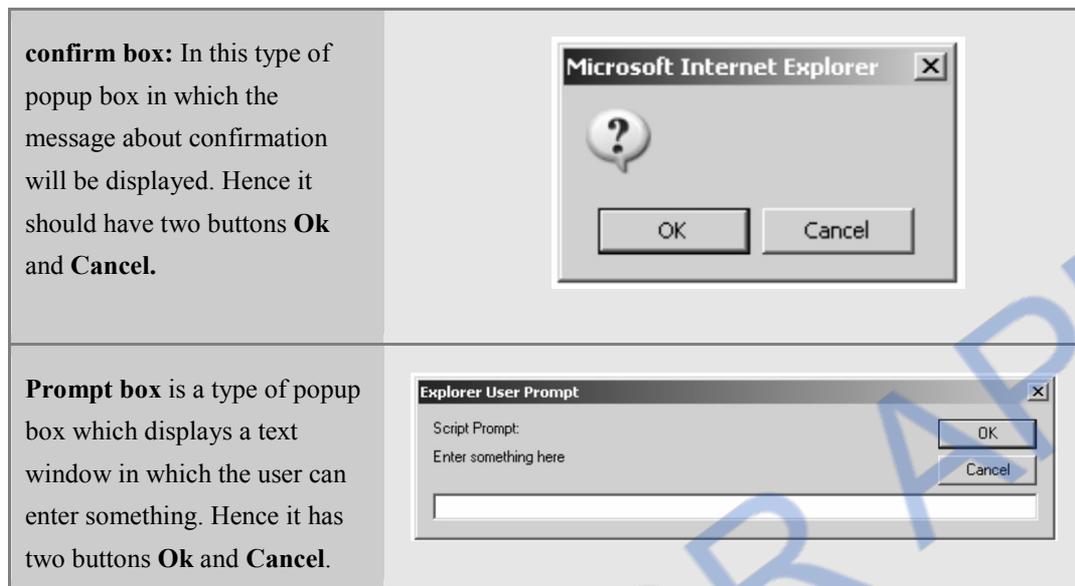
```
var my_msg="Great India";  
document.write("<strong>" + my_msg + "</strong>");
```

4.8.2 Popup Box

- One of the important features of JavaScript is its **interactivity** with the user.
- There are **three types of popup boxes** used in JavaScript by which user can interact with the browser.

alert box : In this type of popup box some message will be displayed.





JavaScript[PopupBox.html]

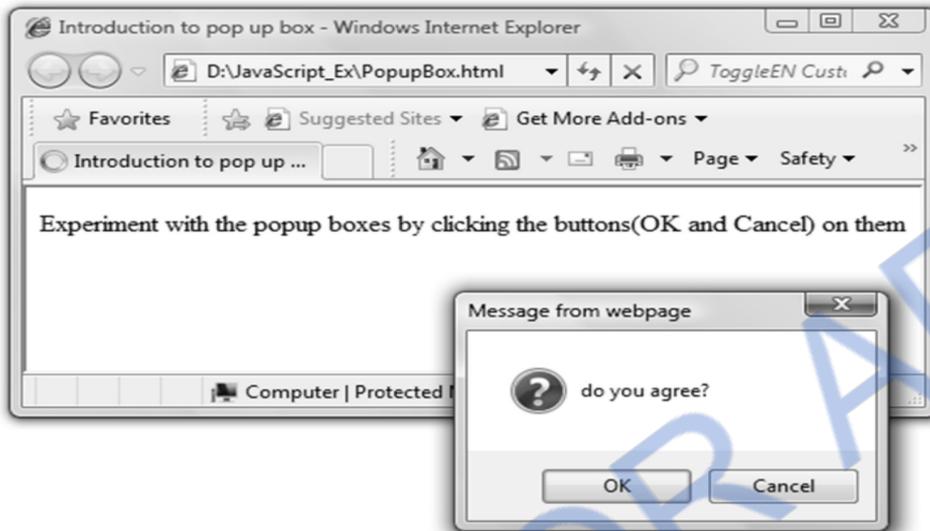
```

<!DOCTYPE html>
<html >
<head>
  <title>Introduction to pop up box</title>
</head>
<body>
<p>Experiment with the popup boxes by clicking the buttons(OK and Cancel) on them</p>

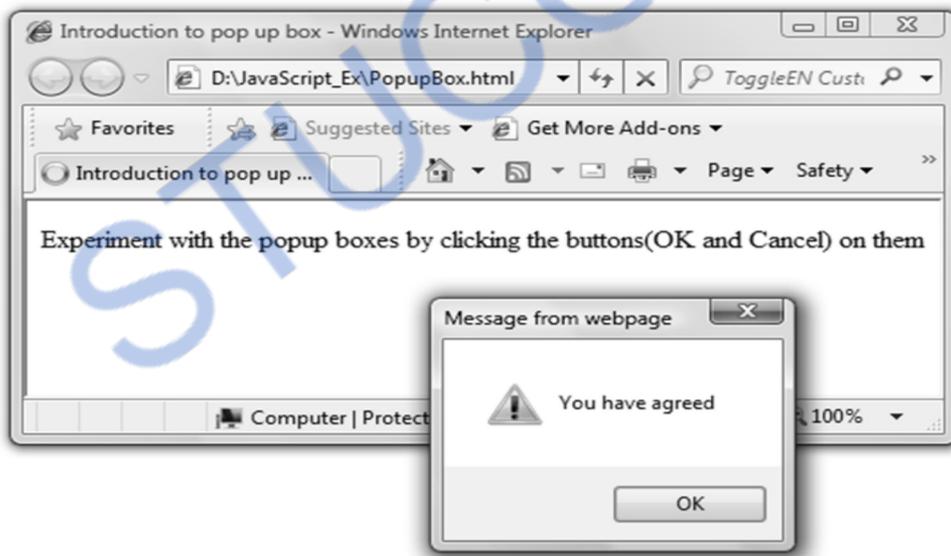
<script type="text/javascript">
if(confirm("do you agree?"))
  alert("You have agreed");
else
  input_text=prompt("Enter some string here.."," ");
/*the value entered in prompt box is returned
and stored in the variable text */
  alert("Hi "+input_text);
</script>
</body>
</html>

```

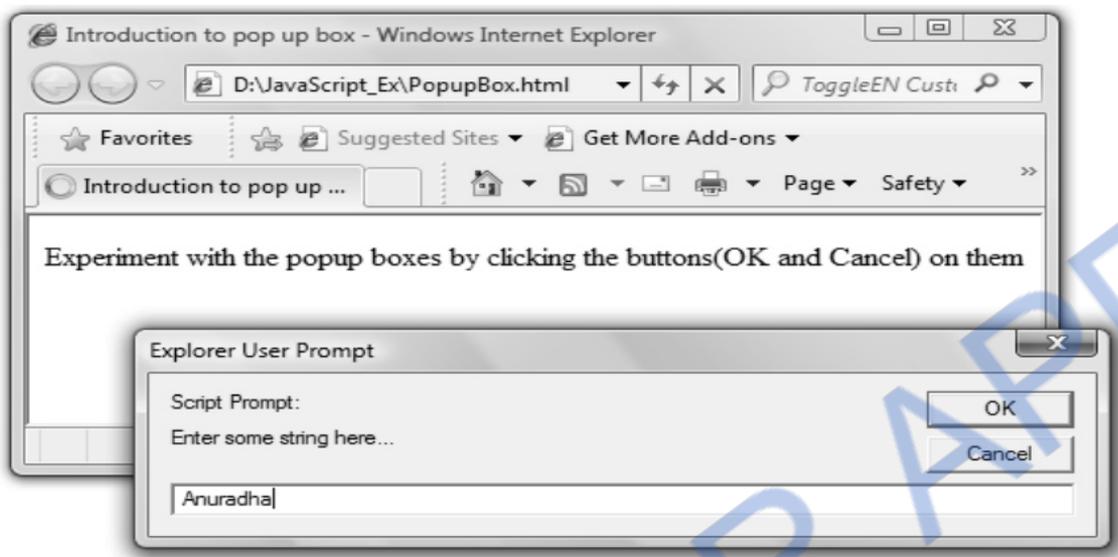
Output



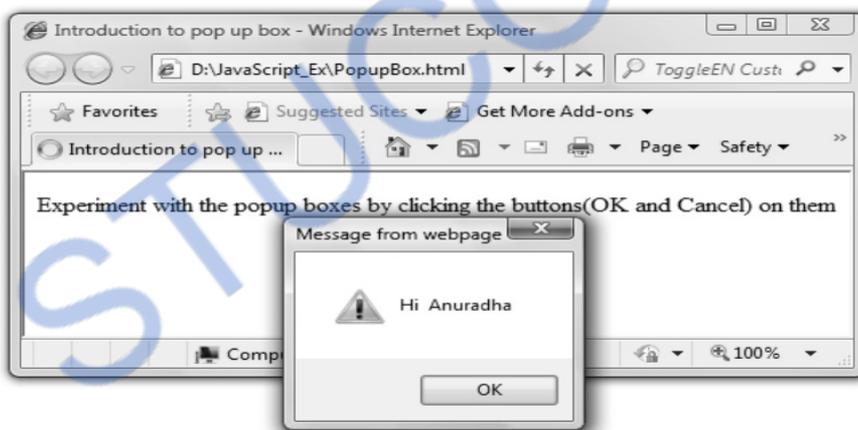
If we click on OK button then an alert box will appear.



On load, the confirm box appears and if we click on Cancel button then the prompt box will appear. We can type some string within it.



Click OK button and we will get the alert box as follows -



Thus alert, confirm and prompt boxes cause the browser to wait for user response. The user responds by clicking the button on these popup boxes.

Script Explanation

On loading this script using web browser first of all a confirm box will be displayed. If we click on OK button an alert box will appear. Otherwise a prompt box will be displayed. Again if we click on the OK button of prompt box again an alert box will appear which will display the string which you have entered on prompt box. If you run the above script you will get all these effects.

4.9 Control Structures**AU : Dec.-08, May- 10, 11, Marks 8**

Various control structures used in JavaScript are

Statement	syntax	Example
if-else	<pre>if(condition) statement else statement</pre>	<pre>if(a>b) document.write(" a is greater than b"); else document.write("b is greater than a");</pre>
while	<pre>while(condition) { statements }</pre>	<pre>while(i<5) { i=i+1; document.write("value of i"+i) }</pre>
do..while	<pre>do { }while(condition);</pre>	<pre>do { i=i+1; document.write("value of i"+i) }while(i<5);</pre>
for	<pre>for(initialization;test condition;stepcount) { statements }</pre>	<pre>for(i=0;i<5;i++) { document.write(i); }</pre>
switch...case	<pre>switch(expression) { case 1: statements break; case 2: statements break; ... default: statements }</pre>	<pre>switch(choice) { case 1: c=a+b; break; case 2:c=a-b; break; }</pre>
break : Similar to C or C++, the break statement is used to break the loop.	break;	<pre>for(i= 10;i>=0;i--) { if(i==5) break; }</pre>
continue: The continue statement is used in a loop in order to continue(skip). The keyword continue is used to make use of continue statement in a loop.	continue;	<pre>for(i= 10;i>=0;i--) { if(i==5) { x=i; continue; } }</pre>

Control structure is essential part of JavaScript. The Control Structures in JavaScript are just similar to that of C or C++. Various control structures are -

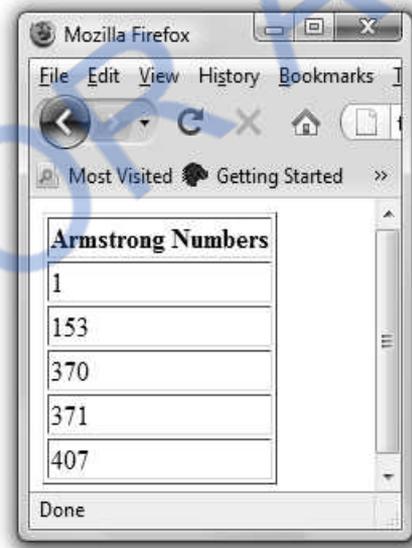
1. if statement
2. While
3. Do-while
4. for
5. switch case
6. break
7. Continue

Let us discuss various examples that make use of control structures.

Ex. 4.9.1 : Develop a javascript to generate 'ARMSTRONG NUMBERS' between the range 1 to 100. [Eg : 153 is an Armstrong number, since sum of the cube of the digits is equal to the number i.e. $[13+53+33=153]$]

Sol. :

```
<html>
<body>
<table border="1" align="center">
<th>Armstrong Numbers</th>
<script type="text/javascript">
var num,i,temp,sum;
var n=0;
i=1;
do
{
  num=i;
  sum=0;
  while(num>0)
  {
    n=num%10;
    n=parseInt(n);
    num=num/10;
    num=parseInt(num);
    sum=sum+(n*n*n);
  }
  if(sum==i)
  {
    document.write("<tr><td>" + i + "</td></tr>");
  }
  i++;
}while(i<=1000);
</script>
</table>
</body>
</html>
```



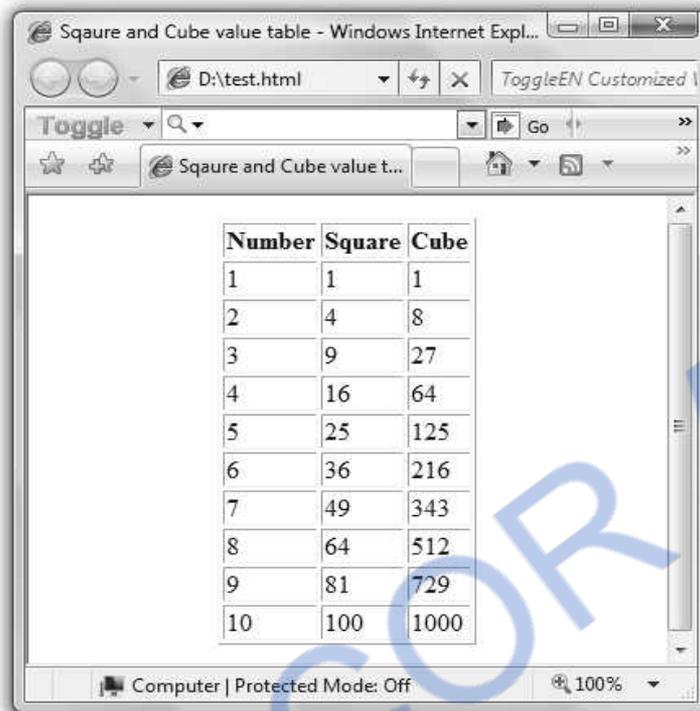
Ex. 4.9.2 : Write a javascript that displays the as per following:

(Calculate the squares and cubes of the numbers from 0 to 10)

Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

Sol. :

```
<html>
<head>
<title>Sqaure and Cube value table</title>
</head>
<body>
<table border=1 align="center">
<th>Number</th><th>Square</th><th>Cube</th>
<script type="text/javascript">
for (i=1; i<=10; i++)
{
document.write("<tr><td>" + i + "</td> <td>" + (i*i) + "</td> <td>" + (i*i*i) + "</td></tr>");
}
</script>
</table>
</body>
</html>
```



The screenshot shows a browser window titled "Sqaure and Cube value table - Windows Internet Expl...". The address bar shows "D:\test.html". The page content is a table with three columns: "Number", "Square", and "Cube". The table contains the following data:

Number	Sqaure	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

The browser status bar at the bottom shows "Computer | Protected Mode: Off" and a zoom level of "100%".

Ex. 4.9.3 : Write a script that reads an integer and displays whether it is a prime number or not.

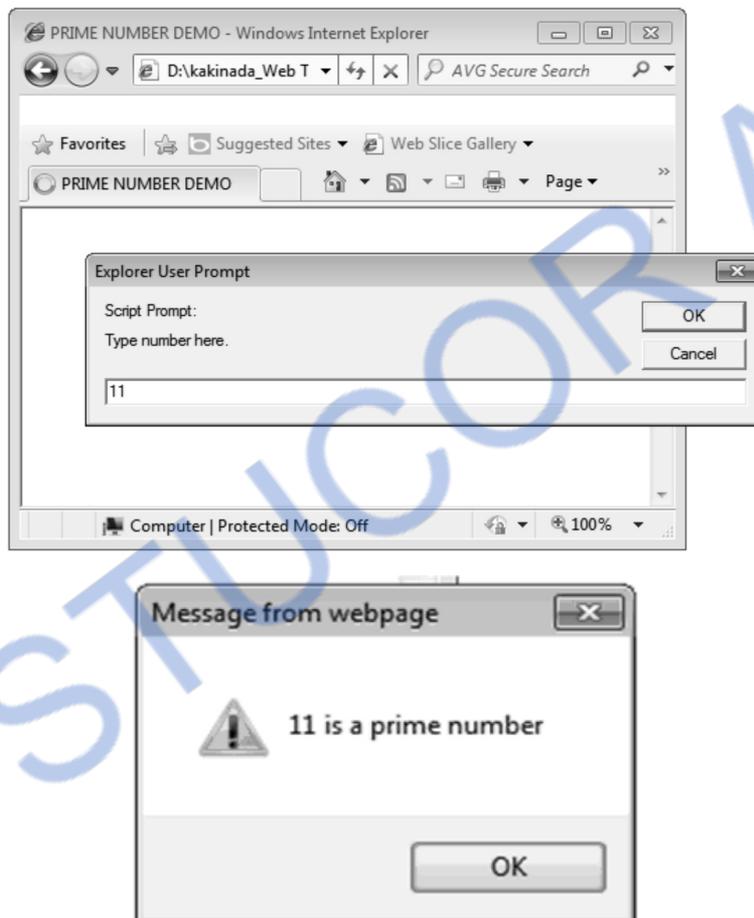
Sol. :

```
<html>
<head>
  <title>PRIME NUMBER DEMO</title>
</head>
<body>
  <script type="text/javascript">
    var num=prompt("Type number here.", "");
    var b;
    var flag=1;
    for(i=2;i<num;i++)
    {
      b=num%i;
      if(b==0)
      {
        flag=0;

        break;
      }
    }
  </script>
</body>
</html>
```

```
if(flag==0)
    alert(num+" is not a prime number");
else
    alert(num+" is a prime number");
</script>
</body>
</html>
```

Output



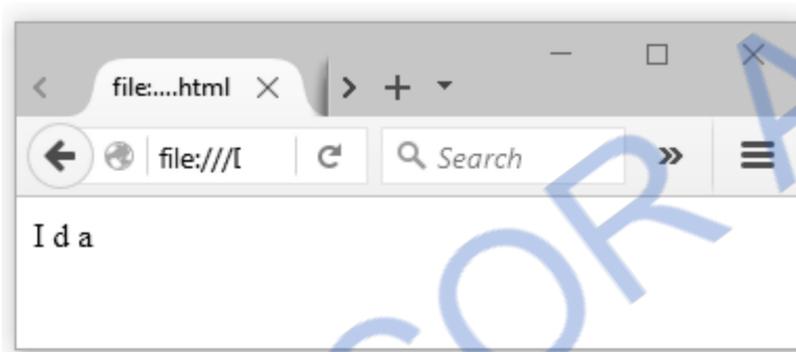
Ex. 4.9.4 : Write a JAVAScript to print characters of a string at odd positions.

(for example for the string India, I, d and a should get printed).

Sol. :

```
<html>
<body>
<script>
var str="India";
var k=str.length;
```

```
for(i=0;i<=k;i=i+2)
{
n=str.charAt(i);
document.write(n);
document.write(" ");
}
</script>
</body>
</html>
```

Output

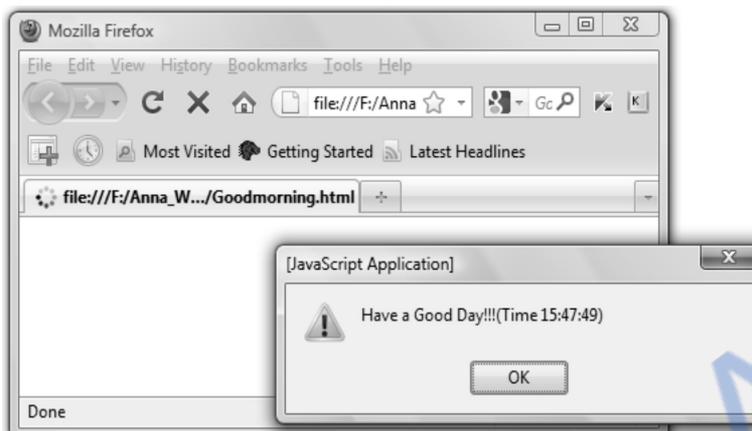
Ex. 4.9.5 : Develop a JavaScript page to demonstrate an If condition by which the time on your browser is less than 10; you will get a “Good morning” greeting.

AU : Dec.-08, Marks 2

Sol. : Goodmorning.html

```
<html>
<head>
<script type="text/javascript">
function GreetMsg()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
if(h<10)
alert("Good Morning!!!(Time "+h+": "+m+": "+s+"");
else
alert("Have a Good Day!!!(Time "+h+": "+m+": "+s+"");
}
</script>
</head>
<body onload="GreetMsg()">
</body>
</html>
```

Output



Ex. 4.9.6 : Write a JavaScript to find and print the largest and smallest values among 10 elements of an array.

AU : May- 10, Marks 8

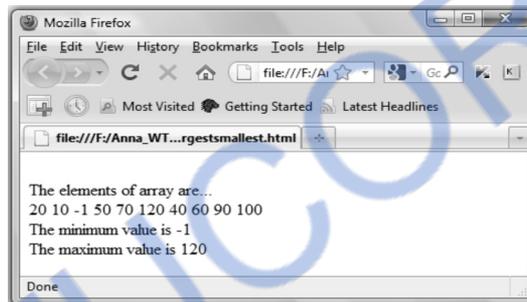
Sol. : largestsmallest.html

```
<html>
<head>
<script type="text/javascript">
function fun()
{
a=new Array(10);
a[0]=20;
a[1]=10;
a[2]=-1;
a[3]=50;
a[4]=70;
a[5]=120;
a[6]=40;
a[7]=60;
a[8]=90;
a[9]=100;
document.write("<br/>The elements of array are...<br/>");
max_val=a[0];
for(i=0;i<10;i++)
document.write(" " +a[i]);
min_val=a[0];
max_val=a[0];
for(i=0;i<10;i++)

{
```

```
    if(a[i]<min_val)
        min_val=a[i];
    if(a[i]>max_val)
        max_val=a[i];
}
document.write("<br/>The minimum value is "+min_val);
document.write("<br/>The maximum value is "+max_val);
}
</script>
<title>Finding largest and smallest Element</title>
</head>
<body onload=fun()>
</body>
</html>
```

Output



Ex. 4.9.7 : Write a JavaScript to find the product of first 15 even numbers.

Sol. :

```
<html>
<body>
<script type="text/javascript">
prod=1;
for(i=1;i<15;i++)
{
    if(i%2)
    {
        prod=prod*i;
    }
}
document.write("Product of odd numbers between 1 to 15 is "+prod);
</script>
</body>
</html>
```

University Question

1. State and explain the types of statements in JavaScript.

AU : May-11, Marks 8

4.10 Functions

AU : Dec.-07, 08, May-11,12,13, Marks 8

- We can write the functions in the JavaScript for bringing the modularity in the script.
- Separate functions can be created for each separate task. This ultimately helps in finding the bug from the program efficiently.
- We can define the function anywhere in the script either in head or body section or in both. But it is a standard practice to define the function in the head section and call that function from the body section.
- The keyword **function** is used while defining the function.
- The syntax for defining the function is

```
function name_of_function (arg1,arg2,...argn)
{
...
Statements
}
```

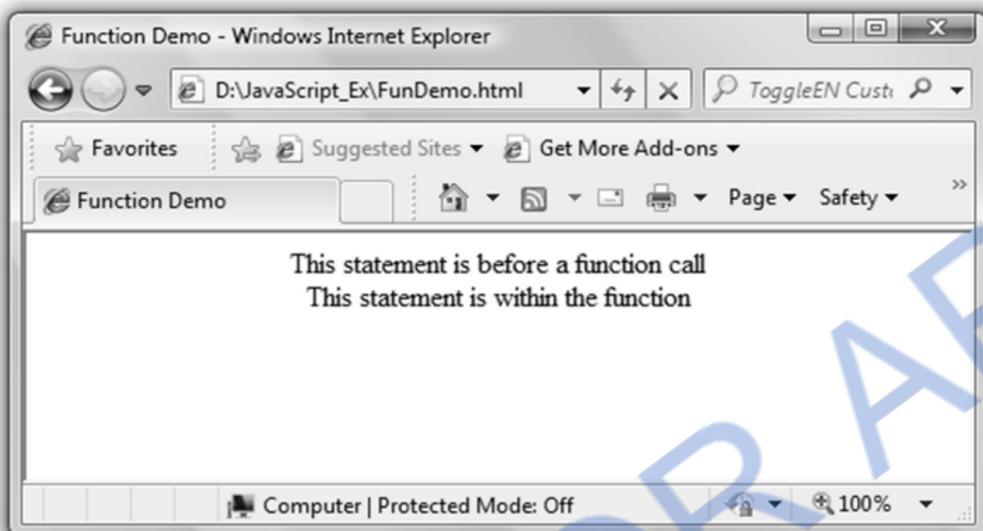
Here is a simple illustration in which we have written a function **my_fun()**

JavaScript[FunDemo.html]

```
<!DOCTYPE html>
<html>
<head>
<title>Function Demo</title>
<script type="text/javascript">
function my_fun()
{
document.write("This statement is within the function");
}
</script>
</head>
<body>
<center>
<script type="text/javascript">
document.write("This statement is before a function call");
document.write("<br>");
my_fun();
call to the function
</script>
</center>
</body>
</html>
```

Definition of function

Output

**Script Explanation**

The above code is pretty simple. We have defined one function named **my_fun** in the **head** section of the HTML document and called it from the **body** section. The corresponding **write** statements are used to display the messages on the browser window.

4.10.1 Returning Value from the Function

- We can return some value from the function using a keyword **return**.
- This return value is either stored in variable or directly displayed on the browser window.
- Following is a simple illustration in which the value is returned from the function and is directly displayed on the browser window using **document.write**.

JavaScript[FunRetDemo.html]

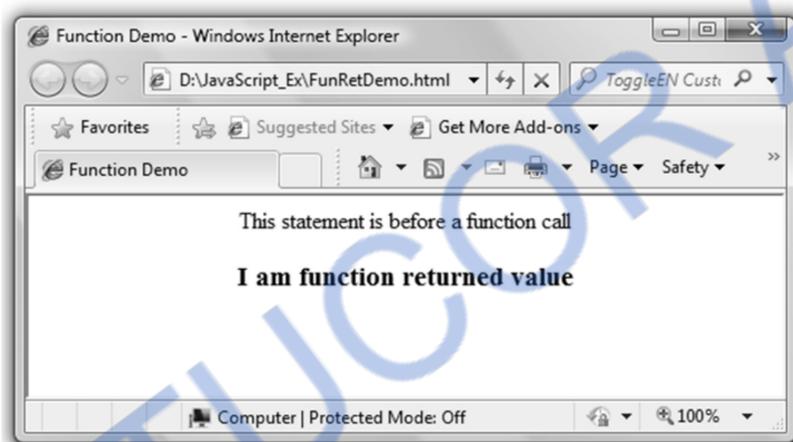
```
<!DOCTYPE html>
<html>
<head>
<title>Function Demo</title>
<script type="text/javascript">
function my_fun()
{
    str="I am function returned value";
    return str;
}
</script>
```

```

</head>
<body>
<center>
  <script type="text/javascript">
    document.write("This statement is before a function call");
    document.write("<br>");
    document.write("<h3>" + my_fun() + "<h3>");
  </script>
</center>
</body>
</html>

```

Output



4.10.2 Passing the Parameters to the Function

- Similarly we can pass some arguments to the function.
- The syntax is

```

function function_name(aregument1, argument2...,argumentn)
{
  //body of function
}

```

- In the following program, we have passed two arguments to the function and returning some value from the function.

JavaScript[FunPassRet.html]

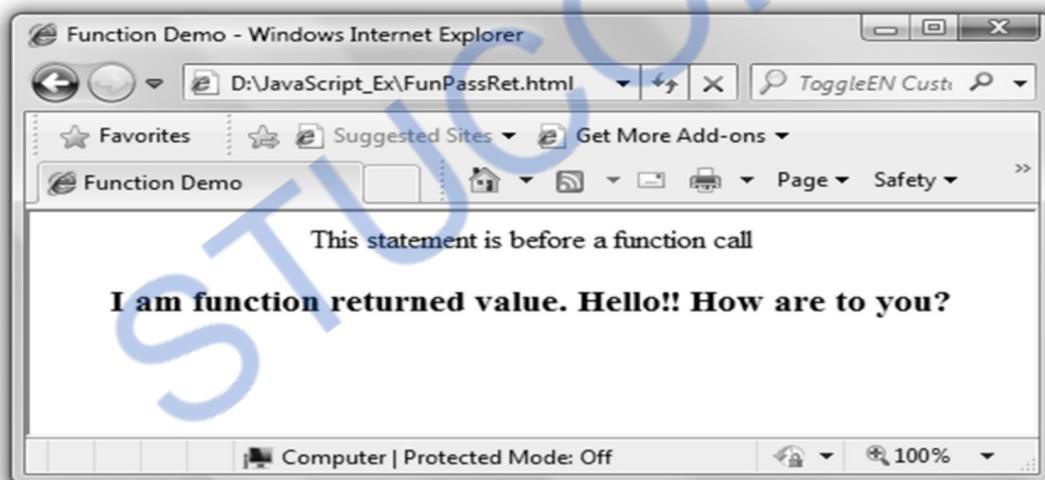
```

<!DOCTYPE html >
<html>
<head>
<title>Function Demo</title>
<script type="text/javascript">

```

```
function my_fun(str1,str2)//two arguments str1 and str2 are passed
{
    str="I am function returned value."+" "+str1+" "+str2 ;
    return str; //returning one string value from function
}
</script>
</head>
<body>
<center>
<script type="text/javascript">
    document.write("This statement is before a function call");
    document.write("<br>");
    document.write("<h3>"+my_fun("Hello!!","How are to you?")+ "<h3>");
</script>
</center>
</body>
</html>
```

Output



4.10.3 Passing an Array to the Function

- Similar to C or C++ we can pass an entire array as a parameter to the function.
- This method of array passing is called as **call by reference**.
- When an array is passed to the function the array name is simply passed.
- In the following JavaScript we have initialized an array in the body part and to display those contents of an array we have called a **display()** function. And to this function an array is passed as an argument. Let see the JavaScript.

JavaScript[FunPassArr.html]

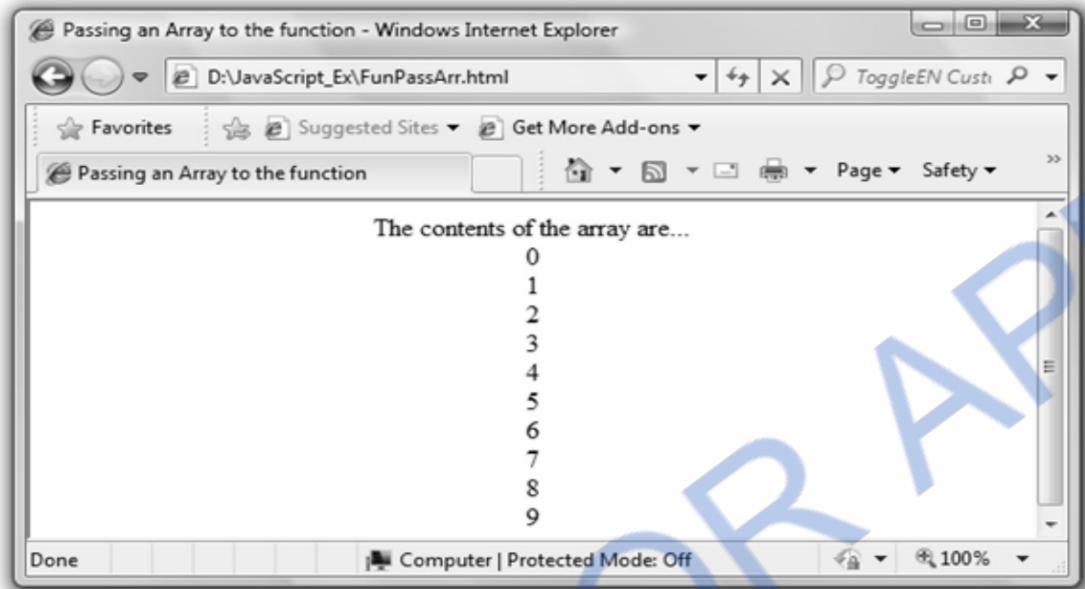
```
<!DOCTYPE html>
<html>
<head>
<title>Passing an Array to the function</title>
<script type="text/javascript">
function display(a)
{
document.write("The contents of the array are..." + "<br>");
i=0;
for(i in a)
{
document.write(a[i] + "<br>");
i++;
}
}
</script>
</head>
<body>
<center>
<script type="text/javascript">
var ar=new Array(10);
for(i=0;i<=9;i++)
{
ar[i]=i;
}
display(ar);
</script>
</center>
</body>
</html>
```

Function definition with array as an argument

Function call

In above given script we have called a **display** function by passing an array as a parameter to it. Hence we will get following output.

Output



Ex. 4.10.1 : Write a JavaScript to sort the elements of an array in ascending order

Sol. :

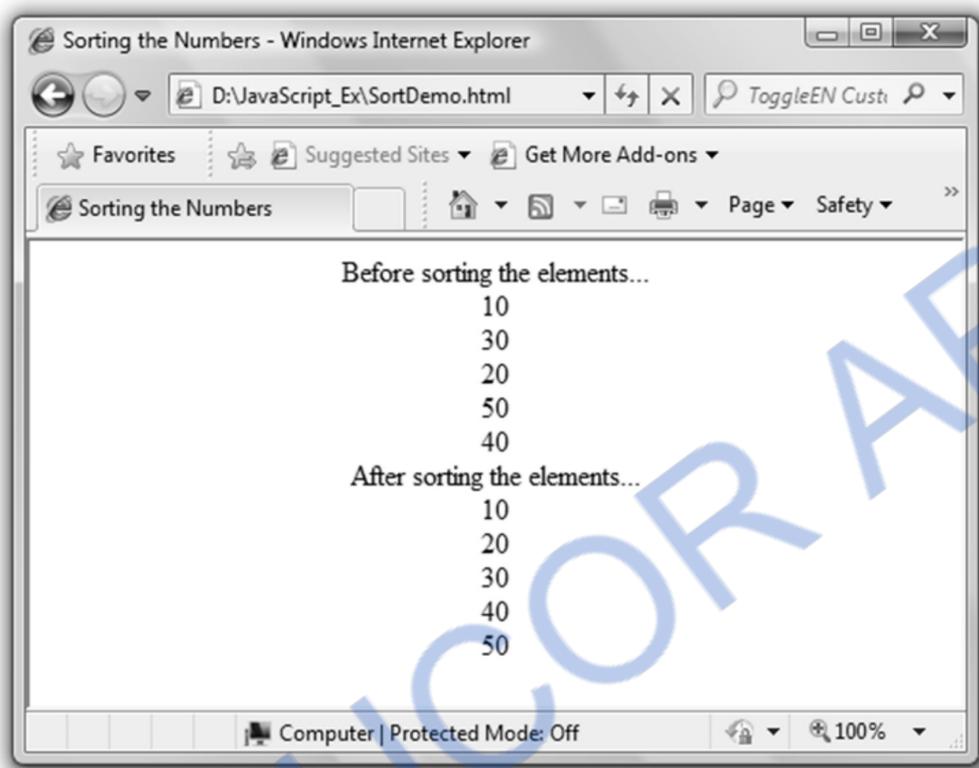
```
<!DOCTYPE html>
<html>
<head>
<title>Sorting the Numbers</title>
<script type="text/javascript">
function display(a)
{
i=0;
for(i in a)
{
document.write(a[i] + "<br>");
i++;
}
}
function sort(a)
{
for(i=0;i<a.length-1;i++)
{
for(j=i+1;j<a.length;j++)
```

```
{
  if(a[i]>a[j])
  {
    temp=a[i];
    a[i]=a[j];
    a[j]=temp;
  }
}
}
}
}
</script>
</head>
<body>
<center>
<script type="text/javascript">
  var ar=new Array(10,30,20,50,40);
  document.write("Before sorting the elements..."+"<br>");
  display(ar);
  sort(ar);
  document.write("After sorting the elements..."+"<br>");
  display(ar);
</script>
</center>
</body>
</html>
```

Script Explanation

1. In the above program we have stored some values in the array in the body part.
2. Then in order to display the contents of an array we have called **display** function.
3. Then we have called one more function called **sort** in which the elements of an array are sorted.
4. Note that in the sort function we have used *length* property which returns total number of elements in an array.
5. The simple bubble sort method is used to sort the array in an ascending order.
6. Again the display function is called in order to display the sorted array.

Output



Ex. 4.10.2 : Write an HTML and JavaScript program which accepts N as input and displays first N Fibonacci numbers as list.

Sol. :

```

<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function fun(str)
{
var num=Number(str);
var i,j,k,count;
document.write("<h3>"+ " The Fibonacci series is as follows..."+"</h3>");
i=0;
j=1;
document.write(j);
for(count=0;count<=num;count++)
{
k=i+j;
i=j;

```

```

j=k;
document.write(", "+k);
}
}
</script>
</head>
<body>
<script type="text/javascript">
var input_str=prompt("Enter some number","");
fun(input_str);
</script>
</body>
</html>

```

Ex. 4.10.3 : Write a javascript that uses a loop, that searches a word in sentence held in an array, returning the index of the word.

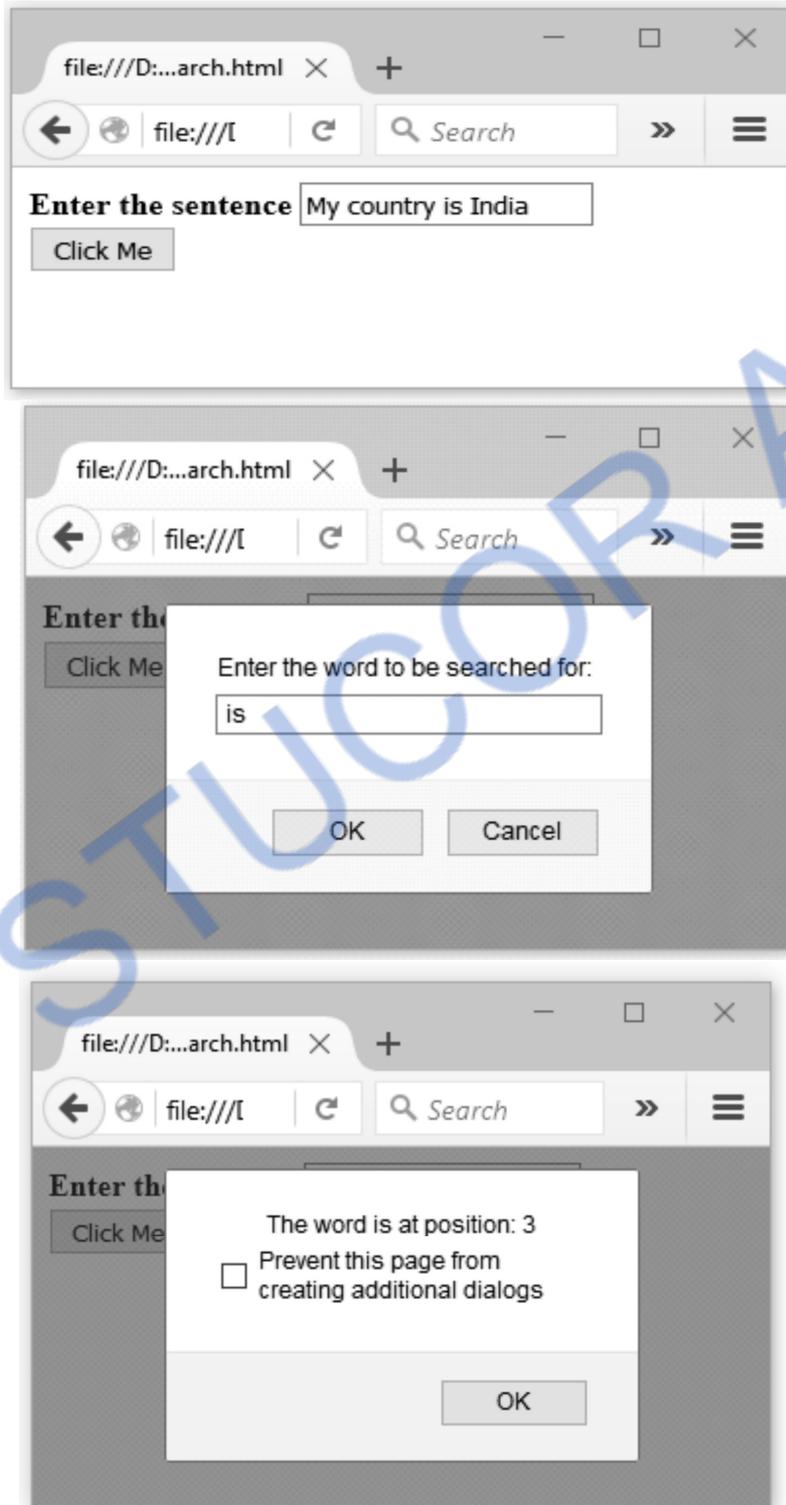
Sol. :

```

<!DOCTYPE html>
<html >
<head>
<script type="text/javascript">
function Search(form1)
{
var Sentence= form1.input.value;
var pattern=prompt("Enter the word to be searched for: ","");
alert(pattern);
temp=Sentence.split(" ");
for(i=0;i<temp.length;i++)
{
if(temp[i]==pattern)
{
alert("The word is at position: " +(i+1));
}
}
}
}
</script>
</head>
<body>
<form name="form1">
<b> Enter the sentence </b>
<input type="text" name="input"><br/>
<input type="button" value="Click Me" onclick="Search(form1)">
</form>
</body>
</html>

```

Output



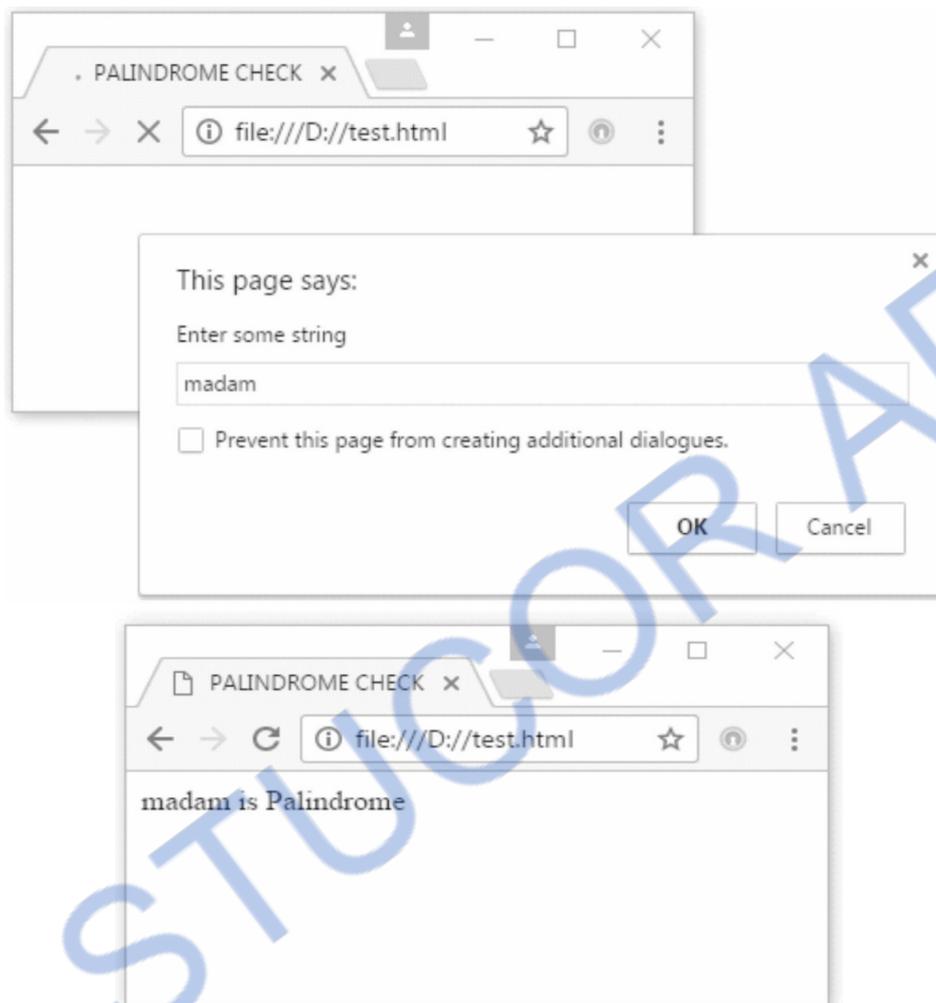
Ex. 4.10.4 : Write a JavaScript to check input string is palindrome or not.

Sol. :

```
<html>
<head>
<title>PALINDROME CHECK</title>
<script type="text/javascript">
function palindrome(str)
{
    var len = str.length;

    for ( var i = 0; i < Math.floor(len/2); i++ )
    {
        if (str[i] !== str[len - 1 - i])
        {
            return false;
        }
    }
    return true;
}
</script>
</head>
<body>
<script type="text/javascript">
str=prompt("Enter some string","");
if(palindrome(str))
    document.write(str+" is Palindrome");
else
    document.write(str+"Not Palindrome");
</script>
</body>
</html>
```

Output



Ex. 4.10.5 : Develop and demonstrate a HTML file that includes JavaScript that uses a function for the following problems :

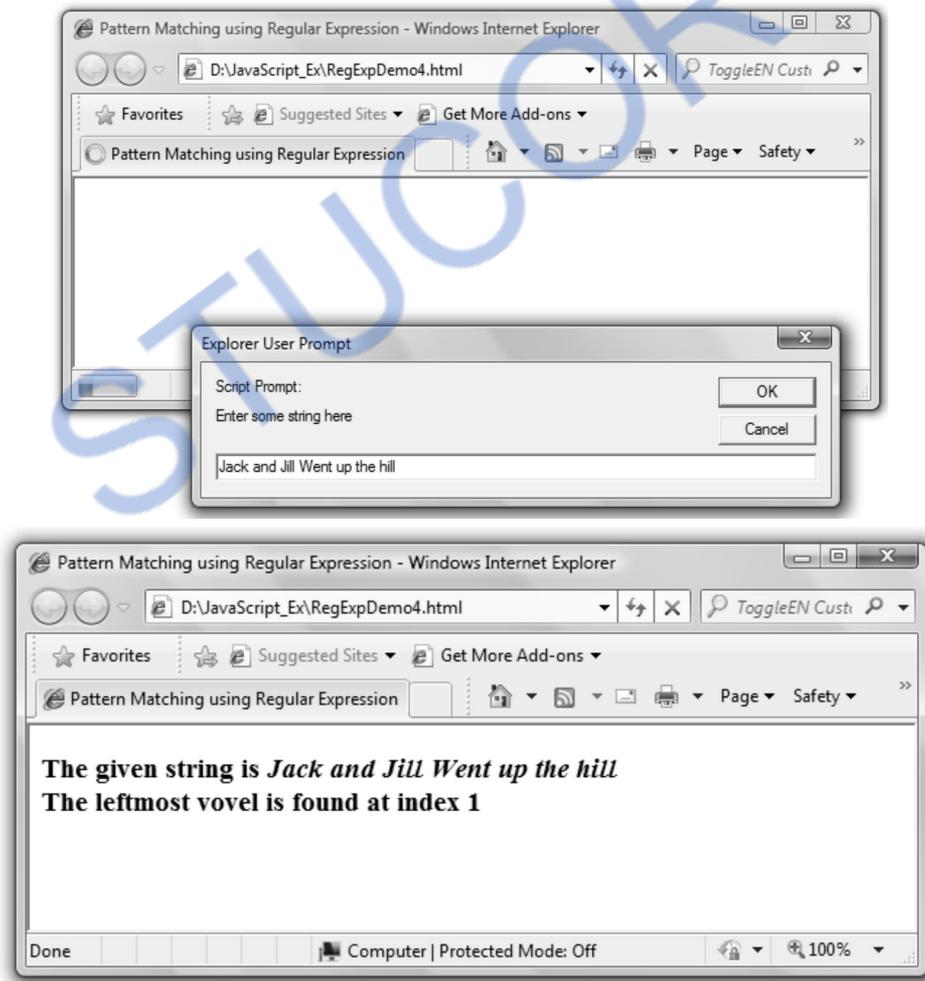
- (a) Parameter : A string Output : The position of the string of the left-most vowel
 (b) Parameter : A number Output : The number with digits in the reverse order.

Sol. : (a) [RegExpDemo4.html]

```
<!DOCTYPE html>
<html >
<head>
<title>Pattern Matching using Regular Expression </title>
<script type="text/javascript">
function TestString(str)
{
  document.write("The given string is ");
```

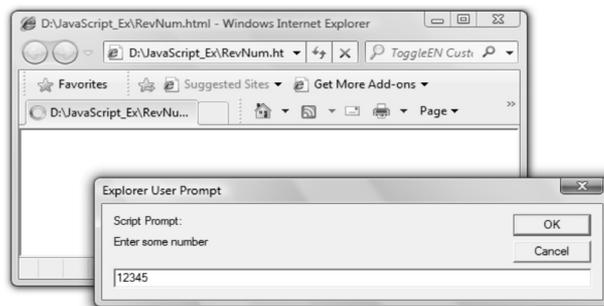
```
document.write("<em>" + str + "</em>" + "<br/>");
var i = str.match(/[aeiouAEIOU][a-zA-Z]*/);
return i;
}
</script>
</head>
<body>
  <h3>
    <script type="text/javascript">
      var input_str=prompt("Enter some string here","");
      p=TestString(input_str);
      document.write("The leftmost vowel is found at index "+p.index);
    </script>
  </h3>
</body>
</html>
```

Output

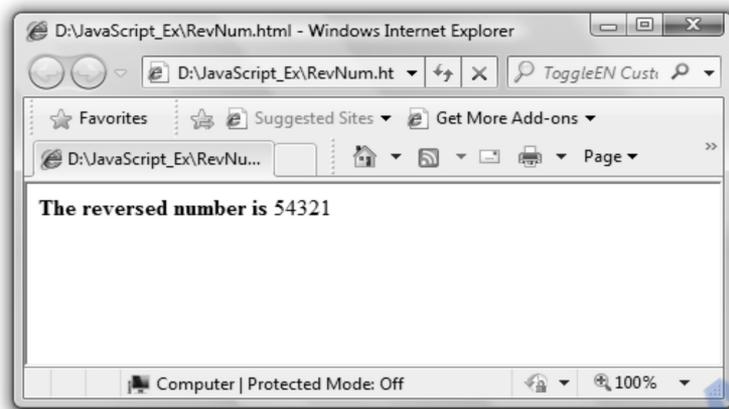


b) [RevNum.html]

```
<!DOCTYPE html>
<html >
<head>
<script type="text/javascript">
function fun(str)
{
var num=Number(str);
var n=0;
document.write("<b>"+"The reversed number is "+"</b>");
while(num>0)
{
n=num%10;
n=parseInt(n);
num=num/10;
num=parseInt(num);
document.write(n);
}
}
</script>
</head>
<body>
<script type="text/javascript">
var input_str=prompt("Enter some number","");
fun(input_str);
</script>
</body>
</html>
```

Output

Click OK button and you will get the reversed number as follows -

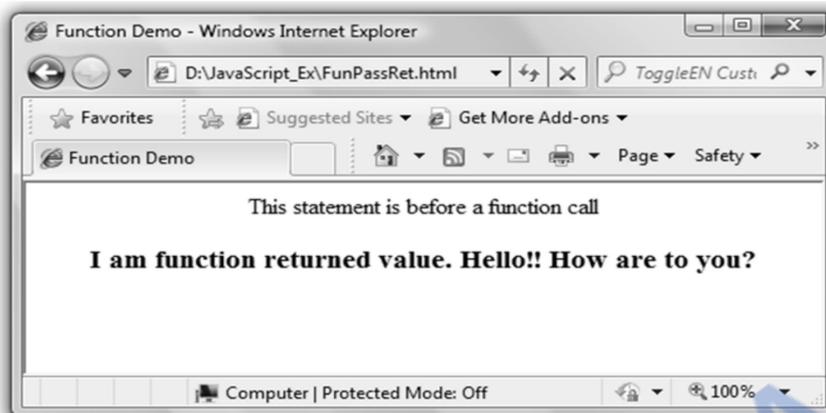


Ex. 4.10.6 : Write a JavaScript function to find the product of two arguments and return the result.

AU : Dec.-08, Marks 5

Sol. :

```
<!DOCTYPE html>
<html>
<head>
<title>Function Demo</title>
<script type="text/javascript">
function my_fun(num1,num2)
{
return num1*num2;
}
</script>
</head>
<body>
<center>
<script type="text/javascript">
document.write("The product of 10 and 20 is ");
document.write("<br>");
document.write("<h3>"+my_fun(10,20)+"</h3>");
</script>
</center>
</body>
</html>
```



Ex. 4.10.7 : Write a JavaScript function named `drawstrips` that will draw a number of colored strips by generating an HTML table in which each row has (possibly) different color. Your function must have 3 parameters :

- Width, the width of each strip
- Height, the height of each strip
- Colors an array in which each element is the name of color. Each color name is a valid HTML color (can be used as the value of `BGCOLOR` attribute or CSS color property)

AU : Dec.-07, CSE, Marks 16

Sol. : [stripeDemo.html]

```
<html>
<head>
<title></title>
<script>
var colors = new Array('#ff0000','#00ff00','#0000ff','#ff00ff');
function drawstrips(id)
{
    if(document.getElementById)
    {
        var table = document.getElementById(id);//get the table
        var rows = table.getElementsByTagName("tr");    //get the row of table
        var wd,ht;
        for(i = 0; i < rows.length; i++)
        {
            wd=prompt("Enter the width of the strip"+i,"");
            ht=prompt("Enter the height of the strip"+i,"");
            w=Number(wd);
            h=Number(ht);
            display(rows[i],i,w,h);
        }
    }
}
```

```

    }
  }
}
function display(row,i,w,h)
{
  row.style.backgroundColor = colors[i%colors.length]; //using CSS style
  row.style.width = w;
  row.style.height = h;
}
</script></head>
<body onLoad="drawstrips('MyTable')">
<table id="MyTable" cellspacing="10" cellpadding="10">
  <tr><td>Strip 1</td></tr>
  <tr><td>Strip 2</td></tr>
  <tr><td>Strip 3</td></tr>
</table>
</body>

```

Varying color, width
and height of each

Note : For the sake of understanding following output is given for the above application]



Ex. 4.10.8 : Write a JavaScript to find the sum of first n even numbers and display the result. Get the value of n from the user.

AU : May-13, Marks 8

Sol. :

```

<html>
<head>
<script type="text/javascript">
function EvenNumSum(str)
{

```

```

    var n=Number(str);
    sum=0;
    for(i=1;i<n;i++)
    {
    if(i%2==0)
        sum=sum+i;
    }
    document.write("<br>"+" Sum of first "+n+" even numbers is "+sum+"<br>");
}
</script>
</head>
<body>
<script type="text/javascript">
    var n=prompt("Enter the value of n","");
    EvenNumSum(n);
</script>
</body>
</html>

```

4.10.4 Global Functions of JavaScript

Global functions are the top-level functions in JavaScript that are independent of any specific object. These functions use the built-in objects of the JavaScript. Following are some global functions -

1. encodeURI(URI)

This function is used to encode the URI. This function encodes special characters, except: , / ? : @ & = + \$ #.

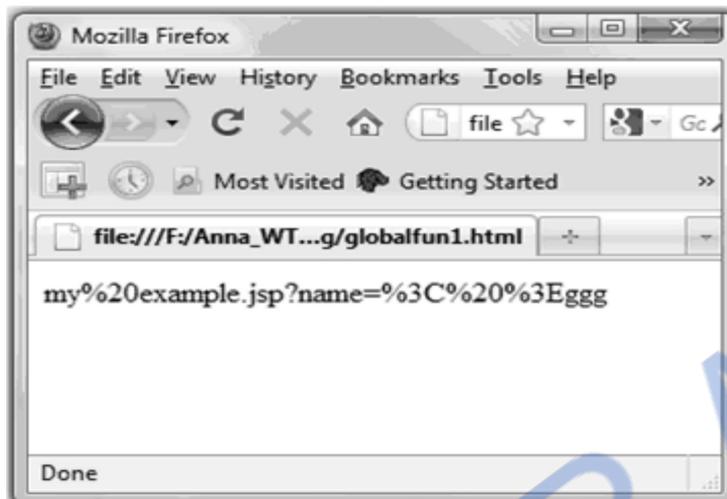
For example

globalfun1.html

```

<html>
<body>
<script type="text/javascript">
    var uri="my example.jsp?name=< >ggg"
    document.write(encodeURIComponent(uri)+ "<br />");
</script>
</body>
</html>

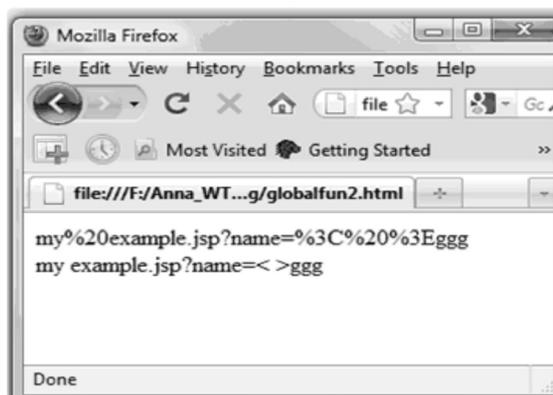
```

Output**2. decodeURI**

This function decodes the encoded URI.

For example**globalfun2.html**

```
<html>
<body>
  <script type="text/javascript">
    var uri="my example.jsp?name=< >ggg"
    document.write(encodeURIComponent(uri)+ "<br />");
    document.write(decodeURI(uri)+ "<br />");
  </script>
</body>
</html>
```

Output

3. parseInt

This function parses a string and returns the integer value.

The syntax is -

```
parseInt(string,radix)
```



String is parsed to obtain integer value

It represents the numerical system such as octal, decimal or hexadecimal. It is optional parameter.

Example

globalfun3.html

```
<html>
<body>
  <script type="text/javascript">
    document.write(parseInt("100"));
  </script>
</body>
</html>
```

The output of above code will be 100.

Similar to the **parseInt** function the **parseFloat** function is used to obtain the real value from the string.

4. eval

The eval function is used to evaluate the expression.

The syntax is

```
eval(string);
```

Example

globalfun4.html

```
<html>
<body>
  <script type="text/javascript">
    document.write(eval("2+3*5"));
  </script>
</body>
</html>
```

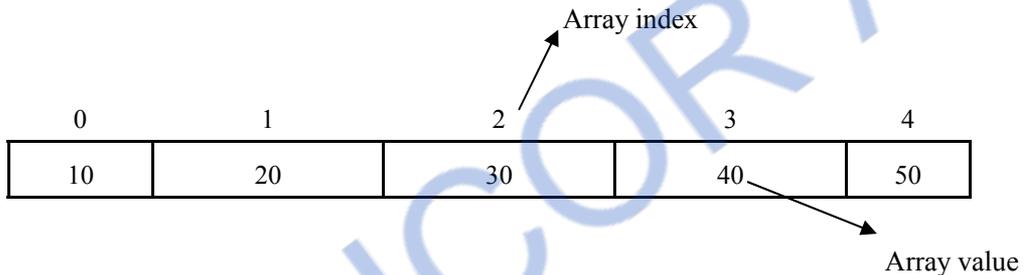
The output of above code will be 17.

University Questions

1. Explain how functions can be written in JavaScript with an example. **AU : May-11, Marks 8**
2. Explain how local and global functions can be written using JavaScript. **AU : May-12, Marks 8**

4.11 Arrays**AU : May-08,12, Dec.-13, Marks 8**

- Arrays is a collection of similar type of elements which can be referred by a common name.
- Any element in an array is referred by an array name followed by “[“ followed by position of the element followed by].
- The particular position of element in an array is called array index or subscript.

For example –**Fig. 4.11.1 Arrays**

- Normally the first element in an array is stored at 0th location, however we can start storing the element from any position.

4.11.1 Array Declaration

- In JavaScript the array can be created using **Array** object.
- Suppose, we want to create an array of 10 elements then we can write,


```
var ar = new Array(10);
```
- Using new operator we can allocate the memory dynamically for the arrays.
- In the brackets the size of an array is mentioned and the var ar denotes the name of the array. Thus by the above sentence an array ar will be created in which we can store 10 elements at the most. Sometimes the above statement can be written like this

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

4.11.2 Array Initialization

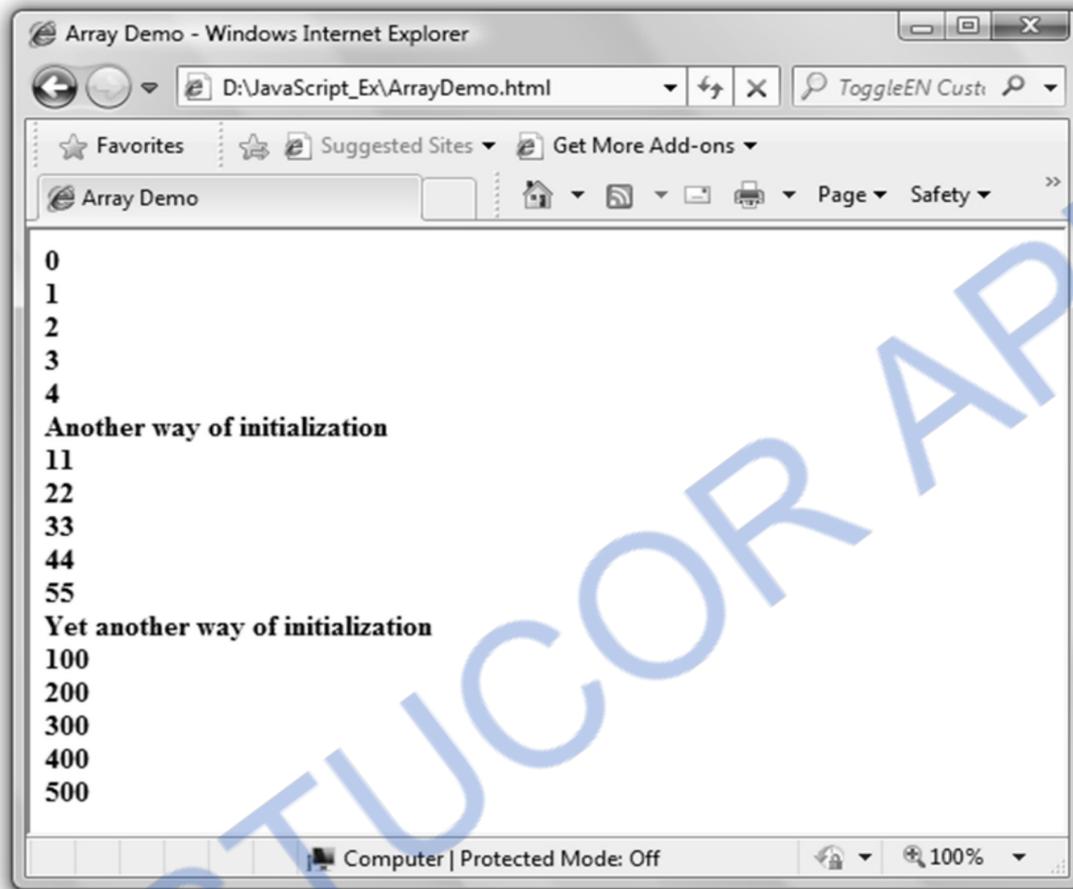
Let us see how to store some elements in an array.

- **JavaScript Program [ArrayDemo.html]**

```
<!DOCTYPE html>
<html>
<head>
  <title>Array Demo</title>
</head>
<body>
<strong>
  <script type="text/javascript">
    a=new Array(5);//creation of array
    for(i=0;i<5;i++)
    {
      a[i]=i;
      document.write(a[i]+"<br>");//displaying array
    }
    document.write("Another way of initialization"+"<br>");
    b=new Array(11,22,33,44,55);//creation of array
    for(i=0;i<5;i++)
    {
      document.write(b[i]+"<br>");//displaying array
    }
    document.write("Yet another way of initialization"+"<br>");
    var c=[100,200,300,400,500];//creation of array
    for(i=0;i<5;i++)
    {
      document.write(c[i]+"<br>");//displaying array
    }
  </script>
</strong>
</body>
</html>
```

As you can notice that, in above JavaScript an array can be initialized in three different ways which is shown by boldface. Hence an output of above script will be

Output



There is one control structure in JavaScript which is closely associated with array elements and such a control structure is **for...in**. Let us see a simple JavaScript which makes use of **for-in** control structure to display elements of an array.

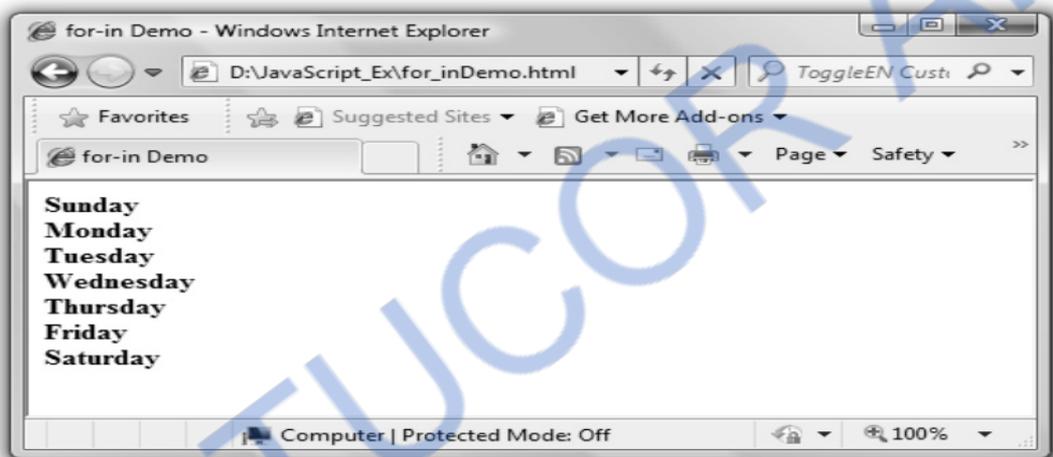
JavaScript [for_inDemo.html]

```
<!DOCTYPE>
<html>
<head>
  <title>for-in Demo</title>
</head>
<body>
<strong>
  <script type="text/javascript">
    Days=new Array();
    Days[0]="Sunday";
    Days[1]="Monday";
    Days[2]="Tuesday";
```

```

Days[3]="Wednesday";
Days[4]="Thursday";
Days[5]="Friday";
Days[6]="Saturday";
for(i in Days)
{
  document.write(Days[i]+"<br>");
}
</script>
</strong>
</body>
</html>

```

Output

Ex. 4.11.1 : Write a javascript that reads ten numbers and displays the count of negative numbers, the count of positive numbers and the count of zero from the list.

Sol. :

```

<html>
  <body>
    <script type="text/javascript">
      ar=new Array();
      ar[0]=10;
      ar[1]=0;
      ar[2]=-9;
      ar[3]=0;
      ar[4]=20;
      ar[5]=30;
      ar[6]=3;
      ar[7]=88;
      i=0;
      for(i in ar)

```

} Reading ten numbers

```

{
  document.write(ar[i]+"<br/>");
  i++;
}
positive_count=0;
negative_count=0;
zero_count=0;
i=0;
for(i in ar)
{
  if(ar[i]==0)
    zero_count++;
  else if(ar[i]>0)
    positive_count++;
  else
    negative_count++;
  i++;
}
document.write("<br>"+ " Total Number of positive elements are
document.write(" Total Number of negative elements are
document.write(" Total Number of zero elements are "+zero_count+"<br/>");
</script>
</body>
</html>

```

Displaying the ten numbers

Computing the count for zero, positive and negative numbers

4.11.3 Two Dimensional Array

Actually JavaScript does not support the multidimensional arrays. Hence for defining the 2D array we make use of single dimensional array, how? Here it is -

JavaScript[TwoDArray.html]

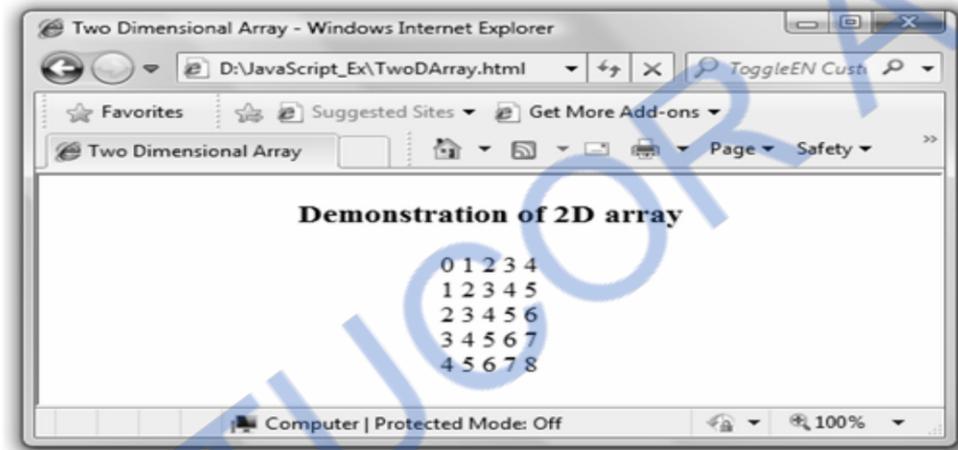
```

<!DOCTYPE html>
<html>
<head>
  <title>Two Dimensional Array</title>
</head>
<body>
  <center>
    <h3> Demonstration of 2D array</h3>
    <script type="text/javascript">
      a=new Array();//creation of rows of array
      for(i=0;i<5;i++)
      a[i]=new Array();//creating columns of array
      for(i=0;i<5;i++)
      {

```

```
for(j=0;j<5;j++)
{
  a[i][j]=i+j;
  document.write(a[i][j] + " ");
}
document.write("<br>");
}
</script>
</center>
</body>
</html>
```

Output



There is another way by which the two dimensional array can be initialized. This method is represented in the following Script

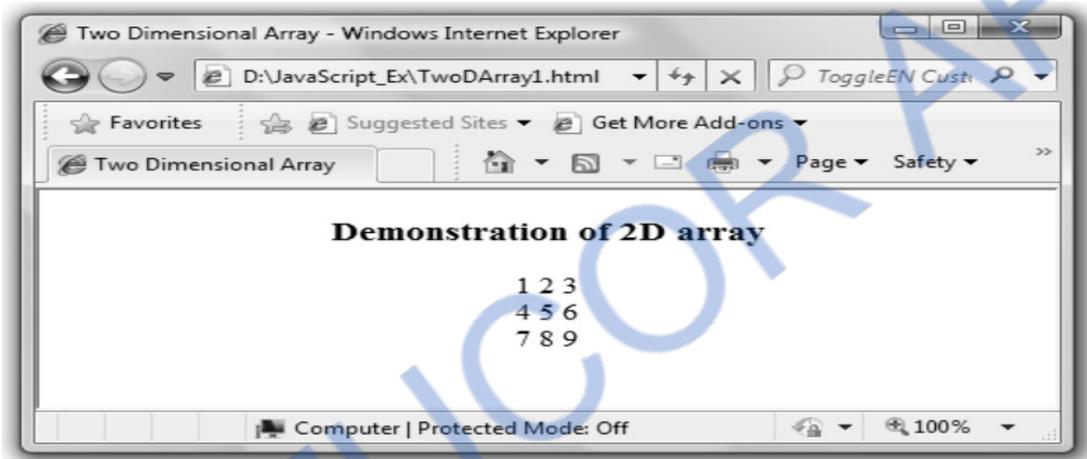
JavaScript[TwoDArray1.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Two Dimensional Array</title>
</head>
<body>
  <center>
    <h3> Demonstration of 2D array</h3>
    <script type="text/javascript">
      var a= [ [1,2,3],
               [4,5,6],
               [7,8,9]
             ]; //creation of array
      for(i=0;i<3;i++)
```

```

{
for(j=0;j<3;j++)
{
document.write(a[i][j] + " ");
}
document.write("<br>");
}
</script>
</center>
</body>
</html>

```

Output

Ex. 4.11.2 : Use a one-dimensional array and write a script to solve following problem . Read in 20 numbers, each of which is between 1 and 100. As each number is read, print it only if it is not a duplicate of a number that has already been read.

AU : May- 08, Marks 8

Sol. : duplicate.html

```

<html>
<head>
<script type="text/javascript">
function change()
{
a=new Array(20);
for(i=0;i<=19;i++)
{
var input_str=prompt("Enter the any number between 1 to 100","");
var val=Number(input_str);
a[i]=val;
}
for(i=0;i<=18;i++)
{

```

```
for(j=i+1;j<=19;j++)
{
  if(a[i]>a[j])//before checking duplicate elements
  {
    temp=a[i]; //arrange them in sorted order
    a[i]=a[j];
    a[j]=temp;
  }
}
}
document.write("The elements (without considering duplicate elements)are...");
for(i=0;i<=18;i++)
{
  flag=0;
  for(j=i+1;j<=19;j++)
  {
    if(a[i]==a[j])//checking if two adjacent elements are duplicate
    {
      a[j]=-1;//-1 is assigned to duplicate elements
      flag=1;//flag indicates a[i] and a[j] are duplicate
    }
  }
  if(flag==1)//flag=1 represents that a[i] element
  a[i]=-1;//needs to be marked as duplicate
}
for(i=0;i<=19;i++)
{
  if(a[i]>0) //-1 means duplicate elements
  document.write(a[i] + ",");//elements that are not -1 are unique elements
}
}
</script>
<title>Eliminating Duplicate Elements</title>
</head>
<body onload=change()>
</body>
</html>
```

University Questions

1. Explain the way in which JavaScript handles arrays with example.
2. Explain the JavaScript array handling and array methods.

AU : May-12, Marks 8

AU : Dec.-13, Marks 8

4.12 Standard Objects**AU : Dec.-09, 11, 13, Marks 16**

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

4.12.1 Math Objects

- For performing the mathematical computations there are some useful methods available from **math** object.
- For example if we want to find out minimum of two numbers then we can write -

```
document.write(math.min(4.5,7.8));
```

The above statement will result in 4.5. Thus using various useful methods we can perform many mathematical computations.

- Here are some commonly used methods from **math** object.

Method	Meaning
sqrt(num)	This method finds the square root of num.
abs(num)	This method returns absolute value of num.
ceil(num)	This method returns the ceil value of num. For example ceil(10.3) will return 11.
floor(num)	This method returns the floor value of num. For example floor(10.3) will return 10.
log(num)	This method returns the natural logarithmic value of num. For example log(7.9) will return 2.

pow(a,b)	This method will compute the ab. For example pow(2,5) will return 32.
min(a,b)	Returns the minimum value of a and b.
max(a,b)	Returns the maximum value of a and b.
sin(num)	Returns the sine of num.
cos(num)	Returns the cosine of num.
tan(num)	Returns the tangent of num.
exp(num)	Returns the exponential value i.e. enum .

JavaScript Program[MathDemo.html]

```
<!DOCTYPE html>
<html >
<head>
  <title>Math Demo</title>
</head>
<body>
  <center>
    <h3>Square root of 100 is </h3>
    <script type="text/javascript">
      var num=100;
      document.write("<h3>" + Math.sqrt(num) + "</h3>");
    </script>
  </center>
</body>
</html>
```

Output



4.12.2 Number Objects

Various properties of number object are -

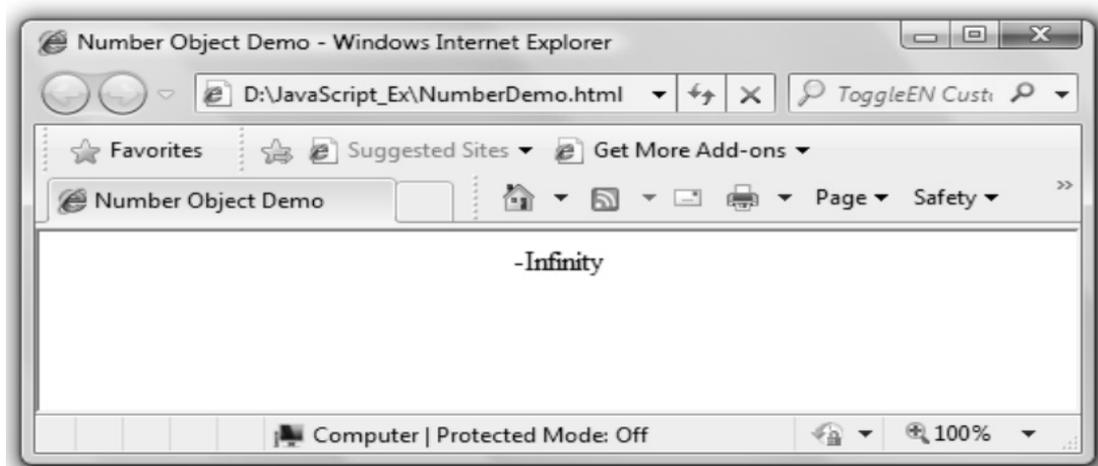
Property	Meaning
MAX_VALUE	Largest possible number gets displayed.
MIN_VALUE	Smallest possible number gets displayed.
NaN	When not a number then NaN is displayed.
PI	The value of PI gets displayed.
POSITIVE_INFINITY	The positive infinity gets displayed.
NEGATIVE_INFINITY	The negative infinity gets displayed.

Using **Number.property_name** we can display the property value. Following JavaScript uses the property of negative infinity.

JavaScript[NumberDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Number Object Demo</title>
</head>
<body>
  <center>
    <script type="text/javascript">
      document.write(Number.NEGATIVE_INFINITY);
    </script>
  </center>
</body>
</html>
```

Output



4.12.3 Date Objects

- This object is used for obtaining the date and time.
- This date and time value is based on computer's local time (system's time) or it can be based on GMT(**Greenwich Mean Time**).
- Nowadays this GMT is also known as UTC i.e. **Universal Co-ordinated Time**. This is basically a world time standard.
- Following are the commonly used methods of Date object.

Method	Meaning
getTime()	It returns the number of milliseconds. This value is the difference between the current time and the time value from 1st January 1970.
getDate()	Returns the current date based on computers local time.
getUTCDate()	Returns the current date obtained from UTC.
getDay()	Returns the current day. The day number is from 0 to 6 i.e. from Sunday to Saturday.
getUTCDay()	Returns the current day based on UTC. The day number is from 0 to 6 i.e. from Sunday to Saturday.
getHours()	Returns the hour value ranging from 0 to 23, based on local time.
getUTCHours()	Returns the hour value ranging from 0 to 23, based on UTC timing zone.
getMilliseconds()	Returns the milliseconds value ranging from 0 to 999, based on local time.
getUTCMilliseconds()	Returns the milliseconds value ranging from 0 to 999, based on UTC timing zone.
getMinutes()	Returns the minute value ranging from 0 to 59, based on local time.
getUTCMinutes()	Returns the minute value ranging from 0 to 59, based on UTC timing zone.
getSeconds()	Returns the second value ranging from 0 to 59, based on local time.
getUTCSeconds()	Returns the second value ranging from 0 to 59, based on UTC timing zone.
setDate(value)	This function helps to set the desired date using local timing or UTC timing zone.

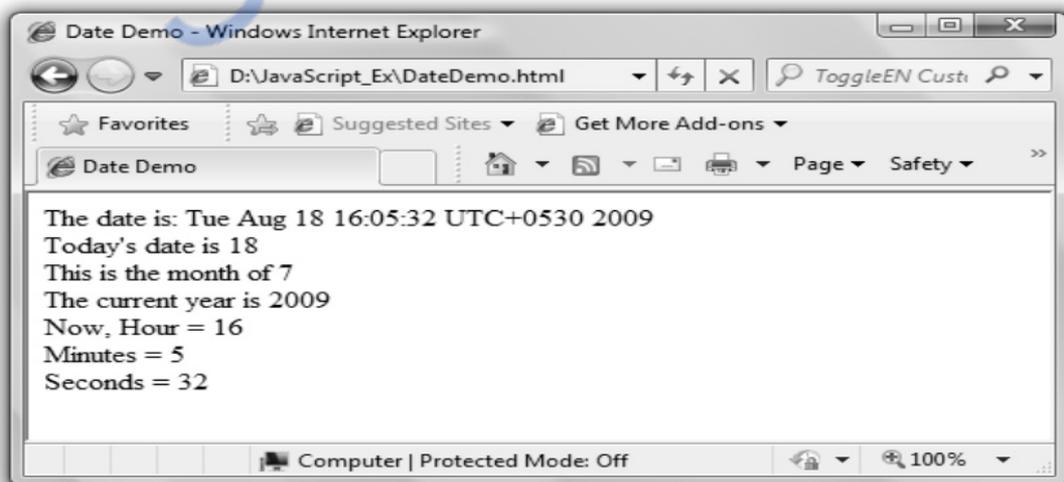
setHour(hr,minute,second,ms)	This function helps to set the desired time using local or UTC timing zone. The parameters that can be passed to this function are hour,minute,seconds and milliseconds. Only hour parameter is compulsory and rest all are the optional parameters.
------------------------------	--

In the following web document we are making use of **Date()** object and some useful methods of it.

JavaScript[DateDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Date Demo</title>
</head>
<body>
  <script type="text/javascript">
    var my_date=new Date();
    document.write("The date is: "+my_date.toString()+"<br>");
    document.write("Today's date is "+my_date.getDate()+"<br>");
    document.write("This is the month of "+my_date.getMonth()+"<br>");
    document.write("The current year is "+my_date.getFullYear()+"<br>");
    document.write("Now, Hour = "+my_date.getHours()+"<br>");
    document.write("Minutes = "+my_date.getMinutes()+"<br>");
    document.write("Seconds = "+my_date.getSeconds()+"<br>");
  </script>
</body>
</html>
```

Output



Script Explanation:

1. In the above script we have created an instance *my_date* of the object **Date()**.
2. Then using **my_date.toString()** method we can display the current date along with the time.
3. Then using the functions like `getDate()`, `getMonth()`, `getFullYear()` we are displaying the date, month and year separately.
4. Similarly using the functions `getHours()`, `getMinutes()` and `getSeconds()` we can display the current hour, minute and seconds separately.

4.12.4 Boolean Objects

- This object is the simplest kind of object which is used especially when we want to represent **true** and **false** values.
- Here is a simple javascript in which the Boolean type variable is used -

Javascript Program

```
<html>
<body>
<script type="text/javascript">
var temp=new Boolean(false);
document.write("<b>"+ "The boolean value is: ");
document.write(temp.toString());
</script>
</body>
</html>
```

Output

4.12.5 String Objects

- String is a collection of characters.
- In JavaScript using **string** object many useful string related functionalities can be exposed off.
- Some commonly used methods of string object are concatenating two strings, converting the string to upper case or lower case, finding the substring of a given string and so on.
- Here is a listing of some methods of string.

Method	Meaning
concat(str)	This method concatenates the two strings. For example s1.concat(s2) will result in concatenation of string s1 with s2.
charAt(index_val)	This method will return the character specified by value index_val.
substring(begin,end)	This method returns the substring specified by begin and end character.
toLowerCase()	This function is used to convert all the uppercase letters to lower case.
toUpperCase()	This function is used to convert all the lowercase letters to upper case.
valueOf()	This method returns the value of the string.

There is one important property of string object and that is **length**. For example

```
var my_str="Hello";
var len;
len=my_str.length;
```

Length of the string "Hello" will be stored in the variable len

Here is sample program in which some methods of string object are used.

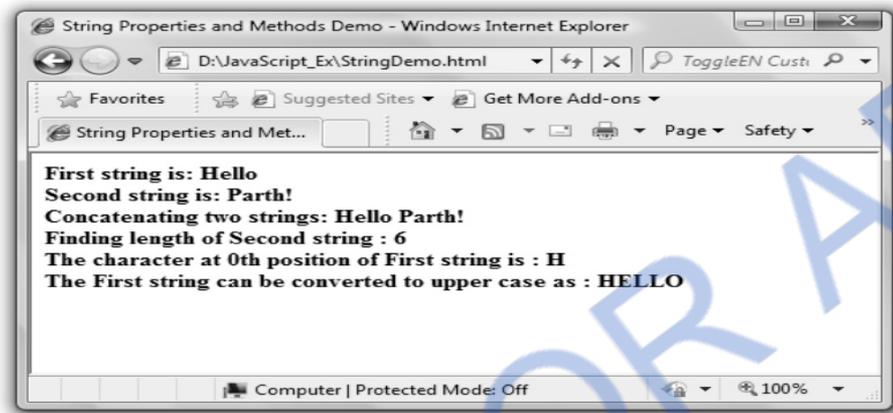
JavaScript[StringDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>String Properties and Methods Demo</title>
</head>
<body>
<strong>
  <script type="text/javascript">
    var s1="Hello ";
    var s2="Parth!";
    document.write("First string is: "+s1+"<br>");
    document.write("Second string is: "+s2+"<br>");
    document.write("Concatenating two strings: "+s1.concat(s2)+"<br>");
    document.write("Finding length of Second string : "+s2.length+"<br>");
    document.write("The character at 0th position of First string is : "+s1.charAt(0)+"<br>");
```

```

document.write("The First string can be converted to upper case as : "+s1.toUpperCase());
</script>
</strong>
</body>
</html>

```

Output

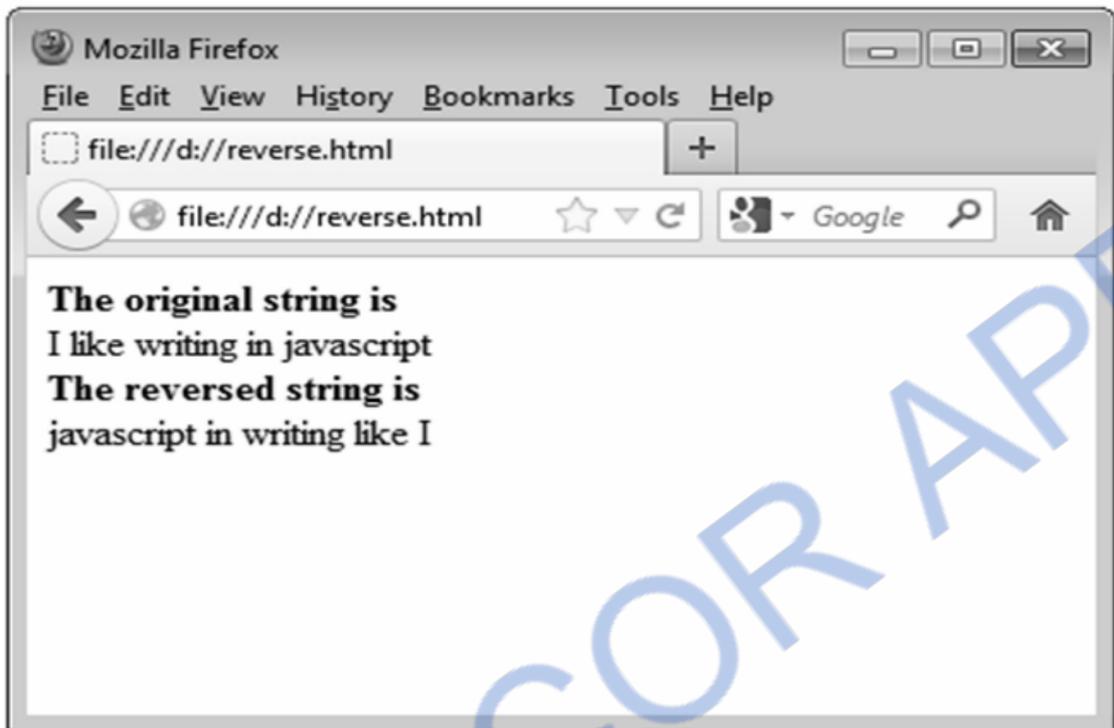
Ex. 4.12.1 : Write a script that inputs a line of text, tokenizes it with String method split and displays the tokens in reverse order.

Sol. :

```

<html>
<body>
<script type="text/javascript">
var str=new String("I like writing in javascript");
var a=new Array();
a=str.split(' ');
document.write("<strong>The original string is</strong>"+"<br/>");
for(i=0;i<a.length;i++)
document.write(a[i]+" ");
document.write("<br/>");
document.write("<strong>The reversed string is</strong>"+"<br/>");
for(i=a.length-1;i>=0;i--)
document.write(a[i]+" ");
</script>
</body>
</html>

```



4.12.6 Object Creation and Modification

- The web designer can create an object and can set its properties as per his requirements.
- The object can be created using new expression.
- Initially the object with no properties can be set using following statements
`Myobj=new Object();`
- Then using dot operator we can set the properties for that object.
- The object can then be modified by assigning the values to this object.

JavaScript[ObjectDemo.html]

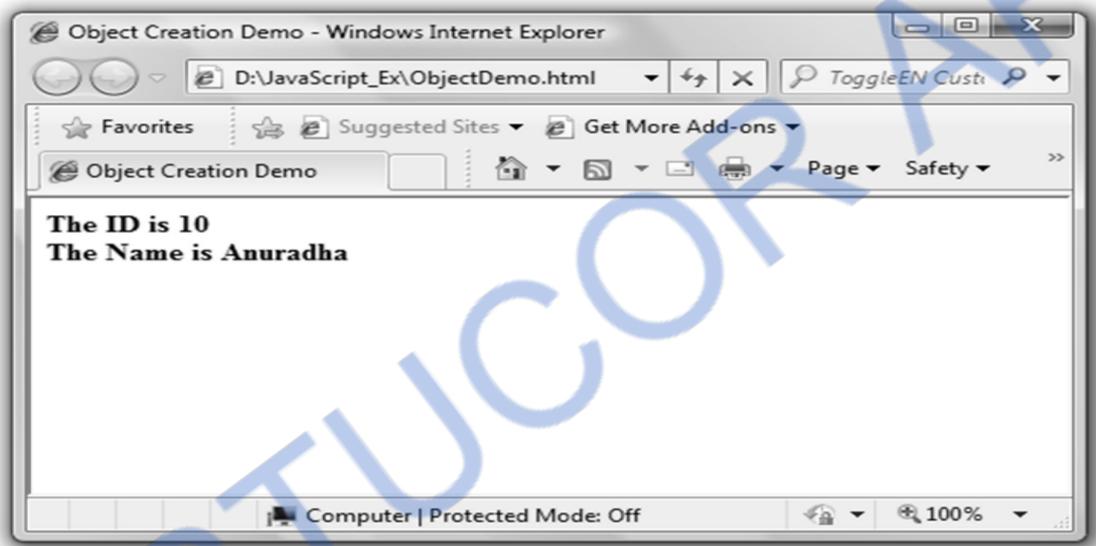
```
<!DOCTYPE html>
<html>
<head>
  <title>Object Creation Demo</title>
</head>
<body>
<strong>
  <script type="text/javascript">
    var Myobj;
    var n,i;
  //creating an object
```

```

Myobj=new Object();
//setting the propertis for newly created object
Myobj.id=10;
Myobj.name="Anuradha";
document.write("The ID is "+Myobj.id+"<br/>");
document.write("The Name is "+Myobj.name);
</script>
</strong>
</body>
</html>

```

Output



Alternative method of creating an object and modifying the properties is as given below -
`var Myobj={name:"Anuradha",id:10};`

Then using **document.write** statement we can display the property values as follows -

```

document.write("The ID is "+Myobj.id+"<br/>");
document.write("The Name is "+Myobj.name);

```

The property of created object can be deleted using an expression **delete**. Normally the property of an object is deleted in order to free the allocated memory so that this memory can be reused by other process.

University Questions

1. Discuss javascript array object in detail. **AU : Dec.-11, Marks 8**
2. Explain the javascript object Math. **AU : Dec.-13, Marks 8**
3. What are the methods associated with array object in JavaScript ? Explain each one with example. **AU : Dec.-09, Marks 16**

4.13 Form Processing in JavaScript**AU : Dec.-09, May-08, 14, Marks 8**

- Various control objects can be placed on the form. These control objects are also called as widgets.
- Various widgets used in JavaScript are -Text box, Push button, Radio button, Check box and so on.

We have discussed how to place various widgets on HTML page in section 2.10. JavaScript can be associated with the form elements for validation purpose. Let us now discuss the form validation with the help of examples

Ex. 4.13.1 : Write a javascript to display a welcome message using alert whenever a button of a html form is pressed.

AU : Dec.-09, Marks 8**Sol. :**

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function my_fun()
{
alert("WELCOME");
}
</script>
</head>
<body>
<center>
<form>
<input type="button" name="Button1" value="Click" onClick="my_fun()"/>
</form>
</center>
</body>
</html>
```

Ex. 4.13.2 : Write a Javascript that inputs a telephone number as a string in the form of (555) 5555555. The script should use string method split to extract the area code as a token, the first 3 digits of the phone number as a token, the last 4 digits of the phone number as a token.

Display the area code in one text field and seven digit phone number in another text field.

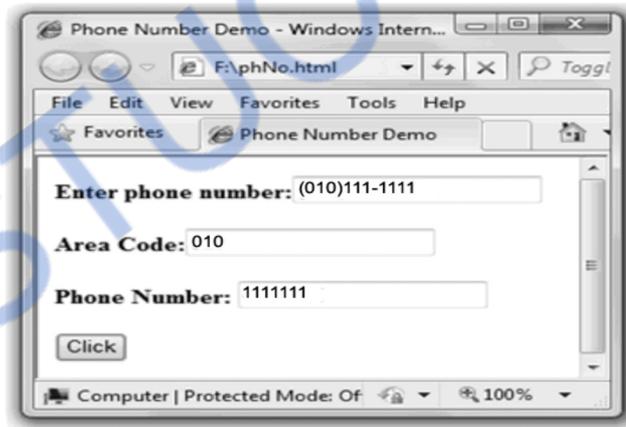
AU : May-08, CSE, Marks 8**Sol. : HTML Document[phNo.html]**

```
<html>
<head>
<title>Phone Number Demo</title>
<script type="text/javascript">
function TestString()
```

```

{
  var str=document.form1.input.value;
  var i=str.match(/"(\d{3})"\" "(\d{3})"-\"(\d{4})/);
  var temp1=str.split("(");
  var temp2=temp1[1].split(")")
  document.form1.area.value=temp2[0];
  var temp3=temp2[1].split(" ");
  var temp4=temp3[1].split("-")
  document.form1.phnum.value=temp4[0]+temp4[1];
}
</script>
</head>
<body>
  <form name="form1">
    <b>Enter phone number:</b><input type="text" name="input" value=""><br/><br/>
    <b>Area Code:</b><input type="text" name="area" value=""><br/><br/>
    <b>Phone Number:</b> <input type="text" name="phnum" value=""><br/><br/>
    <input type="button" value="Click"onclick=TestString(input)>
  </body>
</html>

```

Output

Ex. 4.13.3 : Write a JavaScript for password verification.

Sol. :

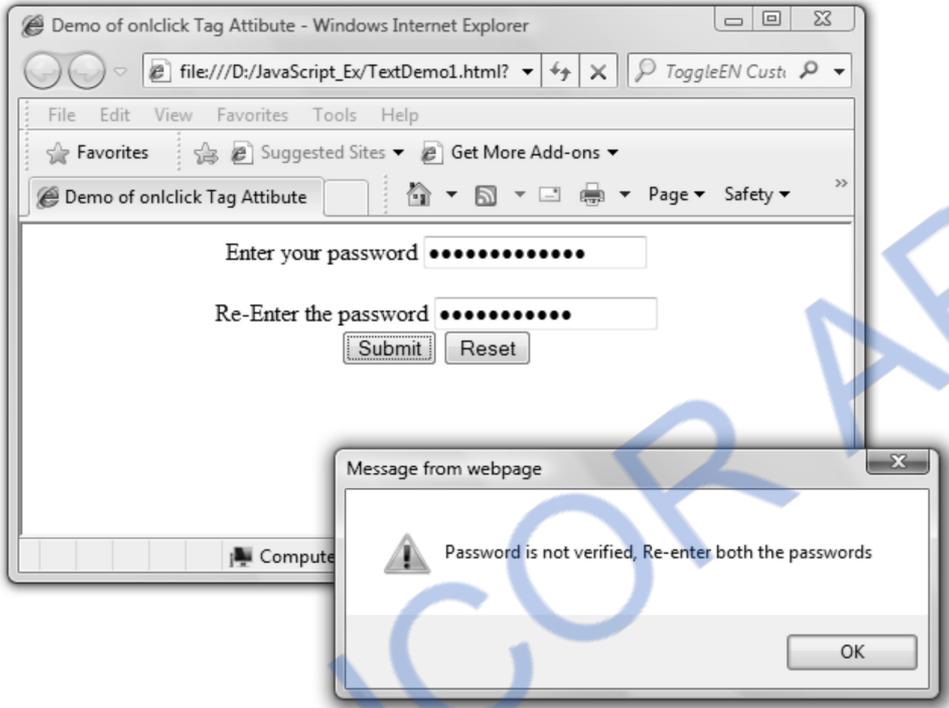
```

<!DOCTYPE html>
<html>
<head>
  <title>Demo of onclick Tag Attibute</title>
<script type="text/javascript">
  function my_fun()
  {

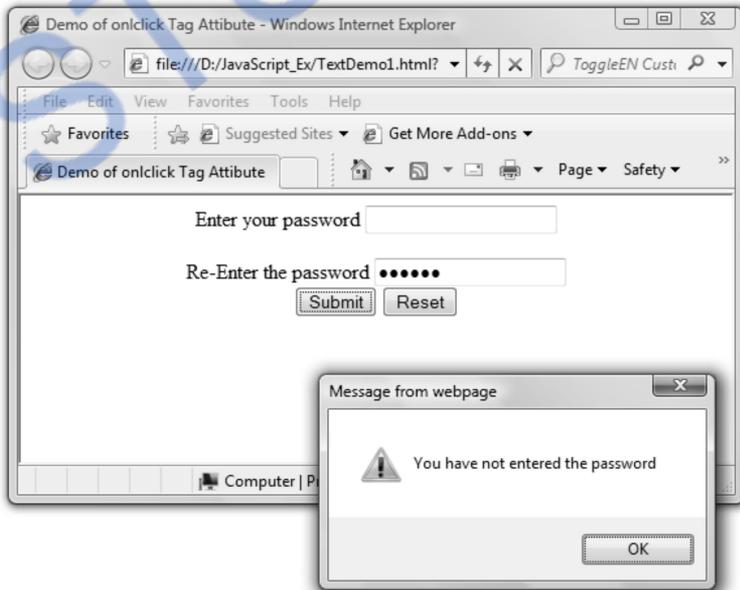
```

```
var mypwd=document.getElementById("pwd");
var my_re_pwd=document.getElementById("re_pwd");
if(mypwd.value=="")
{
    alert("You have not entered the password");
    mypwd.focus();
    return false;
}
if(mypwd.value!=my_re_pwd.value)
{
    alert("Password is not verified, Re-enter both the passwords");
    mypwd.focus();
    mypwd.select();
    return false;
}
else
{
    alert("Congratulations!!!");
    return true;
}
}
</script>
</head>
<body>
<center>
<form id="form1">
<label> Enter your password
<input type="password" value="" id="pwd" />
</label>
<br/><br/>
<label> Re-Enter the password
<input type="password" value="" id="re_pwd" onblur="my_fun();"/>
</label><br/>
<input type="submit" value="Submit" name="submit" onsubmit="my_fun();"/>
<input type="reset" value="Reset" name="reset"/><br/>
</form>
</center>
</body>
</html>
```

Output(Run1)



Output(Run2)



Output(Run3)



Ex. 4.13.4 : Write JavaScript to validate a form consisting of Name, Age, Address, EmailID, hobby (check box), Gender (radio box), country (Drop down menu).

Sol. : ApplicationForm.html

```

<html>
<head>
<title>The Student Registration Form</title>
<script type=text/javascript>
function validate()
{

    var i;
    var name_str=document.my_form.name;
    var phoneID=document.my_form.ph_txt;
    var ph_str=document.my_form.ph_txt.value;
    var str=document.my_form.Email_txt.value;
    if((name_str.value==null) || (name_str.value==""))
    {
        alert("Enter some name")
        return false
    }
    if(document.my_form.Age_txt.value=="")// validating age
    {
        alert("Enter Some Age")
        return false
    }
    if((document.my_form.Age_txt.value<"5")&&(document.my_form.Age_txt.value>"21"))
    {

```

Validating Name

```
alert("Invalid Age")
return false
}
```

```
if(ph_str.length<1 || ph_str.length>11)
{
alert("Invalid length of Phone Number")
return false
}
for (i = 0; i < ph_str.length; i++)
{
    var ch = ph_str.charAt(i);
    if (((ch < "0" || (ch > "9"))){
alert("Invalid Phone Number")
```

Validating Phone Number

```
phoneID.focus()
return false
}
}
```

```
var index_at=str.indexOf("@")
var len=str.length
var index_dot=str.indexOf(".")
var emailID=document.my_form.Email_txt
if ((emailID.value==null) || (emailID.value==""))
{
alert("Please Enter your Email ID")
emailID.focus()
return false
}
if (str.indexOf("@")==-1)
{
    alert("Invalid E-mail ID")
    return false
}
if (str.indexOf(".")==-1 || str.indexOf(".")==0
    || str.indexOf(".")==index_at)
{
    alert("Invalid E-mail ID")
    return false
}
if (str.indexOf("@",(index_at+1))!=-1)
{
    alert("Invalid E-mail ID")
    return false
}
if (str.indexOf(" ")!=-1)
```

Validating Email ID

Validating Email ID

```

    {
        alert("Invalid E-mail ID")
        return false
    }
    if (!document.my_form.group1[0].checked && !document.my_form.group1[0].checked)
    {
        alert("Please Select Sex");
        return false;
    }
    if (!document.my_form.group1[0].checked && !document.my_form.group1[0].checked)
    {
        alert("Please Select Sex");
        return false;
    }
    return true
}
</script>
</head>
<body bgcolor=aqua>
<center><h3>Application Form</h3></center>
<form name=my_form onsubmit=validate()>
<strong>Name: </strong>
<input type=text name=name><br/>

<strong>Age: </strong>
<input type=text name=Age_txt><br/>

<strong>Phone No:</strong>
<input type=text name=ph_txt><br/>
<strong>Email: </strong>
<input type=text name=Email_txt><br/><br/>
<strong>Sex: </strong>
<input type="radio" name="group1" value="Male">Male
<input type="radio" name="group1" value="Female">Female<br/><br/><br/>
<strong>Hoby: </strong>
<input type="checkbox" name ="option1" value="Singing">Singing<br/>
<input type="checkbox" name ="option1" value="Reading">Reading<br/>
<input type="checkbox" name ="option1" value="T.V.">Watching T.V<br/>
<br/><br/>
<strong>Country:</strong>
<select name="My_Menu">
<option value="India">India</option>
<option value="China">China</option>

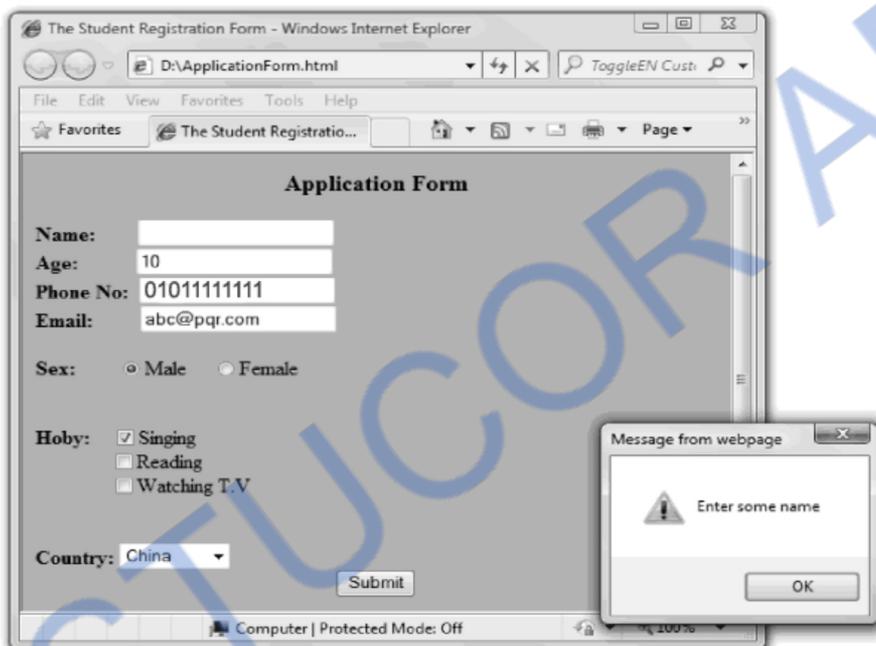
```

```

<option value="Shrilanka">Shrilanka</option>
</select>
<center>
<input type="submit" value="Submit"> </br>
</center>

</body>
</html>

```

Output

Ex. 4.13.5 : Design a HTML form for validation the users with fields user name and password and ok button which should receive the input from the user and responses as authorized or invalid user name or invalid password.

AU : May-14, Marks 8

Sol. :

```

<head>
<title>HTML FORM</title>
<script type="text/javascript">
function validate()
{
var user=document.form1.username;
var pass=document.form1.password;
if((user.value==null) || (user.value==""))
{
alert("Enter some user name");
}
}

```

```
if((pass.value == null) || (pass.value == ""))
{
    alert("Enter some password");

}
if(user.value == "admin")
{
    alert("Valid user name");

}
if(pass.value == "password")
{
    alert("Valid password");
}
else
    alert("Invalid username/password");
}
</script>
</head>
<body>
    <form id="form1" name="form1" onsubmit=validate()>
    <table width="510" border="0" align="center">
    <tr> <td>Username:</td>
    <td><input type="text" name="username"/></td>
    </tr>
    <tr> <td>Password</td>
    <td><input type="password" name="password"/></td>
    </tr>
    <tr>
    <td>&nbsp;</td>
    <td><input type="submit" name="button" value="OK" /></td>
    </tr>
    </table>
    </form>
</body>
</html>
```

4.14 JavaScript Debuggers

AU : May-12, Marks 8

- JavaScript is a scripting language which is very much similar to a programming language. It supports the arrays, functions and control statements. Hence debugging the javascript is just similar to debugging any programming language.
- The standard traditional method of debugging the JavaScript is by use of **alert()** to display the values of corresponding variables. But it is a complex method of debugging as within the loop if we write the alert statement then each time the popup window will appear to display the corresponding messages.
- However we can make use of some other API methods for debugging the javascript.
- Another way is to use the **log()** function **console API**.
- **Example Program :** Following is a simple script that illustrates how to debug the values of the variable using the **log()** function of **console**.

```
<!DOCTYPE html>
<html>
<head>
<title>Continue Statement</title>
</head>
<body>
<script type="text/javascript">
  for(i=10;i>=0;i--)
  {
    if(i==5)
    {
      x=i;
      console.log('x is equal to 5', x == 5);
      continue;
    }
    console.log('i= '+i,i == 5);
    document.write(i);
    document.write("<br>");
  }
  document.write("The number "+x+" is missing in above list" );
</script>
</body>
</html>
```

Here `x==5` is true hence it will return **true** along with the message.

Just open the above code in Google Chrome. Press F12, click on the Console to get the debugging done using the log function. Here is the snapshot for the same.

Output

```

10
9
8
7
6
4
3
2
1
0
The number 5 is missing in above list

i = 10 false
i = 9 false
i = 8 false
i = 7 false
i = 6 false
x is equal to 5 true
i = 4 false
i = 3 false
i = 2 false

```

University Question

1. Discuss about javascript debugging.

AU : May-12, Marks 8

4.15 Examples

AU : May-08, 09, Dec.-08, 09, Marks 16

Ex. 4.15.1 : Design web page to create a clock with a timing event.

AU : June- 09, Marks 4

Sol. : clock.html

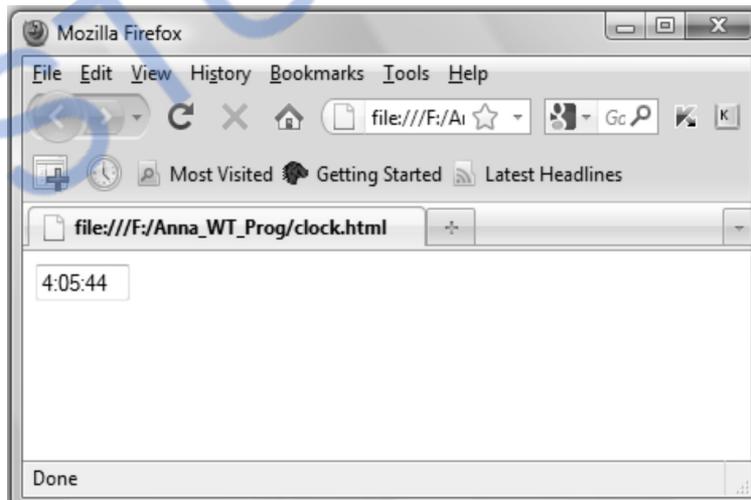
```

<html>
<head>
<script type="text/javascript">
function Display()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
h=ConvertTwelve(h);
m=SingleDigit(m);

```

```
s=SingleDigit(s);
document.getElementById("mytext").value=h+":"+m+": "+s;
t=setTimeout('Display()',500);
}
function ConvertTwelve(h)
{
  if(h >= 12)
  h=h-12;
  return h;
}
function SingleDigit(i)
{
  if (i<10)
  {
    i="0" + i;//writing 0 before single digit number
  }
  return i;
}
</script>
</head>
<body onload="Display">
  <input type="text" value="" size="5" id="mytext"/>
</body>
</html>
```

Output



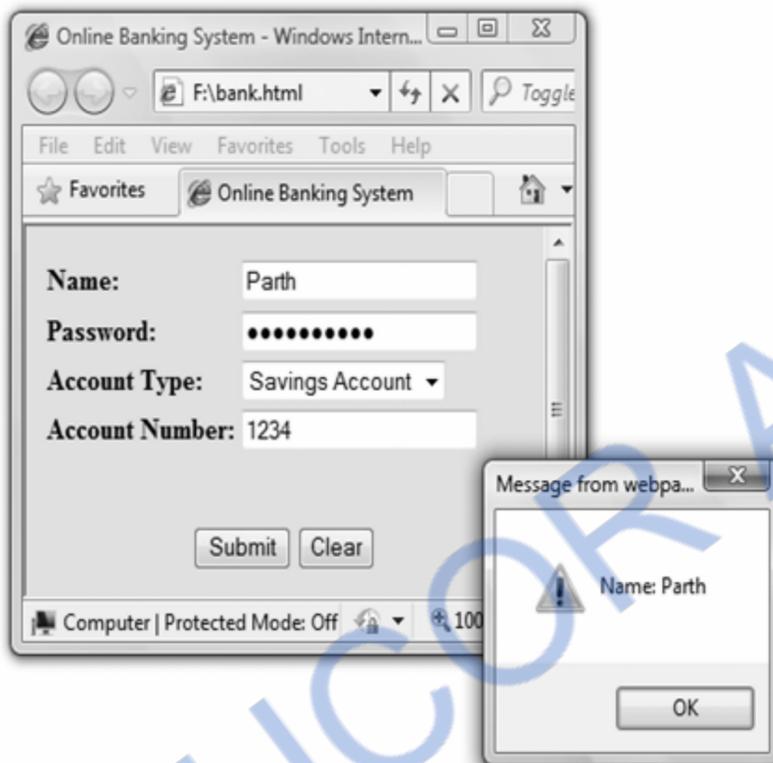
Ex. 4.15.2 : Write HTML program for the registration of new customer to the online banking system (Customer Data collected using a form, after submitting account number and type of account and username, password is displayed as output)

AU : May-08, CSE, Marks 8

Sol. : HTML Document[bank.html]

```
<html>
<head>
  <title>Online Banking System</title>
  <script type="text/javascript">
function fun()
{
  alert("Name: "+form1.userName.value);
  alert("Password: "+form1.pwd.value);
  alert("Account Type: "+form1.MyMenu.value);
  alert("Account Number: "+form1.AccNo.value);
}
</script>
</head>
<body bgcolor="khaki">
  <form name="form1" onsubmit="fun()">
    <table>
      <tr><td><b>Name:</b></td><td><input type="text" name="userName"></tr>
      <tr><td><b>Password:</b></td><td><input type="password" name="pwd"></tr>
      <tr><td><b>Account Type:</b></td>
      <td><select name="MyMenu">
        <option value="Savings Account">Savings Account</option>
        <option value="Current Account">Current Account</option>
      </select>
      </td>
    </tr>
      <tr><td><b>Account Number:</b></td><td><input type="text" name="AccNo"></tr>
    </table>
    <br/><br/>
    <center>
      <input type="submit" value="Submit">
      <input type="reset" value="Clear">
    </center>
  </form>
</body>
</html>
```

Output

**Ex. 4.15.3 : Develop a HTML page which accepts**

- Any mathematical expression
- Evaluate the expression
- Displays the result of the evaluation.

AU : Dec.-08, CSE, Marks 16

Sol. : HTML Document[eval.html]

```
<html>
<head>
  <title>Expression Evaluation</title>
  <script type="text/javascript">
    function Compute()
    {
      document.form1.ans.value=eval(document.form1.my_exp.value);
    }
  </script>
</head>
<body>
  <form name="form1">
    <b>Enter Expression here:</b><br/> <input type="text" name="my_exp">
```



```

if(document.myform.Item4.checked){cost=cost+29;}
document.myform.total.value="$"+cost;
}
</script>
</head>
<body bgcolor="pink">
<form name="myform" action="http://localhost/cgi-bin/purchaseInfo.cgi" enctype="text/plain"
method="post">
<center> <h2>Online shopping Application</h2> </center>
<strong>Personal Details:</strong>
<table>
<tr><td>name:</td><td> <input type="text" name="customer"/></td></tr>
<tr><td>City:</td><td> <input type="text" name="city"/></td></tr>
<tr><td>Phone:</td><td> <input type="text" name="phone"/></td></tr>
<tr><td>Email_Id:</td><td> <input type="text" name="email"/></td></tr>
</table>
<hr/>
<strong>Description:</strong>
<p>Following are some items with their prices.</p>
<p>Choose the required items by checking the chekboxes.</p>
</div>
<hr/>
<strong>Items to be chosen:</strong>
<br/>
<input type="checkbox" name="Item1" value="Item1_chosen" onclick="fun()"> Item One($20.15)
<p><input type="checkbox" name="Item2" value="Item2_chosen" onclick="fun()"> Item
Two($10.10)
<p><input type="checkbox" name="Item3" value="Item3_chosen" onclick="fun()"> Item Three($26)
<p><input type="checkbox" name="Item4" value="Item4_chosen" onclick="fun()"> Item Four($29)
<input type="text" name="total" value="0"/>
<br/><br/>
<center>
<input type="submit" value="Place the Order" />
<input type="reset" value="Clear" />
</center>
</form>
</body>
</html>

```

Step 2 : When user click the Place the order button the submitted input can be collected in a cgi file. The sample cgi file can be written as follows -

CGI Document[purchaseInfo.cgi]

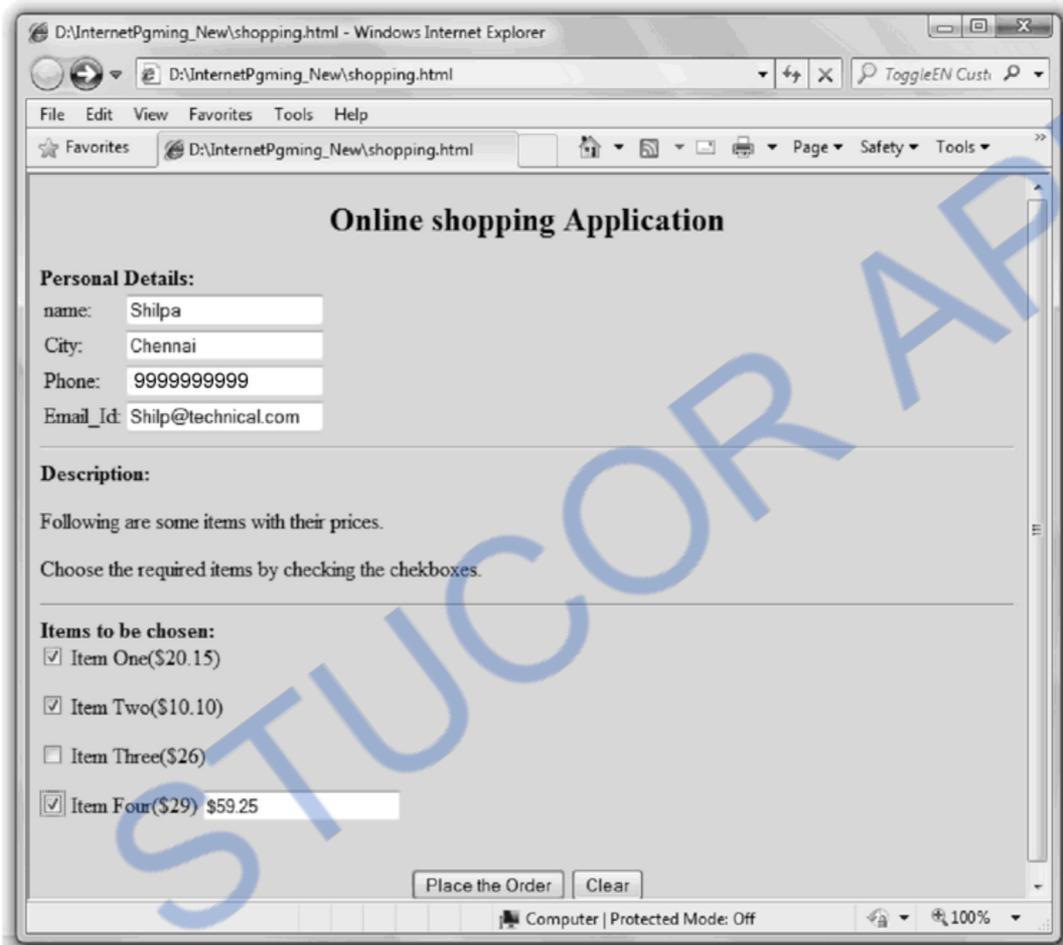
```

#!/program files/perl/bin/perl
use CGI qw(:standard);
print header;
print start_html(-BGCOLOR =>"Khaki");
print h2("Thank you for spending your valuable time with us!!!<br/>");
my($name)=param("customer");

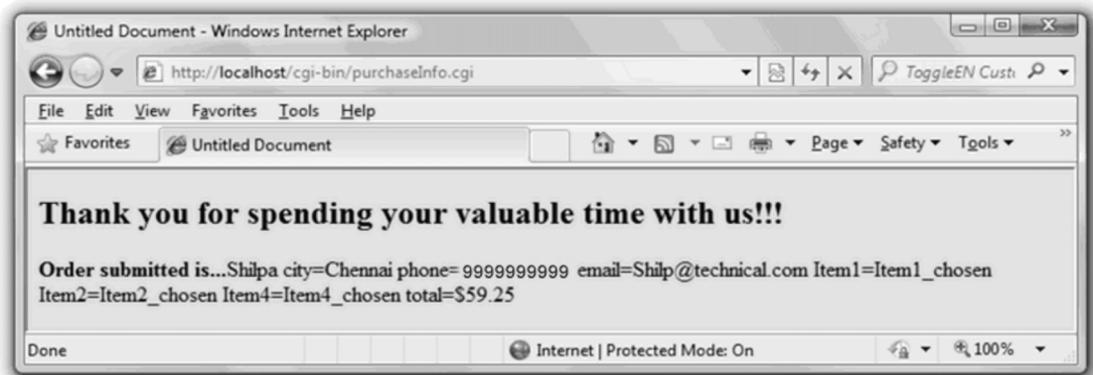
```

```
print"<b>Order submitted is...</b>".$name";  
print end_html;
```

Step 3 : The output can be viewed as follows -



Click the 'Place the order' button and following output can be seen



Ex. 4.15.5 : Write a Javascript to validate radio button, operator field and check box.

Sol. :

JavaScript Program

```

<html>
<head>
<script LANGUAGE="JavaScript">

function ValidateForm(form)
{
    if ((form.gender[0].checked == false ) && ( form.gender[1].checked == false ))
    { alert ( "Please choose your Gender: Male or Female" );
      return false;
    }
    if ((form.status[0].checked == false ) && ( form.status[1].checked == false ))
    { alert ( "Please choose your choice: AC or non AC" );
      return false;
    }
    if ( form.course.selectedIndex == 0 )
    { alert ( "Please select Some Course." );
      return false;
    }
    return true;
}
</script>
</head>
<body>
<br>
<form>
Your Gender: <input type="radio" name="gender" value="Male"> Male
<input type="radio" name="gender" value="Female"> Female
<br/>
Choice: <input type="checkbox" name="status"> AC
<input type="checkbox" name="status"> Non AC
<br/>
Course:
<select name="course">
  <option value="">Select an Option:</option>
  <option value="Computer">Computer</option>
  <option value="Mechanical">Mechanical</option>
  <option value="E&Tc">E&Tc</option>
</select>
<br/><br/>

```

Validating Radio Button

Validating Checkbox

Validating Option

```
<input type="button" name="SubmitButton" value="Submit"
onClick="ValidateForm(this.form)">
<input type="reset" value="Reset">
</form>
</body>
</html>
```

Two Marks Questions with Answers

Q.1 What is the benefit of using JavaScript code in an HTML document ?

AU : Dec.-07, CSE

Ans. : Following are some significant benefits of using JavaScript code in an HTML document

1. The client side data validation can be possible using JavaScript.
2. We can access the HTML elements. That means we can have control over HTML, BODY, TITLE tags.
3. Since JavaScript is based on the programming constructs we can control the logic of a simple scripting language. For example if a mouse pointer is moving over a text then change its colour.
4. JavaScript can determine the visitor's browser and can load the page accordingly.

Q.2 What is the need for client side scripting ?

AU : May-09, CSE

Ans. : Following are the issues that get handled using client side scripting -

1. Simple scripting language like HTML or XML represents the data on the web page in simplest manner.
2. The scripting languages like JavaScript and DHTML are useful for enhancing the interactivity of the user with the web browser.
3. Mathematical evaluation is also possible using the client side scripting.
4. Using client side scripting languages the web page which gets designed for the user provides a graphical user interface using which the user can input his data in an elegant manner. This data can then be submitted to database or any server side scripting language in order to process it further.
5. The input validation can also be possible using client side scripting language like JavaScript.
6. Using cascading stylesheets with the client scripts the look and feel of the web page can be enhanced.

Q.3 List out the objects used in JavaScript with its purpose.

AU : May-09, CSE, May-12

Ans. : Various objects that can be used in JavaScript are -

1. **Math Object** - This object provides useful methods that are required for mathematical computations.

For example : For computing square root of some number `sqrt(num)` method is used.

For finding out minimum value of a and b `min(a,b)` method is used.

2. **Number Object** - This object is used has various numeric properties.

For example : Largest possible number gets displayed using `MAX_VALUE`, for displaying the pi value `PI` property is used.

3. **Date Object** - This object is used for obtaining date and time. For example using `getTime()` function the number of milliseconds can be obtained.

4. **Boolean Object** - This is the simplest object used to obtain true or false values.

5. **String Object** - This object provides many useful string related functionalities.

For example : For finding out the substring of a given string `substring(begin,end)` method can be used.

Q.4 Comment on the statement. "Each object of a class has its own instance of static member variable."

AU : Dec.-08

Ans. : When we create an object for a class then each object has its own distinct copies of variables. Such variables are called instance variables. For example if declare a class `MyCar` as follows

```
class MyCar
{
private String color;
private String model;
private double price;
}
```

This class has instance variables `color`, `model` and `price`. Now, each object of class `MyCar` will have its own values for these variables. These values will be static in nature. These objects can be stored at different memory locations. Thus every instance of the class(object) shares a class variables. Hence it is said that every object of a class has its own instance of static member variable.

Q.5 What is JavaScript statement ? Give an example.

AU : Dec.-11

Ans. : The assignment statement in JavaScript is very much similar to C. For example

```
Sum+ =10
Sum=sum+10
```

Can be written in JavaScript.

Q.6 Give any three uses of JavaScript.

Ans. :

- 1) The JavaScript can be used to create some web applications such as Calculator, Calender, Paint like applications.
- 2) JavaScript can be used to detect the visitor's browser.
- 3) JavaScript can be used to validate the data.

Q.7 What kind of comments are supported by the JavaScript ?

Ans. : JavaScript supports following comments -

The // is used to specify the single line comment.

The /* and */ can be used to specify the multi-line comments.

The XHTML <!-- -> and <- -> is used in JavaScript.

Q.8 Explain how the string literals are used in JavaScript ?

Ans. : The string is a collection of characters. The string literal is used within the single quote or double quotes.

Q.9 What is the use of the word 'var' in the JavaScript ?

Ans. The word var is used to define the variables of numeric or string type.

Q.10 What do you mean by Coercion ?

Ans. : JavaScript supports the automatic type conversion. The coercion is a type conversion method. This is implicit type of conversion.

Q.11 What is the use of parseInt() and parseFloat() methods ?

Ans. :

The parseInt() and parseFloat() methods are used to separate out the integer and float values respectively from the string. For example parseInt("12abc") will return 12 from the string.

Q.12 What is the use of pop up boxes in JavaScript ?

Ans. : There are three types of popup boxes used in JavaScript. Using these popup boxes the user can interact with the web application.

Q.13 What is the use of toString method with respect to arrays ?

Ans. : The toString method converts the array to string.

Q.14 What are the advantages of indirectly embedding the JavaScript in the Web document ?

Ans. : Following are the advantages of indirectly embedding the JavaScript in the web document -

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

Q.15 What are the properties and methods object window for screen output ?

Ans. : Following are some properties and methods of object window -

1. The property **write** of object window is used to display something on the screen.
2. The methods **alert**, **prompt** and **confirm** are useful for handling screen output and keyboard input.

Q.16 Explain array creation in JavaScript with example.**AU : May-11**

Ans. : In JavaScript the array can be created using Array object. Suppose, we want to create an array of 10 elements then we can write,

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

Using new operator we can allocate the memory dynamically for the arrays. In the brackets the size of an array is mentioned and the var ar denotes the name of the array. Thus by the above sentence an array ar will be created in which we can store 10 elements at the most. Sometimes the above statement can be written like this

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

Q.17 What are global functions in JavaScript ?

Ans. : The global functions are the top level functions in JavaScript that are independent of any specific object. These functions are built in objects of JavaScript. Examples of global functions are –

Name of the Function	Purpose
encodeURIComponent	This function is used to encode URI
decodeURI	This function is used to decode the encoded URI
parseInt	This function parses the string and returns the integer value.
eval	The eval function is used to evaluate the expression.

Q.18 What is the meaning of widgets ?

Ans. : Various graphical controls such as text box, check box, radio buttons, push buttons and so on can be placed on the form for user interactivity. These control objects are called the widgets.

Q.19 State the types of java script statements with examples.**AU : Dec.-13**

Ans. : Various types of javascript statements are -

1. Simple assignment statements
2. Conditional statements
3. Object manipulation statements

4. Comment statements

5. Exception handling statements

Q.20 Write a Java Script to print “Good Day” using IF-ELSE condition.

AU : May-14

Ans. :

```
<html>
<head>
<script>
function MyMessage()
{
    var today=new Date();
    var h=today.getHours();
    if(h<12)//After 12 O'clock Say Good Bye
        //Before 12 O'clock say Good Day
        document.write("Good Day");
    else
        document.write("Good Bye");
}
</script>
</head>
<body onload="MyMessage()">
</body>
</html>
```

Q.21 List any four methods of date object.

AU : Dec.-16

Ans.

Method	Meaning
getTime()	It returns the number of milliseconds. This value is the difference between the current time and the time value from 1st January 1970.
getDate()	Returns the current date based on computers local time.
getDay()	Returns the current day. The day number is from 0 to 6 i.e. from sunday to saturday.
getHours()	Returns the hour value ranging from 0 to 23, based on local time.
getSeconds()	Returns the second value ranging from 0 to 59, based on local time.

Q.22 How exceptions are handled in JavaScript ?**AU : May 17**

Ans. : The exceptions are handled in Javascript using try... catch block.s

For example –

```
<html>
<head>
  <script type="text/javascript">
    <!--
      function myFunc()
      {
        var a = 100;
        var b = 0;
        try{
          if ( b == 0 ){
            throw( "Divide by zero error." );
          }
          else
          {
            var c = a / b;
          }
        }
        catch ( e ) {
          alert("Error: " + e );
        }
      }
    //-->
  </script>
</head>
```

Q.23 What are object literal in JavaScript.**AU : Dec.-18**

Ans. : The object literal in javascript is a name-value pair wrapped in curly braces.

For example -

```
Var myobj = {
    name : "AAA",
    rollno : 2,
    marks : 82
};
```



Unit III

5	Document Object Modeling
----------	---------------------------------

Syllabus

Host Objects : Browsers and the DOM-Introduction to the Document Object Model DOM History and Levels-Intrinsic Event Handling-Modifying Element Style-The Document Tree-DOM Event Handling-Accommodating Noncompliant Browsers Properties of window.

Contents

5.1	<i>Introduction to the Document Object Model</i>	
5.2	<i>DOM History and Levels</i>	
5.3	<i>The Document Tree</i>	May-11,12 Dec.-13, Marks 8
5.4	<i>Modifying Element Style</i>	Dec.-12, 15,..... Marks 16
5.5	<i>Intrinsic Event Handling</i>	Dec-09,12, May-10,11, Marks 8
5.6	<i>DOM2 Event Model</i>	May-08,12,16, Dec.-08, 11,12,13,..... Marks 16
5.7	<i>Accommodating Noncompliant Browsers</i>	
5.8	<i>Properties of Window</i>	May-10, Marks 2
	<i>Two Marks Questions with Answers</i>	

5.1 Introduction to the Document Object Model

- The **Document Object Modeling (DOM)** is for defining the standard for accessing and manipulating HTML, XML and other scripting languages.
- It is the **W3C recommendation** for handling the structured documents.
- Normally the structured information is provided in XML, HTML and many other documents.
- Hence DOM provides the standard set of programming interfaces for working with XML and XHTML and JavaScript.

What is DOM?

Document Object Model (DOM) is a set of platform independent and language neutral Application Programming Interface (API) which describes how to access and manipulate the information stored in XML, HTML and JavaScript documents.

5.2 DOM History and Levels

- Various levels of DOM are enlisted in the following table -

Level	Description
DOM 0	This model is supported by the early browsers. This level could support JavaScript. This version was implemented in Netscape 3.0 and Internet Explorer 3.0 browsers.
DOM 1	This version was issued in 1998 which was focused on XHTML and XML.
DOM 2	This version was issued in 2000 that could specify the style sheet. It also supports the event model and traversal within the documents.
DOM 3	This is the current release of DOM specification published in 2004. This version could deal with XML with DTD and schema, document validations, document views and formatting.

5.3 The Document Tree

AU : May-11,12 Dec.-13, Marks 8

- Basically DOM is an **Application Programming Interface (API)** that defines the interface between HTML document and application program. That means, suppose application program is written in Java and this Java program wants to access the elements of HTML web document then it is possible by using a set of Application Programming Interfaces (API) which belongs to the DOM.
- The DOM contains the **set of interfaces** for the document tree node type. These interfaces are similar to the Java or C++ interfaces. They have **objects, properties and methods** which are useful for respected node type of the web document.

- The documents in DOM are represented using a tree like structure in which every element is represented as a node. Hence the tree structure is also referred as DOM tree.
- For example :** Consider following XHTML document.

```
<html>
<head>
  <title>This is My Web Page </title>
</head>
<body>
  <h1>Hello Friends </h1>
  <h2>How are you?</h2>
  </h3>See you</h3>
</body>
</html>
```

- The DOM tree will be

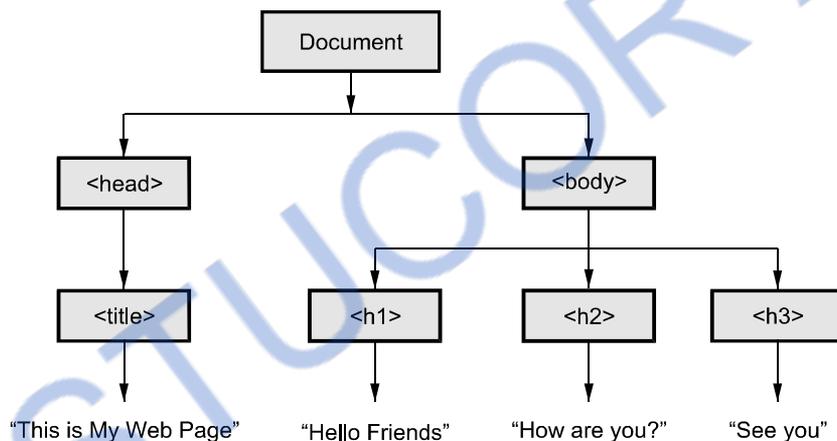


Fig. 5.3.1 DOM structure for simple web document

- We can describe some basic terminologies used in DOM tree as follows -
 - Every element in the DOM tree is called **node**.
 - The topmost single node in the DOM tree is called the **root**.
 - Every child node must have a parent **node**.
 - The bottommost nodes that have no children are called **leaf nodes**.
 - The nodes that have the common parent are called **siblings**.

University Questions

- Explain about document tree in detail.
- List and explain the various types of document nodes

AU : May-11,12 Marks

AU : Dec.-13, Marks 8

5.4 Modifying Element Style**AU : Dec.-12, 15, Marks 16**

We can access or change the contents of document by using various methods. Some commonly used properties and methods of DOM are as follows :

Methods

Method	Meaning
getElementById	This method is used to obtain the specific element which is specified by some id within the script.
createElement	This method is used to create an element node.
createTextNode	Useful for creating a text node.
createAttribute	Useful for creating attribute.
appendChild	For adding a new child to specified node, this method is used.
removeChild	For removing a child node of a specific node, this method is used.
getAttribute	This method is useful for returning the specified attribute value.
setAttribute	This method is useful for setting or changing the specified attribute to the specified value.

Properties

Property	Meaning
attributes	This property is used to get the attribute nodes of the node.
parentNode	This DOM property is useful for obtaining the parent node of the specific node.
childNodes	This DOM property is useful for obtaining the child nodes of the specific node.
innerHTML	It is useful for getting the text value of a node.

5.4.1 Accessing Elements using DOM

- There are several ways by which we can access the elements of the web document.
- To understand these methods of accessing we will consider one simple web document as follows.

```
<html >
<head>
  <title>This is My Web Page </title>
</head>
<body>
<form name="form1" >
  <input type="text" name="myinput"/>
</form>
```

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

Method 1

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

```
var Dom_Obj=document.forms[0].elements[0];
```

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

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

then index reference gets changed. Hence another approach of accessing the elements is developed.

Method 2

- We can access the desired element from the web document using JavaScript method **getElementById**. The element access can be made as follows -

```
var Dom_Obj=document.getElementById("myinput");
```

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

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

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

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

5.4.2 Modifying Elements using DOM

Appending an element Using DOM

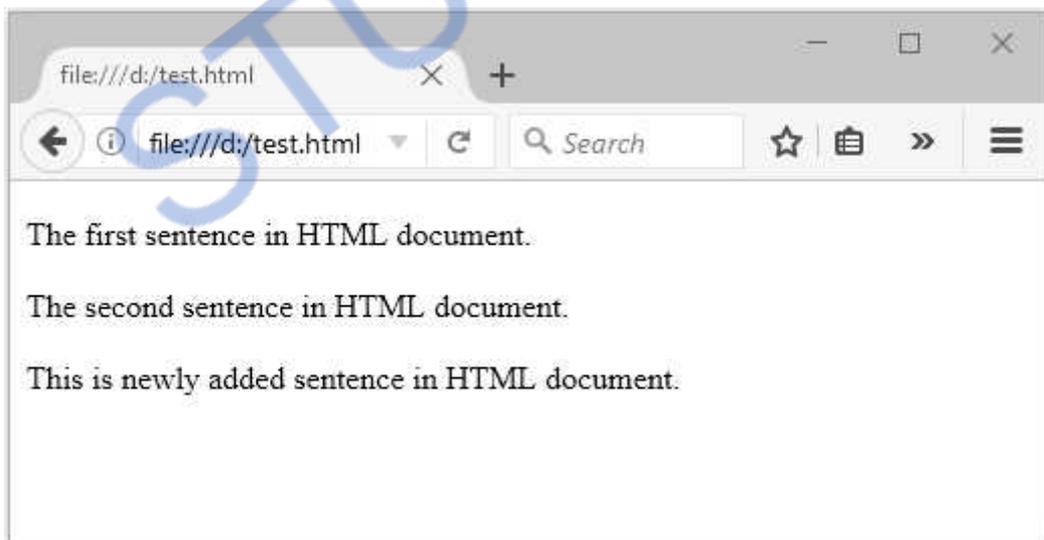
We can add new node in the HTML document using the DOM method **appendChild** method

Following example illustrates the idea

```
<!DOCTYPE html>
<html>
<body>

<div id="div1">
<p id="p1">The first sentence in HTML document.</p>
<p id="p2">The second sentence in HTML document.</p>
</div>

<script>
var pNode = document.createElement("p");
var node = document.createTextNode("This is newly added sentence in HTML document.");
pNode.appendChild(node);
var element = document.getElementById("div1");
element.appendChild(pNode);
</script>
</body>
</html>
```



Inserting an element Using DOM

We can insert a node in between the nodes by using **appendChild** and **insertBefore** method. For example

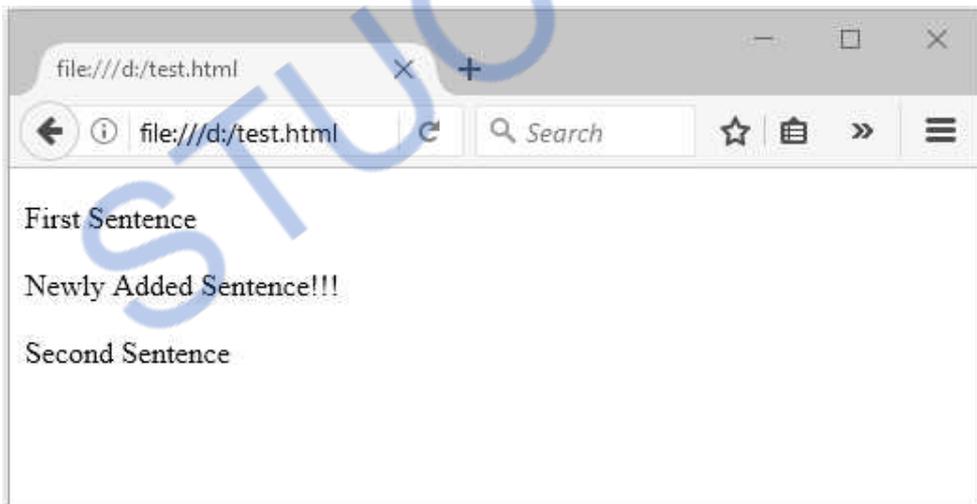
```
<!DOCTYPE html>
<html>
<body>

<div id="div1">
<p id="p1">First Sentence</p>
<p id="p2">Second Sentence</p>
</div>

<script>
var pNode = document.createElement("p");
var node = document.createTextNode("Newly Added Sentence!!!");
pNode.appendChild(node);
var element = document.getElementById("div1");
var nextNode = document.getElementById("p2");
element.insertBefore(pNode,nextNode);
</script>

</body>
</html>
```

Output



Removing an element Using DOM

We can remove or delete particular from HTML document using DOM's **removeChild** method.

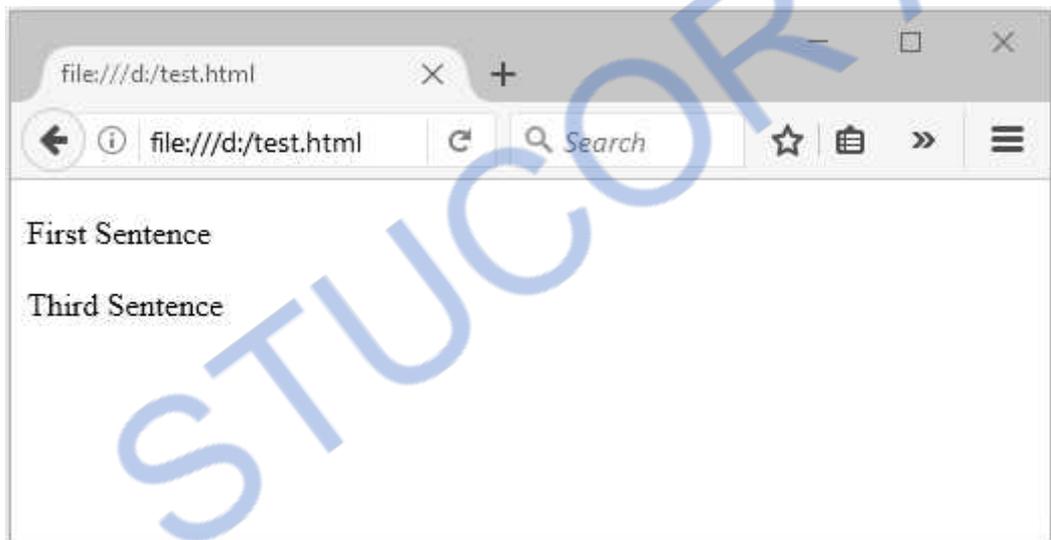
Here is illustration

```
<!DOCTYPE html>
<html>
```

```
<body>

<div id="div1">
<p id="p1">First Sentence</p>
<p id="p2">Second Sentence</p>
<p id="p3">Third Sentence</p>
</div>

<script>
var parentNode = document.getElementById("div1");
var node = document.getElementById("p2");
parentNode.removeChild(node);
</script>
</body>
</html>
```

Output

Ex. 5.4.1 : Write a JavaScript to display sum of two elements. Make use of appropriate DOM method.

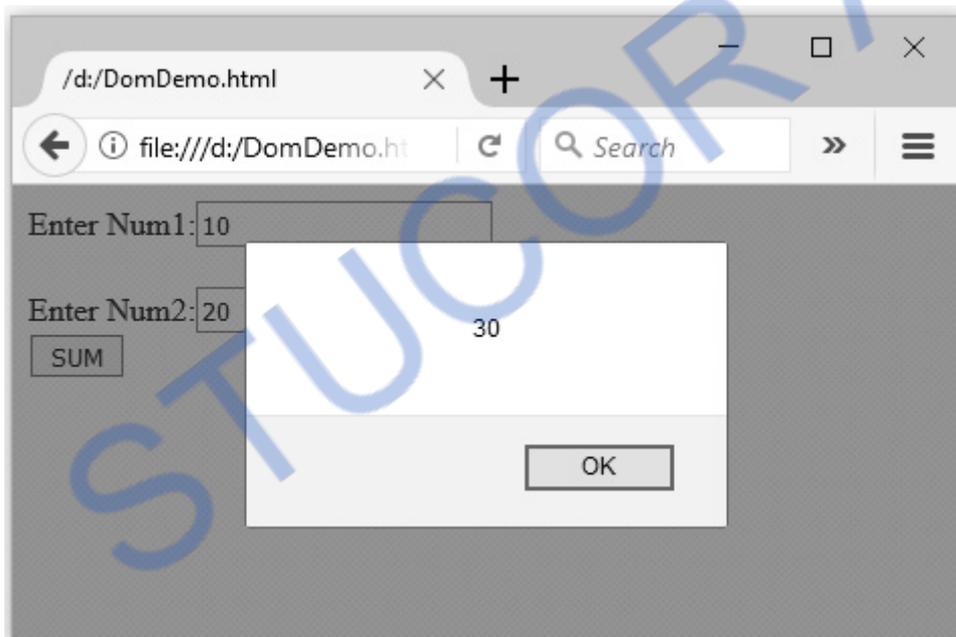
Sol. :

```
<!Doctype html>
<html>
<head>
<script type="text/javascript">
function getSum()
{
    var number1=document.getElementById("number1").value;
    var number2=document.getElementById("number2").value;
    var num1=Number(number1);
```

```

var num2=Number(number2);
alert(num1+num2);
}
</script>
</head>
<body>
<form>
Enter Num1:<input type="text" id="number1" /><br/> <br/>
Enter Num2:<input type="text" id="number2" /><br/>
<input type="button" value="SUM" onclick="getSum()"/>
</form>
</body>
</html>

```

Output

Ex. 5.4.2 : Write a JavaScript to welcome the user by his/her name when he enters his/her name in textbox. Make use of appropriate DOM method.

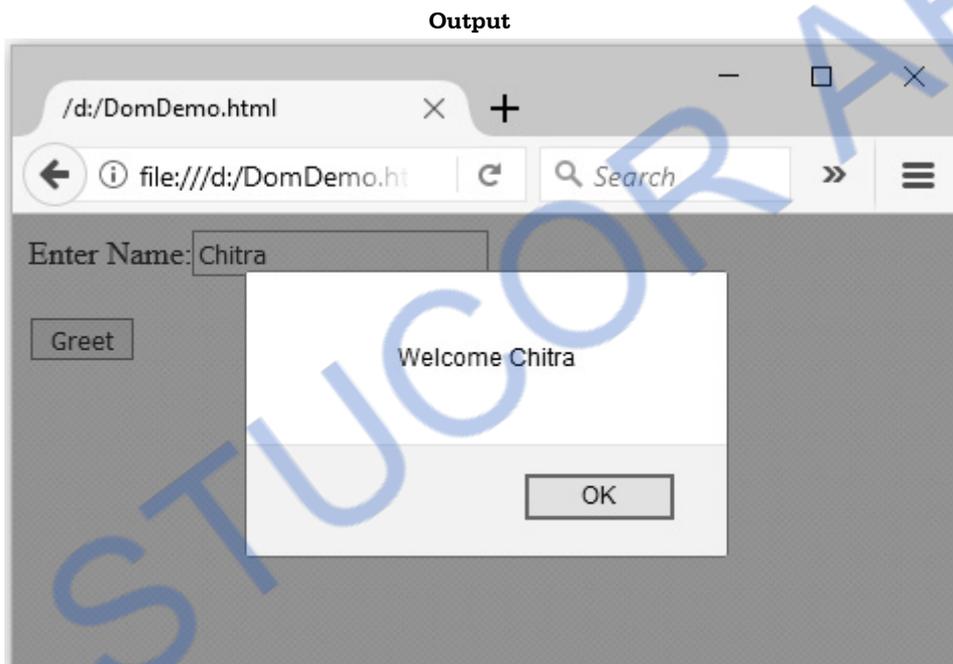
Sol. :

```

<!doctype html>
<html>
<head>
<script type="text/javascript">
function display()
{
var name=document.getElementsByName("user");

```

```
    alert("Welcome "+name[0].value);
}
</script>
</head>
<body>
<form>
Enter Name: <input type="text" name="user" /> <br/> <br/>
<input type="button" value="Greet" onclick="display()"/>
</form>
</body>
</html>
```



University Questions

1. With a simple example illustrate how the elements of the HTML document tree structure can be accessed using Javascript. **AU : Dec.-12, Marks 16**
2. Explain DOM model. **AU : Dec.-15, Marks 8**

5.5 Intrinsic Event Handling

AU : Dec-09,12, May-10,11, Marks 8

- **Definition of Event :** Event is an activity that represents a change in the environment. For example mouse clicks, pressing a particular key of keyboard represent the events. Such events are called intrinsic events.

- **Definition of Event Handler :** Event handler is a script that gets executed in response to these events. Thus event handler enables the web document to respond the user activities through the browser window.
- JavaScript support this special type of programming in which events may occur and these events get responded by executing the event handlers. This type of programming is called event-driven programming.
- Events are specified in lowercase letters and these are case-sensitive.
- **Event Registration :** The process of connecting event handler to an event is called event registration. The event handler registration can be done using two methods -
 - Assigning the tag attributes
 - Assigning the handler address to object properties.

Internet Programming

On occurrence of events the tag attribute must be assigned with some user defined functionalities. This will help to execute certain action on occurrence of particular event.

Commonly used events and tag attributes are enlisted in the following table -

Events	Intrinsic event attribute	Meaning	Associated tags
blur	onblur	Losing the focus.	<button> <input> <a> <textarea> <select>
change	onchange	On occurrence of some change.	<input> <textarea> <select>
click	onclick	When user clicks the mouse button.	<a> <input>
dblclick	ondblclick	When user double clicks the mouse button.	<a> <input> <button>
focus	onfocus	When user acquires the input focus.	<a> <input> <select> <textarea>

keyup	onkeyup	When user releases the key from the keyboard.	Form elements such as input,button,text,textarea and so on.
keydown	onkeydown	When user presses the key down	Form elements such as input,button,text,textarea and so on.
keypress	onkeypress	When user presses the key.	Form elements such as input,button,text,textarea and so on.
mousedown	onmousedown	When user clicks the left mouse button.	Form elements such as input,button,text,textarea and so on.
mouseup	onmouseup	When user releases the left mouse button.	Form elements such as input,button,text,textarea and so on.
mousemove	onmousemove	When user moves the mouse.	Form elements such as input,button,text,textarea and so on.
mouseout	onmouseout	When the user moves the mouse away from some element.	form elements such as input,button,text,textarea and so on.
mouseover	onmouseover	When the user moves the mouse away over some element.	Form elements such as input,button,text,textarea and so on.
load	onload	After getting the document loaded.	<body>
reset	onreset	When the reset button is clicked.	<form>
submit	onsubmit	When the submit button is clicked.	<form>
select	onselect	On selection.	<input> <textarea>
unload	onunload	When user exits the document.	<body>

The use of these tag attributes for handling the events is illustrated by following code sample

```
<input type =“button” name=“My_button”
onclick=“My_fun()”; />
```



Tag attribute Event handler

That means when the user clicks the **button** then as an event handler a user defined function **My_fun()** gets called. Basically in this function user writes the instructions that need to be executed on the button click event.

Ex.5.5.1 : Discuss the properties of mouse events associated with DOM2 with an example.

AU : Dec-12, Marks 8

Sol. : Various types of mouse events associated with DOM2 are click, mousedown, mouseup, mouseout and mouseover.

Events	Intrinsic event attribute	Meaning	Associated tags
mousedown	onmousedown	When user clicks the left mouse button.	Form elements such as input,button,text,textarea and so on.
mouseup	onmouseup	When user releases the left mouse button.	Form elements such as input,button,text,textarea and so on.
mousemove	onmousemove	When user moves the mouse.	Form elements such as input,button,text,textarea and so on.
mouseout	onmouseout	When the user moves the mouse away from some element	form elements such as input,button,text,textarea and so on.
mouseover	onmouseover	When the user moves the mouse away over some element	Form elements such as input,button,text,textarea and so on.
click	onclick	When user clicks the mouse button.	<a> <input>
dblclick	ondblclick	When user double clicks the mouse button.	<a> <input> <button>

Following is an example in which image can be dragged and dropped anywhere in the HTML document. Accordingly the co-ordinates will be displayed.

```
<html>
<head>
<title> Drag and Drop Demo</title>
<script type = "text/javascript">
  var X_offset, Y_offset, Item;
  function Catch_function(e)

{
  Item = e.currentTarget;
  var posX = parseInt(Item.style.left);
  var posY = parseInt(Item.style.top);
  X_offset = e.clientX - posX;
  Y_offset = e.clientY - posY;
  document.addEventListener("mousemove", Move_function, true);
  document.addEventListener("mouseup", Drop_function, true);
  e.stopPropagation();
  e.preventDefault();

}
function Move_function(e)
{
  Item.style.left = (e.clientX - X_offset) + "px";
  Item.style.top = (e.clientY - Y_offset) + "px";
  e.stopPropagation();
}
function Drop_function(e)
{
  document.removeEventListener("mouseup", Drop_function, true);
  document.removeEventListener("mousemove", Move_function, true);
  e.stopPropagation();
}
</script>
</head>
<body>
<h3>
Click on image below-drag it, move it and then drop anywhere!!!
<br /><br />

</h3>
</body>
</html>

```

Output



5.5.1 Handling Events from the Body Elements

To understand how events works in JavaScript let us put some Form components on the JavaScript. The **onload** event gets activated as soon as the web page gets loaded in the browser's window. Following script along with its output illustrate the **onload** tag attribute.

JavaScript[OnloadDemo.html]

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Demo of onload Tag Attibute</title>
<script type="text/javascript">
function my_fun()
{
//This message will be displayed on page loading

```

```

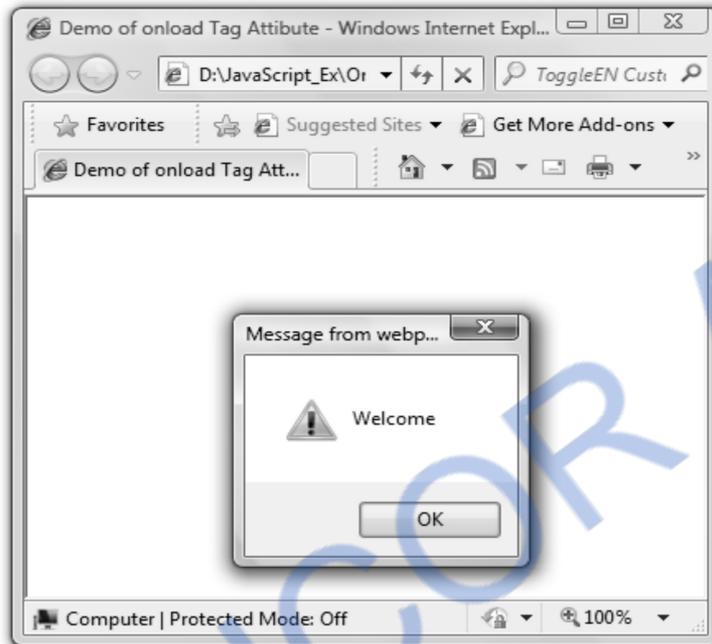
alert("Welcome");
}
</script>
</head>
<body onload="my_fun()">

```

When web document gets loaded on the browser window then **my_fun()** will be called

</body>

</html>

Output**University Questions**

1. Explain the process of event registration.
2. Explain DOM event handling in detail.
3. Describe the event object properties.
4. Explain any four intrinsic event attributes.
5. Write a javascript to demonstrate the onload event.

AU : May-11, Marks 8**AU : May-10, Marks 8****AU : Dec.-09, Marks 6****5.6 DOM2 Event Model****AU : May-08,12,16, Dec.-08, 11,12,13, Marks 16**

The DOM2 event model provides the generic event system in which the events can be registered. The event flow can be represented in a hierarchical manner. Using DOM2 event model information of each event can be obtained.

The DOM2 provides the method for capturing the events so that one can perform specific action in response to the. It also provides an **Event** and **MouseEvent** objects. These are basically the interfaces which contain information specific to a given event. This information can then be used by your event processing code.

Following table enlists the Event Interface and types of various Events that can be triggered

Event interface	Various events
Event	blur,change,abort,error,focus,load,reset,resize,select,scroll,unload,submit
MouseEvent	mousedown, mouseup, mouseover, mousemove, mouseout, click

In DOM0 the connection between event and event handler is simple. But in DOM2 the event model is complicated one.

5.6.1 Event Propagation

- Event propagation means travelling of an event. To understand the event propagation consider one scenario -

Suppose there is an element inside other element. And both these elements have **onchange** event handlers. Then the question occurs here is that which event will be handled first whether the event in the outer element or the event in inner element? Unfortunately there are two opinions about it - the first opinion is the event in the outer element should be handled first and then the event in the inner element should get handled (*event capturing*) (Refer Fig. 5.6.1).

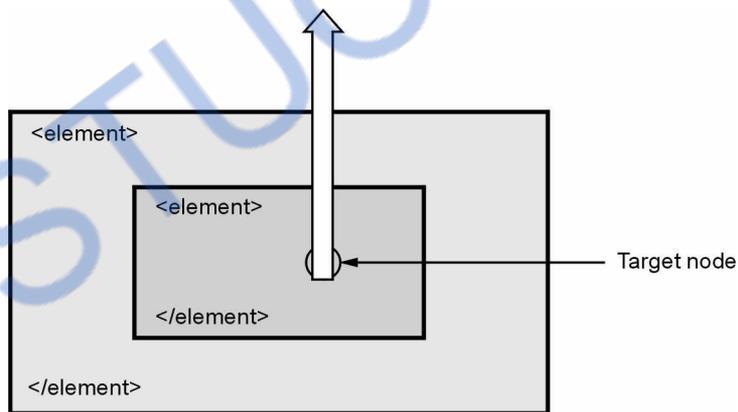


Fig. 5.6.1 Event capturing

According to other opinion, first directly the event which is present in the inner element should be handled and then the event in the outer element must be handled (*event bubbling*). (Refer Fig. 5.6.2).

The DOM0 supports this kind of event handling.

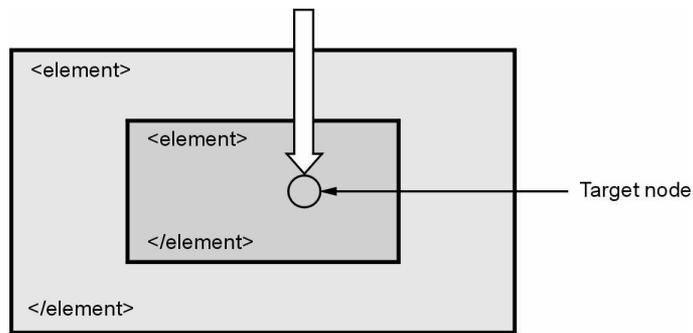


Fig. 5.6.2 Event bubbling

- Event object is created at a node in the document tree. This node is called the **target node**.
- To solve the controversy of event handling whether it is using capturing or using bubbling, the W3C suggested the **DOM2 event model** in which both the approaches are considered.
- According to DOM2 event model the event handling is a three-phase process

Phase 1

This phase is called the **capturing phase** in which the event starts at root node in the document tree and propagates towards the target node. In this direction if any event handler is found(not the target node event handler) then first of all it is checked whether they are enabled or not. If there is any enabled event then it is executed by the capturing phase(may or may not be enabled event). Thus event then reaches to the target node.

Phase 2

In this phase the handlers registered to the target node get executed.

Phase 3

This phase is called the **bubbling phase** in which the event starts at the target node and moves up to the document tree node. In this phase the event bubbles up the document tree to the root node. On this trip if any event handler registered for event gets executed by the bubbling phase(may or may not be enabled event).

- There are some events that can be bubbled or captured. All the mouse events can be bubbled but load and unload events do not get bubbled.
- The **stopPropagation** method of event object is useful for stopping the propagation of an event.
- Events cause certain action on occurrence of particular action. Using **preventDefault** method the default action can be prevented on occurrence of an event.

5.6.2 Event Handler Registration

- The event handler registration in DOM0 is called the traditional method of event registration. In DOM0 there are **two methods** by which the event registration is done -

Method 1 :

- In this method the event handler code can be assigned to the attribute of an element. For example -

```
element.onclick=my_fun;
```

That means when user clicks on the element the event handler function **my_fun()** will be called. Note that we must not use parenthesis while specifying the function **my_fun**.

Method 2 :

- In this method name of the event handler is assigned to the property associated with the event. For example

```
<input type="button" value="Enter" onclick="my_fun();" />
```

Thus the **onclick** becomes the **property** of the button element to which the **event handler code** is assigned. Hence on clicking the button, the code for **my_fun()** will get invoked as the event handler.

- **Drawback** : The drawback of using DOM0 event model for registering the event is that we **cannot assign two different handlers** for the same event. Sometimes we want two different functionalities to get executed for the same type of event. For instance we may want to check the spelling on button click or we may want to edit the code on button click. This can not be achieved in DOM0 event registration model.
- To overcome the drawback of traditional event registration model W3C has introduced the new model called **DOM2 event registration model**. That means at a time we can have more than one event handler functions that can be associated with the same event.
- In this model a method **addEventListener** is called to which the name of the event and event listener is passed as a parameter.
- The **addEventListener** is defined in **EventTarget** interface.

```
addEventListener("name_of_event",handler_function,true or false);
```

The first parameter is the name of the event which must be given in quotes as it is considered to be the string literal. The second parameter is the event handler function which is basically the user defined function for specifying the actions to be taken when the event gets fired. The third parameter is either true or false. If the value is **true** then that means the particular event is enabled for using it in **capturing phase**. If the value is **false** then that means the particular event is enabled for using it in **bubbling phase**.

- For example -

```
document.my_button.addEventListener("click",my_fun,false);
```

When user clicks the button element whose id is **my_button** the event handler code **my_fun** gets invoked. The third parameter **false** indicates that the event is enabled for the bubbling phase. Thus `addEventListener` is useful in registering the event.

- For removing or deleting the event registration the method **removeEventListener** is used. This method takes the same parameters as the **addEventListener**. Hence

```
removeEventListener("name_of_event",handler_function,true or false);
```

- The **this** keyword is used to refer the owner of a function. In event registration function if we use the **this** keyword then it refers to the element the event is handled by

For example -

```
document.my_button.addEventListener("click",my_fun,false);
```

- The event handler function **my_fun** will be

```
function my_fun()
```

```
{
  this.style.background="pink";
}
```

That means if we click the button `my_button` then the background color gets changed to pink.

- There is a **event** property of global window object which provides the event details.

For example

```
<input type="text" value="" id="my_txt" />
```

...

```
Txt_obj=getElementById("my_txt");
Txt_obj.addEventListener("change",my_fun,false);
....
```

```
function my_fun(event)
{
  var My_name=event.currentTarget;
  alert(My_name.value);
}
```

Placing the text box on the form

Registering the event when user types something in the text box

The contents that are typed by the user in the text box gets displayed using alert

- Using the `currentTarget` property of the event object the target node of the event can be accessed.
- The **clientX** and **clientY** given the x and y co-ordinates at the time of event.

5.6.3 Examples

In this section we will make use of DOM2 event model for registering the event. Note that we have used two different functions for the same event.

Note that this feature is not supported by Internet Explorer. Hence we must see the output on some another web browser. I have used Mozilla Firefox web browser.

Ex. 5.6.1 : Write a Javascript for validating name and date.

Sol. : JavaScript [EventReg.html]

```

<!DOCTYPE html>
<html>
<head>
  <title>Demo of Event Registration</title>
<script type="text/javascript">
function chkDt(event)
{
var inputstr =event.currentTarget;
var str=inputstr.value;
if(str.match(/\d{1,2}:\d{1,2}:\d{4}/))
{
  a=str.split(":");
  var dd=Number(a[0]);
  var mm=Number(a[1]);
  var yy=Number(a[2]);
  if(mm==1 | mm==01)
  {
    if(dd>=30)
    {
      alert("Invalid date,Please re-enter it(February have 28 or 29 days)");
      inputstr.focus();
      inputstr.select();
    }
  }
  else
    alert("Valid Date");
}
else if(mm==0 | mm==2 | mm==4 | mm==6 | mm==7 | mm==9 | mm==11)
{
  if(dd>31)
  {
    alert("Invalid date,Please re-enter it");
    inputstr.focus();
    inputstr.select();
  }
}
else

```

Accessing the text box created for entering the name using the **event.currentTarget**

```

    alert("Valid Date");
}
else if(mm==3 | mm==5 | mm==8 | mm==10)
{
    if(dd>30)
    {
        alert("Invalid date,Please re-enter it");
        inputstr.focus();
        inputstr.select();
    }
    else
        alert("Valid Date");
}
}
else
{
    alert("invalid Input format!!,Please try again");
    inputstr.focus();
    inputstr.select();
}
}
function chkName(event)
{
    var inputstr =event.currentTarget;
    var str=inputstr.value;
    if(str.match(/[a-zA-Z]+\s[a-zA-Z]\s[a-zA-Z]+/))
    {
        a=str.split(" ");
        var First_name=a[0];
        var Mid_initial=a[1];
        var Last_name=a[2];
        if(First_name.match(/^[a-zA-Z]{1,15}$/))
        {
            if(Mid_initial.match(/[a-zA-Z]$/))
            {
                if(Last_name.match(/^[a-zA-Z]{1,15}$/))
                alert("Valid name, now enter date");
                else
                alert("InValid Last Name");
            }
        }
        else
            alert("InValid Middle name initial");
    }
}

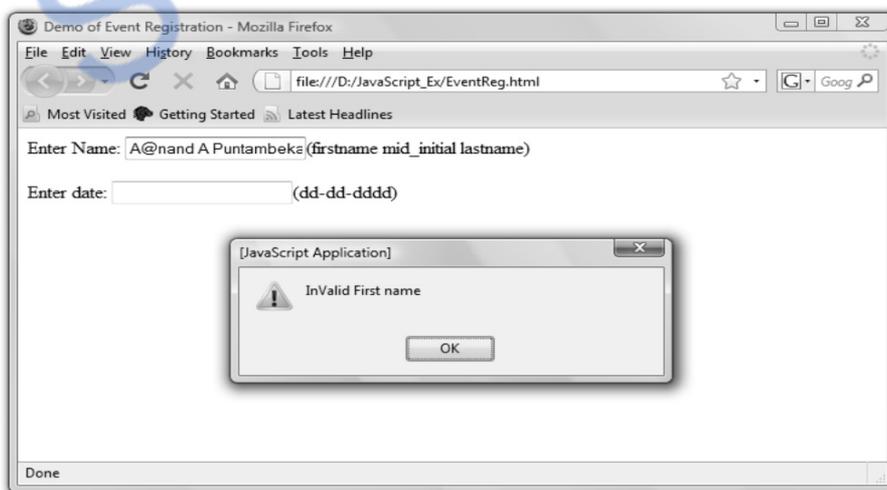
```

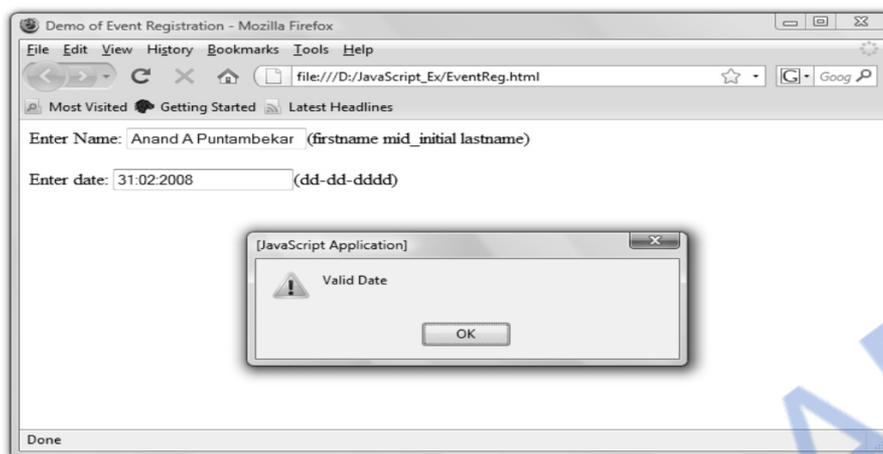
Accessing the text box created for entering the name using the **event.currentTarget**

```
else
  alert("Invalid First name");
}
else
  alert("You have entered wrong input");
}
</script>
</head>
<body>
  <form id="form1">
    <label>Enter Name:
      <input type="text" value="" id="Name_chk" />(firstname mid_initial lastname)
    </label>
    <br/><br/>
    <label>Enter date:
      <input type="text" value="" id="Dt_chk" />(dd-dd-dddd)
    </label>
    <br/>
  </form>
  <script type="text/javascript">
    var name_node=document.getElementById("Name_chk");
    var dt_node=document.getElementById("Dt_chk");
    dt_node.addEventListener("change",chkDt,false);
    name_node.addEventListener("change",chkName,false);
  </script>
</body>
</html>
```

Getting the object for each element and then registering the event handlers

Output





Thus document object model is useful in handling objects and events.

Ex. 5.6.2 : Create a script that repeatedly flashes an image on the screen. Do this by changing the visibility of the image. Allow users to control the “blink speed”.

AU : May-08, Marks 8

Sol. : Blinking.html

```
<html>
<head>
<script type="text/javascript">
function GetSpeed()
{
var input_str=prompt("Enter the speed value here:","");
var i=Number(input_str);
blinkImage(i);
}
function blinkImage(i)
{

if(!document.getElementById('blink').style.visibility)
{
document.getElementById('blink').style.visibility="visible"
}
if(document.getElementById('blink').style.visibility=="visible")
{
document.getElementById('blink').style.visibility="hidden";
}
else
{
document.getElementById('blink').style.visibility="visible";
}
}
```

```

j=i;
timer=setTimeout("blinkImage(j)",i);
}
function stoptimer()
{
clearTimeout(timer);
}
</script>
<title>Image Blinking Demo</title>
</head>
<body onload="GetSpeed()">

</body>
</html>

```

Ex. 5.6.3 : Develop a DHTML page to change the background color using mouse over event on three squares containing different colors.

AU : Dec.-08, Marks 2

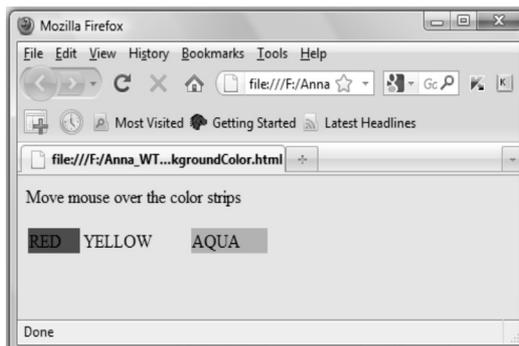
Sol. : backgroundcolor.html

```

<html>
<head>
<script type="text/javascript">
function changeColor(color)
{
document.getElementById('B').style.background=color;
}
</script>
</head>
<body id="B">
<p>Move mouse over the color strips</p>
<p> </p>
<table width="50%">
<tr>
<td bgcolor="red" onmouseover="changeColor('red')"
onmouseout="changeColor('transparent')">RED</td>
<td bgcolor="yellow" onmouseover="changeColor('yellow')"
onmouseout="changeColor('transparent')">YELLOW</td>
<td bgcolor="aqua" onmouseover="changeColor('aqua')"
onmouseout="changeColor('transparent')">AQUA</td>
</tr>
</table>
</body>
</html>

```

Output



Ex. 5.6.4 : Write a DHTML to shake the window of a button click. Make use of two buttons one button used to start the shaking and another button used to stop the shaking of window.

AU : Dec.-08, Marks 10

Sol. : shake button.html

```

<head>
<script language="JavaScript1.2">
function shake(n)
{
if(parent.moveBy)
{
for (i = 100; i > 0; i--)
{
for (j = n; j > 0; j--)
{
parent.moveBy(0,i);
parent.moveBy(i,0);
parent.moveBy(0,-i);
parent.moveBy(-i,0);
}
}
}
}
function Stop_Shake(obj)
{
parent.moveBy=0;
obj.style.left=0;
obj.style.top=0;
}
</script>
</head>
<body>
<center>
<form>
<input type="button" onClick="Stop_Shake(this)" value="Stop">
<input type="button" onClick="shake(100)" value="Shake">

```

```
</form>
</center>
</ form>

</body>
```

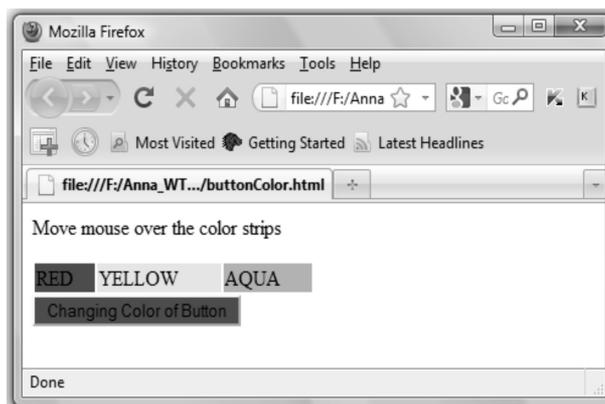
Ex.5.6.5 : Write a DHTML to change the background colour of a button , Mouse over three colored table cells and the background color will change.

AU : Dec.-08, Marks 10

Sol. : buttoncolor.html

```
<html>
<head>
<script type="text/javascript">
function changeColor(color)
{
document.getElementById('MyButton').style.background=color;
}
</script>
</head>
<body>
<p>Move mouse over the color strips</p>
<p> </p>
<table width="50%">
<tr>
<td bgcolor="red" onmouseover="changeColor('red')">RED</td>
<td bgcolor="yellow" onmouseover="changeColor('yellow')">YELLOW</td>
<td bgcolor="aqua" onmouseover="changeColor('aqua')">AQUA</td>
</tr>
</table>
<form>
<input id="MyButton" type="button" value="Changing Color of Button">
</form>
</body>
</html>
```

Output



Ex.5.6.6 : Write a JavaScript for displaying the context menu. The context menu is one that is shown when user right clicks anywhere in the document.

Sol. :

test1.html

```
<div id="ContxtMenu"
style="width:150px;border:1px solid black;background
color:yellow;visibility:hidden;position:absolute;line-height:30px; padding-left: 10px">
  <a href="http://www.google.com">Search Engine</a><br />
  <a href="http://www.facebook.com">Social Networking</a><br />
  <a href="http://www.gmail.com">Email Service</a><br />
</div>
```

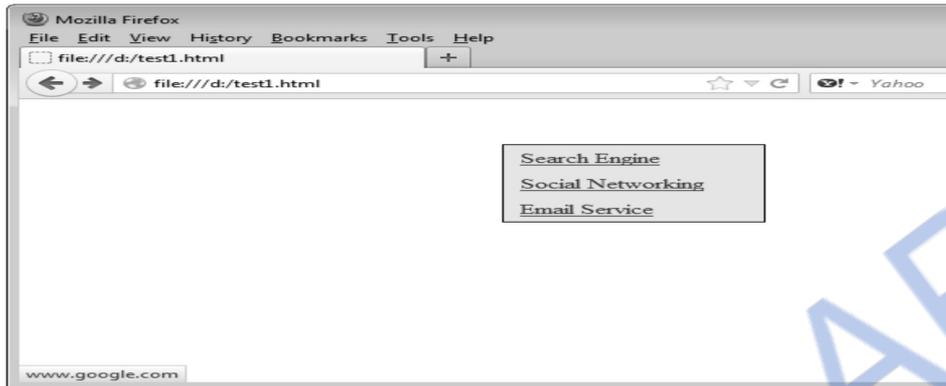
```
<script type="text/javascript">
var browserObj=document.getElementById && !document.all
var visibility=0
```

```
function Display(e)
{
  el=document.getElementById("ContxtMenu")
  visibility=1
  if (browserObj)
  {
    el.style.left=pageXOffset+e.clientX+"px"
    el.style.top=pageYOffset+e.clientY+"px"
    el.style.visibility="visible"
    e.preventDefault()
    return false
  }
}
```

```
function Hide()
{
  if (typeof el!="undefined" && visibility)
  {
    el.style.visibility="hidden"
    visibility=0
  }
}
if (browserObj)
{
  document.addEventListener("contextmenu", Display, true)
  document.addEventListener("click", Hide, true)
}
</script>
```

Output

[Note that the Mozilla FireFox browser is used to get the following output]



Ex.5.6.7 : Use Javascript and HTML to create a page with two panes. The first pane (on the left) should have a text area where HTML code can be typed by the user. The pane on the right side should display the preview of the HTML code typed by the user, as it would be seen on the browser. **AU : May-16, Marks 16**

Sol. :

Step 1 : Create the Main Window containing two frames. The code for this document is as follows -

MainWindow.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Frameset Example</title>
</head>
<frameset cols="50%,50%">
  <frame src="Left.html" name="left"/>
  <frame src="Right.html" name="right"/>
</frameset>
<body>
</body>
</html>
```

Step 2 : Create the left frame containing the textarea in which we can write any HTML code. This document is named as Left.html. It is as follows

Left.html

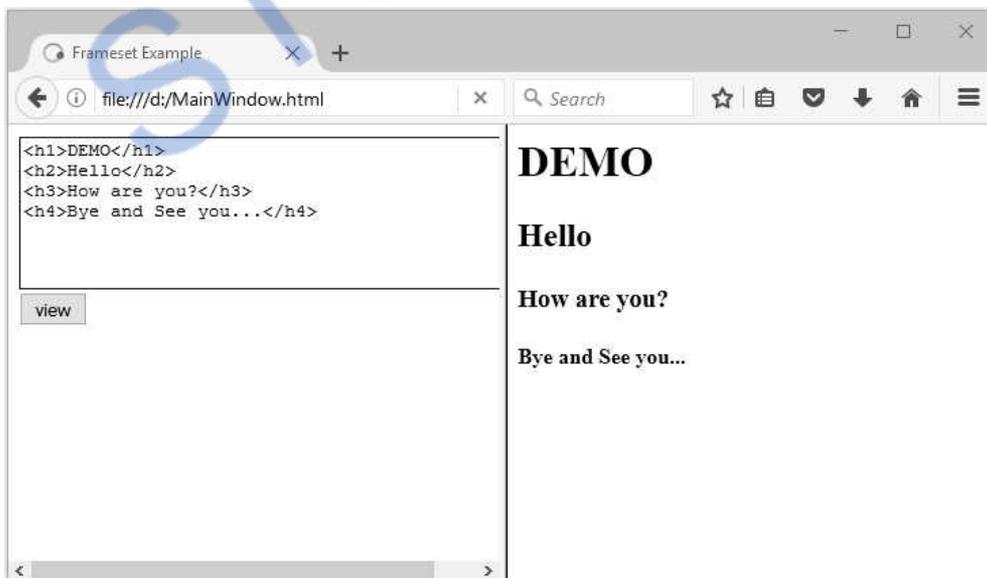
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<form>
```

```
<textarea name="data" cols=45 rows=6></textarea>
  <br>
  <input type="button" value=" view "
onclick="javascript:parent.right.displayHTML(this.form)">
</form>
</body>
</html>
```

Step 3 :

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function displayHTML(form)
{
  var inf = form.data.value;
win = window.open("", "", 'right', 'toolbar = no, status = no');
  win.document.write("" + inf + "");
}
</script>
</head>
<body>
</body>
</html>
```

Step 4 : Open Web browser and open the MainWindow.html document(created in step1). The sample output will be as follows -



University Questions

1. Write a short note on : DOM2 event model.
2. Explain in detail DOM event handling. Also explain with an example of creating a context menu. [Note : A context menu is one that is shown when the user right clicks anywhere in the document.] **AU : Dec.-11, Marks 16**
3. Explain DOM event handling in detail. **AU : May-12, Marks 8**
4. Explain the terms event capturing and event bubbling.
5. With a suitable example discuss about event propagation. **AU : Dec.-12, Marks 8**
6. Explain the properties of the event object. **AU : May-08, Marks 8**
7. What is the drawback of DOM0 event model ?
8. Explain in detail about event object and event listeners with an example. **AU : Dec.-13, Marks 8**

5.7 Accommodating Noncompliant Browsers

- The Mozilla 1.4 and many other browsers support all the features of DOM2.
- However, there are many such browsers that do not have a support for DOM2.
- **For example :** Internet Explorer 6 does not have the support for the DOM2 features. If we use DOM API in some JavaScript and if we execute this script in a non supporting browser then some unusual situations may occur. Hence it is necessary that the JavaScript code must check for the support of DOM2 features.

5.7.1 Detecting Host Objects

- If DOM2 Node object is not present then a **node type constant can be created** by writing following piece of code in the JavaScript.

```
if(!window.Node)
{
    var Node={Element_Node:1};
}
```

- Another way of testing the DOM feature is using the **setProperty()** method of a **style** object

For example

```
if(element.style.setProperty)
{
    //throw an exception if element is not defined
    //throw an exception if style has a null value
}
```

- If an exception is thrown then it indicates that the corresponding browser does not support the DOM feature. Writing such piece of code in JavaScript avoids crashing of the program.

- There is one more popular approach adopted to identify the support for DOM feature. In this approach the code is written to **identify the host browser** itself. If the code is executing on internet explorer then using the **setProperty** method the property value pair is set appropriately.

5.8 Properties of Window

AU : May-10, Marks 2

Various commonly used properties **window** method are -

Method	Description
alert(String)	It displays the alert box with some message and OK button
confirm(String)	It displays the alert box with some message and OK button and Cancel button.
prompt(String)	It displays a dialog box which allows the user to input some data.
open()	<p>This method opens a new window with specified URL. If no URL is specified then a blank window is opened up. The second parameter is for name of the window. However we can pass some values as a second parameter. These values are -</p> <ul style="list-style-type: none"> _blank – URL is loaded in new window. _parent – URL is loaded in parent window _self – URL replaces currently opened window _top – URL replaces the currently opened frameset. <p>The second parameter is for specifying the width and height of the window.</p>
close()	This method is used to close the current window.
moveBy()	This method helps in moving the window from the current position to some other position.
moveTo()	This method moves the window to specific position.
resizeBy()	This method resizes the window by the specified pixels.
resizeTo()	The window is resized to the specified width and height.

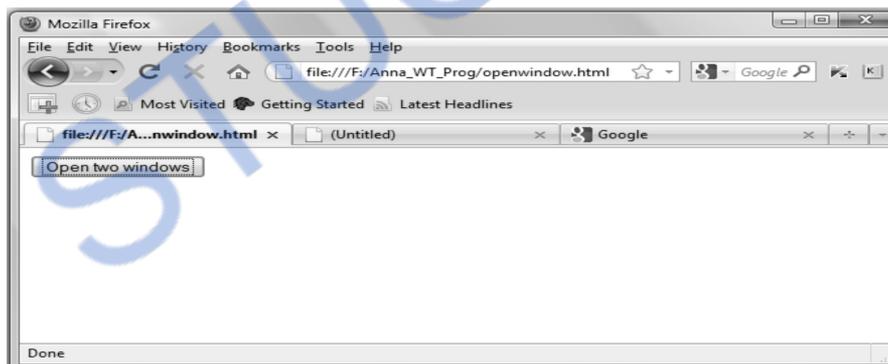
Following are some examples that make use of above methods.

Ex.5.8.1 : Write a JavaScript for opening two windows - one a blank window and second one for www.google.com.

Sol. : JavaScript code[[openwindow.html](#)]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script type="text/javascript">
function fun()
{
window.open("");
window.open("http://www.google.com");
}
</script>
</head>
<body>
<form>
<input type="button" value="Open two windows" onclick="fun()">
</form>
</body>
</html>
```

Output



Ex.5.8.2 : Write a JavaScript for closing a specific window.

Sol. : JavaScript Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script type="text/javascript">
function fun()
{
```

```

window.open("");
window_obj=window.open("http://www.google.com");
}
function fun1()
{
  window_obj.close();
}
</script>
</head>
<body>
<form>
  <input type=button value="Open two windows" onclick="fun()">
  <input type=button value="close" onclick="fun1()">
</form>
</body>
</html>

```

Universtiy Question

1. Name the methods and properties of window object in JavaScript.

AU : May-10, Marks 2

Two Marks Questions with Answers

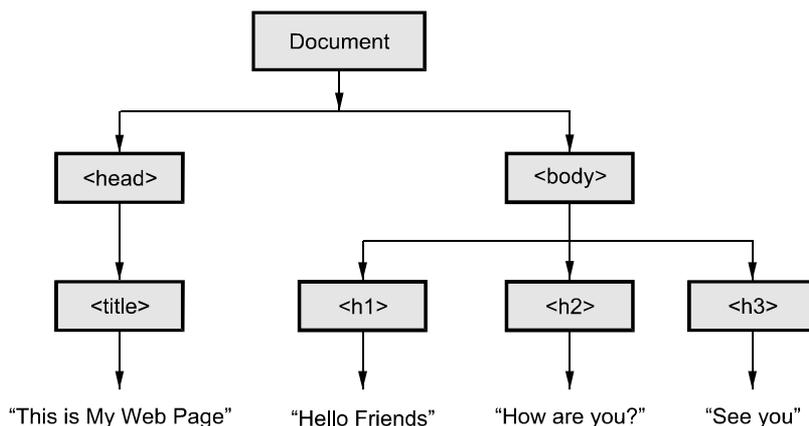
Q.1 What is DOM ?

Ans. : Document Object Model (DOM) is a set of platform independent and language neutral application programming interface (API) which describes how to access and manipulate the information stored in XML, XHTML and JavaScript documents.

Q.2 Define DOM Tree.

Ans. : The documents in DOM are represented using tree like structure in which every element is represented as a node. Such a tree like structure is called DOM Tree.

For example -



Q.3 List the different methods defined in document and window object of JavaScript.

AU : May-11, 13

Ans. : Methods in document object

Methods	Description
document.createAttribute()	Creates an attribute node.
document.createElement()	Creates an element node .
document.createTextNode()	Creates a text node.
document.getElementById()	Returns the element that has ID attribute with the specified value.
document.getElementsByName()	Returns the element specified by its name.
document.write()	Writes HTML expression to a document.

Methods in window object

Methods	Description
alert()	Displays the alert box containing some message and OK button.
confirm()	Displays a dialog box with a message and OK and Cancel button.
createPopup()	Creates a pop up window.
moveTo()	Moves a window to some specific position.
open()	Opens a new window.
print()	Prints the contents in the window.

Q.4 What is the use of 'all' in DOM tree traversal ?

Ans. : The all is a object model collection which is used to refer all the HTML elements. For representing all the elements present on the HTML document the 'all' is used.

Q.5 List out various level of Document Object Modelling.

Ans. : Various levels of DOM are -

DOM0 DOM1 DOM2 DOM3

Q.6 What is getElementById method ?

Ans. : The getElementById is a method which is defined in DOM1. The element access can be made by using this method. For example -

```
var Dom_Obj=document.getElementById("myinput");
```

Q.7 What is the drawback of the DOM0 event Model ?

Ans. : The main drawback of DOM0 event model is that two different event handlers can not be assigned to the same event. This drawback is overcome in DOM2 event model.

Q.8 What do you mean by the term event registration ?

Ans. : The process of connecting event handler to an event is called event registration. The event handler registration can be done using two methods -

- Assigning the tag attributes
- Assigning the handler address to object properties.

Q.9 Define the term intrinsic events.**AU : May-12**

Ans. : Event is an activity that represents a change in the environment. For example mouse clicks, pressing a particular key of keyboard represents the events. Such events are called intrinsic events.

Q.10 List some HTML intrinsic attributes.**AU : May-11, 12**

Ans. : Some of the intrinsic attributes are listed in the following table

Intrinsic Attribute	Meaning
Onblur	This event is for losing the focus
onchange	On occurrence of some change this event occurs
onclick	When user clicks the mouse button this event occurs
ondblclick	When user double clicks the mouse button.

Q.11 Define event bubbling.**AU : May-10**

Ans. : Suppose, there is an element present inside another element. Then during the event handling, if the event which is present in the inner element is handled and then the event of the outer element is handled. This process of event handling is called event bubbling.

Q.12 List the types of event listeners in DOM2.**AU : Dec.-12**

Ans. : There are three types of event listeners in DOM2 -

1. **Capturing listeners :** The capturing listener is associated with the event that occurs at the root node in the document tree and propagates towards the target node. It is created with the call to `addEventListener()`.
2. **Bubbling listeners :** The bubbling listener is associated with the event that starts at the target node and moves up the document tree to the root node. It is created with the call to `addEventListener()`.
3. **Target listeners :** Target listener is the event listener which is added in the target node.

Q.13 What is meant by DHTML ?**AU : Dec.-13**

Ans. : The DHTML stands for Dynamic HTML. This is a markup language used to create interactive and animated web site.

Q.14 Define event programming. Name any two of its techniques.**AU : Dec.-15**

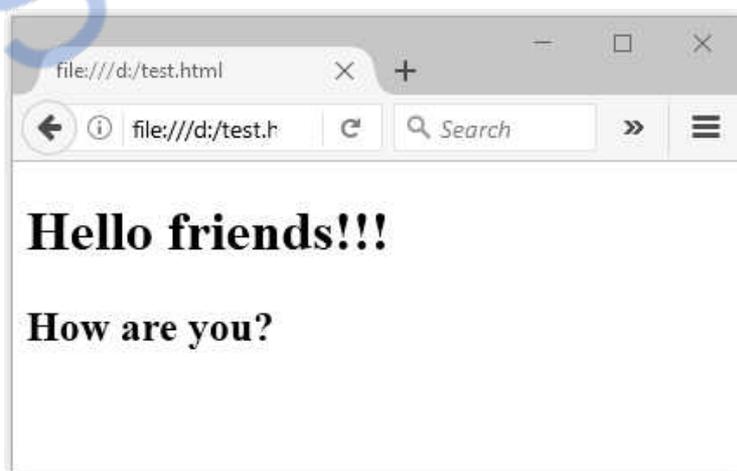
Ans. : • Event is an activity that represents a change in the environment. For example mouse clicks, pressing a particular key of keyboard represent the events. Event programming is technique in which such events are handled using the code.

- Event capturing and event bubbling are the two techniques of event propagation.

Q.15 Write appropriate JavaScript code to remove an element(current element) from a DOM.**AU : May-16**

Ans. :

```
<!DOCTYPE html>
<html>
<body>
  <div id="headers">
<h1 id="s1">Hello friends!!!</h1>
<h2 id="s2">How are you?</h2>
<h3 id="s3"> See you...</h3>
</div>
<script>
  var parent = document.getElementById("headers");
  var child = document.getElementById("s3");
  parent.removeChild(child);
</script>
</body>
</html>
```

Output

Notes

STUCOR APP

Unit III

6	Server Side Scripting
----------	------------------------------

Syllabus

Server-Side Programming: Java Servlets- Architecture -Overview-A Servlet-Generating Dynamic Content-Life Cycle- Parameter Data-Sessions-Cookies-URL Rewriting-Other Capabilities-Data Storage Servlets and Concurrency- Databases and Java Servlets.

Contents

6.1	Introduction to Server Side Programming	
6.2	Java Servlets	
6.3	Architecture	May-08, 13, Marks 8
6.4	Life Cycle	
6.5	Servlet API	Dec.-09, May-10 Marks 8
6.6	Parameter Data	May-08, 14, Dec.-09, Marks 8
6.7	Session Management Technique	
6.8	Session Management using Session ID	May-12,14, Marks 8
6.9	Cookies	May-09, 13, 17, Dec.-16, Marks 16
6.10	URL Rewriting	Dec.-18,..... Marks 15
6.11	Data Storage	
6.12	Servlet and Concurrency	
6.13	Database and Java Servlets	Dec.-09,12, May-13,16,17, . Marks 16
	Two Marks Questions with Answers	

6.1 Introduction to Server Side Programming

- Server side technology deals with the data submitted by the client.
- Server side program basically **handles the request** made by the client. This program then **prepares the response** for the client.
- The server side programming also act as a link between backend database and front end client side programming.
- Various server side technologies include – servlet, JSP, ASP.NET, PHP and so on.

Difference between client side and server side scripting

Sr. No.	Server side scripting	Client side scripting
1.	The server side scripting is used to create the web pages that provide some services.	The client side scripting is used to create the web pages as a request or response to server. These pages are displayed to the user on web browser.
2.	These scripts generally run on web servers.	These scripts generally run on web browser.
3.	A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser.	The processing of these scripts takes place on the end users computer. The source code is transferred from the web server to the users computer over the internet and run directly in the browser.
4.	Uses : Processing of user request, accessing to databases.	Uses : Making interactive web pages, for interacting with temporary storages such as cookies or local storage, sending request to server and getting the response and displaying that response in web browser.
5.	Example : PHP, ASP.NET, nearly all the programming languages including C++, Java and C#	Example : HTML, CSS, JavaScript(primarily)

6.2 Java Servlets

- Servlets are basically the Java programs that run on server. These are the programs that are requested by the XHTML documents and are displayed in the browser window as a response to the request.
- The servlet class is instantiated when web server begins the execution.
- The execution of servlet is managed by **servlet container**, such as Tomcat.

- The servlet container is used in java for dynamically generate the web pages on the server side. Therefore the servlet container is the part of a web server that interacts with the servlet for handling the dynamic web pages from the client.
- Servlets are most commonly used with HTTP (i.e. Hyper Text Transfer Protocol) hence sometimes servlets are also called as "HTTP Servlet".
- The main purpose of servlets is to add up the functionality to a web server.

Working of How Servlet Works ?

- Before learning the actual servlet programming it is very important to understand how servlet works.

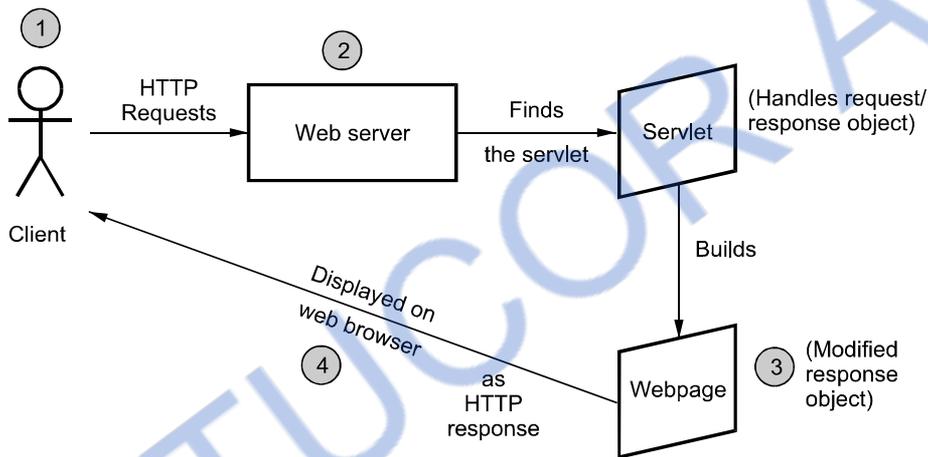


Fig. 6.2.1 How servlet works ?

1. When a client make a request for some servlet, he/she actually uses the **Web browser** in which request is written as a URL.
2. The web browser then sends this request to **Web server**. The web server first finds the requested servlet.
3. The obtained servlet gathers the relevant information in order to satisfy the client's request and builds a **web page** accordingly.
4. This web page is then **displayed** to the client. Thus the request made by the client gets satisfied by the servlets.

6.2.1 Need and Advantages

- Servlets can process and store the data submitted by an HTML form.
- Servlets are useful for providing the dynamic contents. For example retrieving and updating the databases.

- Servlets can be used in the cookies. Cookies are small programs which can make use of the information submitted on currently accessed web pages.
- Similarly servlets are used in session tracking. The session tracking are the useful programs that keeps track of all previously accessed the web pages.

Advantages

- The servlets are very **efficient** in their performance and get executed in the address space of the belonging web server.
- The servlets are **platform independent** and can be executed on different web servers.
- The servlets working is based on **Request-Response**. Any HTML form can take the user input and can forward this input to the servlet. The servlets are then responsible to communicate with the back-end database and manipulate the required business logic. These servlets embedded on the web servers using **Servlets API**.
- Servlets provide a way to generate the **dynamic document**. For instance : A servlet can display the information of current user logged in, his logging time, his last access, total number of access he made so far and so on.
- **Multiple users** can keep a co-ordination for some application among themselves using servlets.
- Using servlets **multiple requests** can be synchronized and then can be concurrently handled.

6.3 Architecture

AU : May-08, 13

- When we write a servlet program, it is necessary to - i) either implement **Servlet** interface or ii) extend a class that implements **Servlet** interface.
- While implementing **Servlet** interface we must include **javax.servlet** package. Hence the first line in our servlet program must be


```
import javax.servlet.*;
```
- **GenericServlet** class is a predefined implementation of **Servlet** interface. Hence we can extend **GenericServlet** class in our servlet program. Similarly, the **HttpServlet** class is a child class of **GenericServlet** class, hence we can extend this class as well while writing the servlet program.
- Hence following are the two ways by which we can write servlet program

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Test extends
GenericServlet
{
    //body of servlet
}
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Test extends HttpServlet
{
    //body of servlet
}
```

- The servlet gets the request from the client for some service. The servlet then processes the request and sends the response back to the client. In order to handle these issues **HttpServletRequest** and **HttpServletResponse** are used in servlet program.
- These requests are handled with the help of some methods that are popularly known as **methods of HttpServlet**. These methods are as follows

Method	Purpose
doGet	This method handles the HTTP GET request
doPost	This method handles the HTTP POST request
doPut	This method handles the HTTP Put request.
doDelete	This method handles the DELETE request.

The doGet and doPost methods

- The **doGet** method requests the data from the source.
- The **doPost** method submits the processed data to the source.
- The protocol of **doGet** method is as follows

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException,IOException
```
- The **ServletException** and **IOException** are thrown to handle the Servlet problems gracefully.
- The **HttpServletRequest** request : contain the client request made by client.
- The **HttpServletResponse** response : contains the response made by servlet back to the client.
- The protocol of **doPost** method is same as **doGet** method. It is as follows -

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
```
- The GET request is more efficient than the POST request.
- The GET request is less secure than the POST request.

How to Write Servlet Program ?

Open Notepad and write the first servlet code to display Greeting messages. It is as follows

FirstServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends HttpServlet
{
```

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException
{
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>My First Servlet</title>");
    out.println("<body>");
    out.println("<h1>Hello How are U?</h1>");
    out.println("<h2>I am enjoying this Servlet Application</h2>");
    out.println("<h3>See You later!</h3>");
    out.println("</body>");
    out.println("</html>");
}
}
```

Program Explanation :

- In the above program, we have imported following files,
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
- Out of these files **java.io** package is useful for taking care of I/O operations.
- The **javax.servlet** and **javax.servlet.http** are important packages containing the **classes** and **interfaces** that are required for the operation of servlets. The most commonly used interface from **javax.servlet** package is **Servlet**. Similarly most commonly used class in this package is **GenericServlet**. The **ServletRequest** and **ServletResponse** are another two commonly used interfaces defined in **javax.servlet** package.
- In the **javax.servlet.http** package **HttpServletRequest** and **HttpServletResponse** are two commonly used interfaces. The **HttpServletRequest** enables the servlet to read data from the HTTP request and **HttpServletResponse** enables the servlet to write data to HTTP response. The **cookie** and **HttpServlet** are two commonly used classes that are defined in this package.
- We have given class name **FirstServlet** which should be derived from the class **HttpServlet**. (Sometimes we can derive our class from **GenericServlet**).
- Then we have defined **doGet method** to which the HTTP request and response are passed as parameters. The commonly used basic exceptions for the servlets are **IOException** and **ServletException**.

- The MIME type is specified as using the **setContentType()** method. This method sets the content type for the HTTP response to type. In this method “text/html” is specified as the MIME type. This means that the browser should interpret the contents as the HTML source code.
- Then an output stream is created using **PrintWriter()**. The **getWriter()** method is used for obtaining the output stream. Anything written to this stream is sent to the client as a response. Hence using the object of output stream ‘out’, we can write the HTML source code in **println** method as HTTP response.

How to execute Servlet program ?

Step 1 : Compile the above program using the javac command at command prompt.

```
D:\test>javac FirstServlet.java
```

The class file for this program gets generated.

Step 2 : Before running any servlet program, it is necessary to have

1. JDK installed
2. Tomcat installed.
3. Class path for above two packages must be set.

For Tomcat installation, I prefer to install the package XAMPP. The XAMP contains a directory for tomcat. The XAMPP package contains Apache Web server, MySQL, PHP and Perl support. It can work on both Windows and Linux operating System.

Step 3 : Copy the class file generated in Step 1 to the path

```
C : \xampp\tomcat\webapps\examples\WEB-INF\classes
```

Step 4 : Go to the directory

```
C : \xampp\tomcat\webapps\examples\WEB-INF
```

Open **web.xml** file and edit it as follows

```
<servlet>
  <servlet-name>FirstServlet</servlet-name>
  <servlet-class> FirstServlet </servlet-class>
</servlet>
...
...
...
<servlet-mapping>
  <servlet-name> FirstServlet </servlet-name>
  <url-pattern>/servlet/ FirstServlet </url-pattern>
</servlet-mapping>
```

The **web.xml** file is popularly known as **deployment descriptor**. This is basically the configuration file that describes how to map the URL of web application to servlet.

```

<servlet>
    Internal alias Name
    <servlet-name>FirstServlet</servlet-name>
    Actual Name of server
    <servlet-class> FirstServlet </servlet-class>
</servlet>
...
...
...
    Internal alias Name
<servlet-mapping>
    <servlet-name> FirstServlet </servlet-name>
    External alias Name
    <url-pattern>/servlet/ FirstServlet </url-pattern>
</servlet-mapping>

```

The Servlet comes with two alias names, internal and external. The internal name is used by the Tomcat and the external name is given (to be written in <FORM> tag of HTML file) to the client to invoke the Servlet on the server. That is, there exists alias to alias. All this is for security. Observe, the names are given in two different XML tags, in the web.xml file, to make it difficult for hacking.

To invoke the **FirstServlet** Servlet, the client calls the server with the name **servlet/FirstServlet**.

When **servlet/FirstServlet** call reaches the server, the Tomcat server opens the **web.xml** file to check the deployment particulars. Searches such a <servlet-mapping> tag that matches **servlet/FirstServlet**. **servlet/FirstServlet** is exchanged with **FirstServlet**.

Then, searches such a <servlet> tag that matches **FirstServlet** and exchanges with **FirstServlet class**. Now the server, loads **FirstServlet** Servlet, executes and sends the output of execution as response to client.

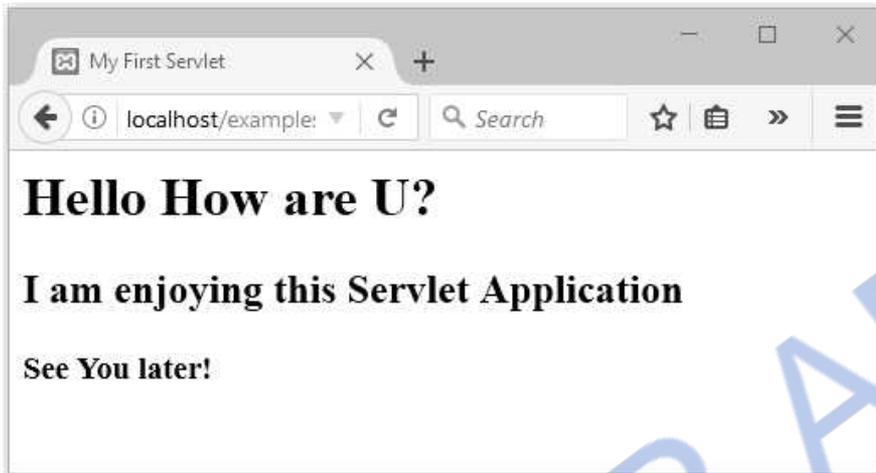
Step 5 : Start tomcat and xampp

Step 6 : Open web browser and type the command

http ://localhost/examples/servlet/FirstServlet

The output will be

Output



University Questions

1. Explain about architecture of servlet.
2. Describe the servlet architecture and various information invoked by the servlet container.

AU : May-13, Marks 6

AU : May-08, Marks 8

6.4 Life Cycle

In the life cycle of servlet there are three important methods. These methods are

1. Init
2. Service
3. Destroy

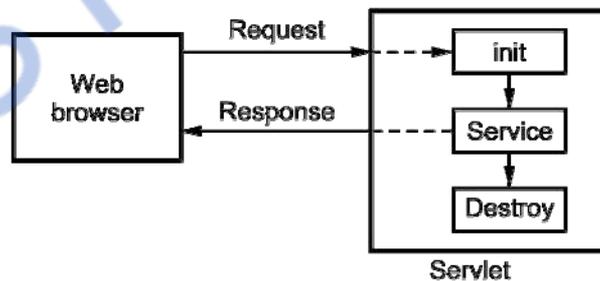


Fig. 6.4.1 Life cycle of servlet

- The client enters the URL in the web browser and makes a request. The browser then generates the HTTP request and sends it to the Web server. (Refer Fig. 6.4.1)
- Web server maps this request to the corresponding servlet.

1. Init () Method : The server basically invokes the **init()** method of servlet. This method is called only when the servlet is loaded in the memory for the first time. Using this

method initialization parameters can also be passed to the servlet in order to configure itself.

2. **service () Method** : Server can invoke the service for particular HTTP request using **service()** method. The servlets can then read the data provided by the HTTP request with the help of **service()** method.
3. **destroy () Method** : Finally server unloads the servlet from the memory using the **destroy()** method.

6.5 Servlet API

AU : Dec.-09, May-10

There are two packages used to implement the **Servlet**. These packages are

1) javax.servlet

2) javax.servlet.http

The classes and interfaces defined in these packages help in implementing the servlet.

Let us discuss them in detail -

6.5.1 The javax.servlet Package

This package is used to implement the servlet. Out of many other interfaces and classes defined in this package the most important interface is **Servlet** . All the servlet must implement this interface. The interfaces and classes that are defined in this package are enlisted below -

6.5.1.1 Interfaces

Interface	Description
Servlet	This interface defines all the life cycle methods.
ServletConfig	This interface obtains the initialization parameters.
ServletContext	Using this interface the events can be logged.
ServletRequest	This interface is useful in reading the data from the client request.
ServletResponse	This interface is useful in writing the data to the client response.

1. The Servlet Interface

This interface defines all the life cycle methods such as **init()**,**destroy()** and **service()**. All the servlets must implement this interface. Some of the methods provided by this interface are –

Method	Description
void init(ServletConfig s)	This method is called for initialising the servlet. The initialisation parameter is obtained from the ServletConfig interface.
void destroy()	This method is called when the servlet has to be unloaded.
ServletConfig getServletConfig()	This method is used to obtain the initialisation parameters.
void Service(ServletRequest req, ServletResponse res)	This method is used to implement the service that a servlet should provide. The clients request is processed and a response is given.
String getServletInfo()	This method is used to obtain description of the servlet.

2. The ServletConfig Interface

This interface is used to obtain the initialisation parameters. Various methods that are used by this interface are -

Method	Description
String getServletName()	This method is used to obtain the name of the servlet.
String getInitParameter(String p)	This method returns the value of the parameter <i>p</i>
Enumeration getinitParamernames()	This method returns the names of initialisation parameters.
ServletContext getServletContext()	This method returns the context for the servlet.

3. The ServletContext Interface

This is the interface that enables the servlet to log events and access information about their environment. Various methods of this interface are -

Method	Description
Object getAttribute(String <i>attribute_name</i>)	The value of the attribute <i>attribute_name</i> in the current session is returned.
void setAttribute(String <i>attribute_name</i> , object <i>value</i>)	The <i>attribute_name</i> is passed to the object <i>value</i> .
String getServerInfo()	This method returns the information about the server.

String log(String <i>str</i>)	Writes the <i>str</i> to the servlet log.
String getMimeType(String <i>file</i>)	It returns the MIME type of the file.

4. The ServletRequest Interface

The **ServletRequest** interface helps in gathering the information about the client. Various methods in ServletRequest are -

Method	Description
Object getAttribute(String <i>attribute_name</i>)	The value of the attribute <i>attribute_name</i> in the current session is returned.
int getContentLength()	It returns the content size of the request.
String getContentType()	It returns the type of the request.
getInputStream()	This method is used to read the binary data from the request.
getProtocol()	Returns the name of the protocol used.
getReader()	This method is useful for reading the text from the request.
getServerName()	It returns the name of the server on which the servlet is running.
int getServerPort()	It returns the port number of the server.

5. The ServletResponse Interface

This interface helps the servlet to formulate the response for the client's request. Various methods of this interface are -

Method	Description
String getCharacterEncoding()	This method returns the character encoding.
ServletOutputStream getOutputStream()	This method returns outputstream which is used to write the data for responding.
PrintWriter getWriter()	This method is used to write the character data to the response.

<code>void setContentLength(int size)</code>	This method sets the length of the content equal to the <i>size</i> .
<code>void.setContentType(String Type)</code>	This methods sets the type of the content

6.5.1.2 Classes

The **javax.servlet** package contains following classes.

Class	Description
GenericServlet	This class implements the Servlet and ServletConfig interfaces.
ServletInputStream	This class provides the input stream for reading the client's request.
ServletOutputStream	This class provides the output stream for writing the client's response.
ServletException	This class is used to raise the exception when an error occurs.

1. The GenericServlet Class

This class is useful for implementing the life cycle methods. It implements the **Servlet** and **ServletConfig** interfaces. It is called "Generic" because it does not assume that the protocol it will process will be HTTP. If we want to write our own protocol, then we must extend this class.

2. The ServletInputStream class

The **ServletInputStream** class extends the **InputStream**. This class provides an input stream which is used by a **Servlet** for reading the data from a client request. For reading the bytes from a client the method used is **readLine**.

3. The ServletOutputStream class

The **ServletOutputStream** class extends the **OutputStream**. This class provides an output stream which is used by a **Servlet** for writing the data for a client response. It also defines **print** and **println** method.

4. The ServletException class

The **javax.servlet** defines two **Servlets** **ServletException** and **UnavailableException**

6.5.2 The javax.servlet.http Package

The **javax.servlet.http** package contains a number of interfaces and classes and their functionality makes it easy to build **Servlets**. These **Servlets** then work with HTTP requests and responses.

6.5.2.1 Interface

The following table describes the interfaces that are provided in this package

Interface	Description
HttpSession	The session data can be read or written using this interface.
HttpServletRequest	The servlet can read the information from the HTTP request using this interface.
HttpServletResponse	The servlet can write the data to HTTP response using this interface.
HttpSessionBindingListener	This interface tells the object about its binding with the particular session.

1. The HttpSession

At every HTTP session, the state information is gathered. The servlet can read or write this information using **HttpSession** interface. This interface is implemented by the server.

Various methods used by this interface are as given below -

Method	Description
String getId()	This method returns the session ID.
Object getAttribute(String <i>attribute_name</i>)	The value of the attribute <i>attribute_name</i> in the current session is returned.
Enumeration getAttributeNames()	Returns the attribute names.

2. The HttpServletRequest

The **HttpServletRequest** is used to obtain the information from client's HTTP request.

Method	Description
String getMethod()	It returns the HTTP method for the client request.
String getPathInfo()	Returns the path information about the servlet path.
HttpSession getSession()	Returns the current session.
String getHeader(String <i>fields</i>)	It returns the value of header fields.
Cookie [] getCookies()	It returns the information in the cookies in the request made.
String getAuth Type()	It returns the type of Authentication.

3. The *HttpServletResponse*

This **HttpServletResponse** interface is used to formulate an HTTP response to the client.

Various methods are -

Method	Description
void addCookie(Cookie <i>cookie</i>)	This method is used to add cookie in the response.
String encodeURL(String <i>url</i>)	This method is used to encode the specified URL.
Boolean containsHeader(String <i>f</i>)	This method returns true if the response header contains the field <i>f</i> .
void sendError(int <i>code</i>)	This method sends error code to the client.

4. The *HttpSessionBindingListener*

When particular listener gets bound or unbound from a value within **HttpSession** object then this class is required. It extends the **HttpSessionEvent**. Various methods that can be defined by this interface are -

Method	Description
Object getValue()	This function returns the value of bounded or unbounded attribute.
String getName()	This function returns the name being bound or unbound.
HttpSession getSession()	This function returns the session to which the listener can be bound or unbound.

6.5.2.2 Classes

The following table describes the classes that are provided in this package

Class	Description
Cookie	This class is used to write the cookies.
HttpServlet	It is used when developing servlets that receive and process HTTP requests.
HttpSessionEvent	This class is used to handle the session events.
HttpSessionBindingEvent	When a listener is bound to a value.

1) The Cookie class

A cookie is a small piece of information that is stored in the client's machine. Sometimes cookies are useful to keep track of the user using the client machine. Various methods used by this class are –

class	Description
String getValue()	This function returns a value of the cookie.
void setValue(String s)	This function sets the value to the cookie.
String getName()	This function returns the name.

2) The HttpServlet class

The **HttpServlet** class extends **GenericServlet**. It is used when developing servlets that receive and process HTTP requests. Various methods of this class are -

Class	Description
Void doGet(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP get request.
Void doPost(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP Post request.
Void doPut(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP Put request.
Void service(HttpServletRequest req, HttpServletResponse res)	This method is invoked for processing HTTP request and response.

3) The HttpSessionEvent

This method encapsulates the session event.

4) HttpSessionBindingEvent

The **HttpSessionBindingEvent** class extends **HttpSessionEvent**. It is generated when a listener is bound to a value. The **getSession()** method obtains the session to which the listener is being bound or unbound.

University Questions

1. List out the commonly used methods to HTTP servlet.
2. Discuss four important methods of HTTP Servlet class.

AU : Dec.-09, Marks 6

AU : May-10, Marks 8

6.6 Parameter Data**AU : May-08, 14, Dec.-09**

- Many times we need to pass some information from web browser to Web server. In such case, the HTML document containing the FORM is written which sends the data to the servlet present on the web server.
- To make the form works with Java servlet, we need to specify the following attributes for the <form> tag:
 - (i) **method="method name"**: to send the form data as an HTTP POST request to the server.
 - (ii) **action="URL/address of the servlet"**: specifies relative URL of the servlet which is responsible for handling data posted from this form.
- The browser uses two methods to pass this information to web server. These methods are GET Method and POST Method.
 - (i) **GET method** : The GET method sends user information along with ? symbol called query string. For instance
`http://localhost/hello?user = aaaa&age = 20`
 In servlet, this information is processed using **doGet()** method.
 - (ii) **POST method** : This is the most reliable method of sending user information to the server from HTML form. Servlet handles this request using **doPost** method.

Difference between GET and POST

GET	POST
Using GET request limited amount of information can be sent.	Using POST large amount of information can be sent.
GET request is not secured as information is visible in URL	This is a secured request.
This request is can be bookmarked.	This request can not be bookmarked.
This request is more efficient.	This request is less efficient.

How does servlet read form data ?

Servlet makes use of following three methods to read the data entered by the user on the HTML form

1. **getParameter()** – You call request.getParameter() method to get the value of a form parameter.

2. **getParameterValues()** – Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
3. **getParameterNames()** – Call this method if you want a complete list of all parameters in the current request.

Programming Examples

Ex. 6.6.1 : Write a HTML that shows the following list :

C,C++,JAVA,C#

Define a form that contains a select statement and submit button. If the user selects the java and press the submit the web page displays “The selected language is Java”.

Write a servlet program using HttpServlet and doGet method.

Sol. : We will write two source files first one is the HTML file named **test.html** in which the list of all the desired languages is displayed along with the submit button. User has to select the language of his choice and then press the submit button.

This data made by this request will be received by the servlet named **my_choiceservlet.java** which reads the selected parameter and displays the message about the selection made. The source files are as follows -

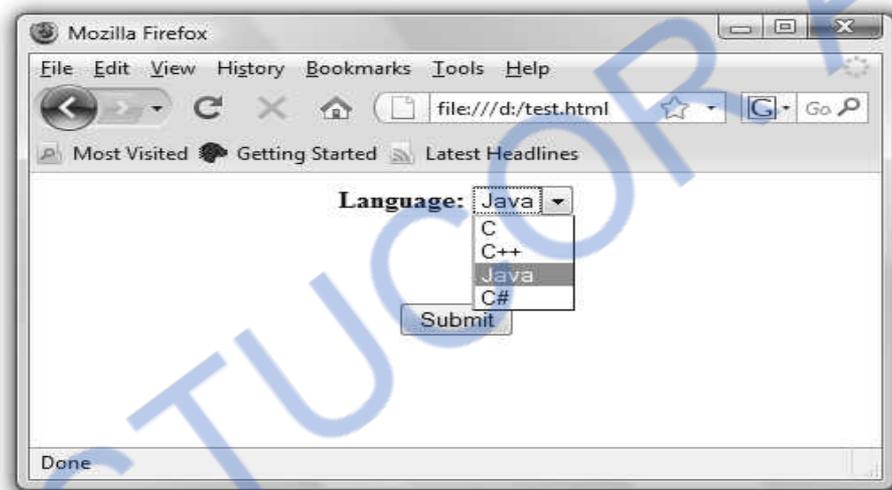
test.html

```
<html>
  <body>
    <center>
      <form name="form1" method=GET
      action="http://localhost:4040/servlets-examples/servlet/my_choiceservlet">
        <b>Language:</b>
        <select name="Language" size="1">
          <option value="C">C</option>
          <option value="C++">C++</option>
          <option value="Java">Java</option>
          <option value="C#">C#</option>
        </select>
        <br><br>
        <input type="submit" value="Submit">
      </form>
    </body>
  </html>
```

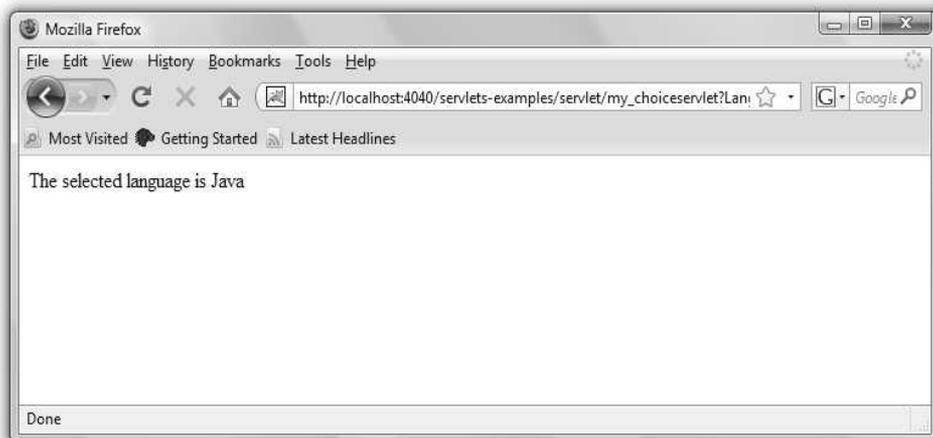
my_choiceservlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class my_choiceservlet extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException
    {
        String lang=req.getParameter("Language");
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.println("The selected language is "+lang);
        out.close();
    }
}
```



On clicking the submit button we will get following output.



Ex. 6.6.2 : Write HTML form to read user name and password. This data is sent to the servlet. If the correct user name and password is given then welcome him/her by his/her name otherwise display the message for invalid user.

Sol. : Step 1 : Create HTML form for accepting user name and password

Input.html

```
<html>
<head>
</head>
<body>
<form action="http://localhost/examples/servlets/servlet/Welcome" method = "get">
User Name:<input type="text" name="uname"/>
<br/>
Password:<input type="password" name="pwd"/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Step 2 : Create the servlet program to read user name and password and validate it.

Welcome.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Welcome extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException
    {
        PrintWriter out=res.getWriter();
        res.setContentType("text/html");

        String username=req.getParameter("uname");
        String password=req.getParameter("pwd");
        if ((username=="Ankita")&&(password=="1234"))
            out.print("Welcome "+username);
        else
            out.println("Invalid username");
    }
}
```

Ex. 6.6.3 : Write a servlet which accept two numbers using POST methods and display the maximum of them.

Sol. : Step 1 : The HTML document for inputting two numbers is as follows -

NumbersInput.html

```
<html>
<head>
<body>
<div align="center">
  <br> <br>
  <form action="http://localhost/examples/servlets/servlet/MaxNumber" method="post">
    Enter First Number :
    <input type="text" value="" name="Number1" size='5'>
    <br/><br/> Enter Second Number :
    <input type="text" value="" name="Number2" size='5'>
    <br/> <br/> <br/>
    <input type="submit" value="Submit">
  </form>
</div>
</body>
</html>
```

Step 2 : The servlet code that handles the Post method and finds the maximum of the two input numbers is as follows -

MaxNumber.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MaxNumber extends HttpServlet
{
  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
  IOException
  {
    res.setContentType("text/html");
    PrintWriter out=res.getWriter();

    // get request parameters for userID and password
    int a = Integer.parseInt(req.getParameter("Number1"));
    int b = Integer.parseInt(req.getParameter("Number2"));

    if (a>b)
      out.println("<h4>The maximum number is:" + a+"</h4>");
```

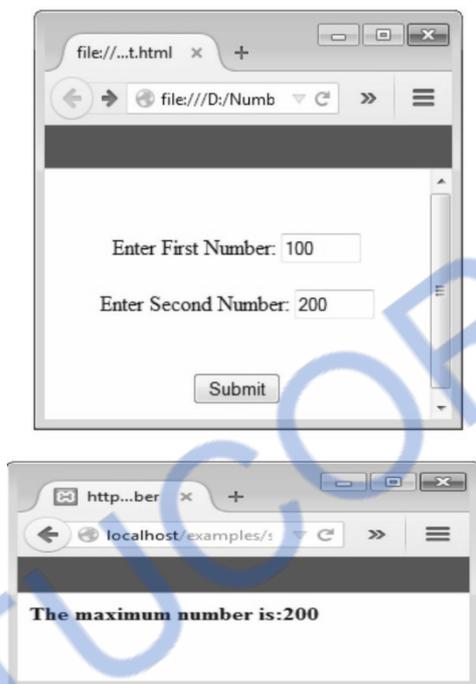
```

else
    out.println("<h4>The maximum number is :"+b+"</h4>");

}
}

```

Step 3 : The output is as follows -



Ex. 6.6.4 Write the code for converting currencies to US Dollar.

AU : May-14, Marks 8

Sol. :

Step 1 : We will write the simple HTML document, which will create the interface. **The user can select the type of currencies and will enter the value.**

```

<html>
<body>
    <formname="form1"method="post"
action="http://localhost/examples/servlets/servlet/DollarDemo">
    Select the type of Currency:
    <Select name="currency">
        <option value="Euro">Euro</option>
        <option value="Pound">Pound</option>
        <option value="Rupees">Rupees</option>
    </select>
    <br/> <br/>
    Enter the value:

```

```
<input type="text" name="currencyVal" value=""/>
<br/> <br/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

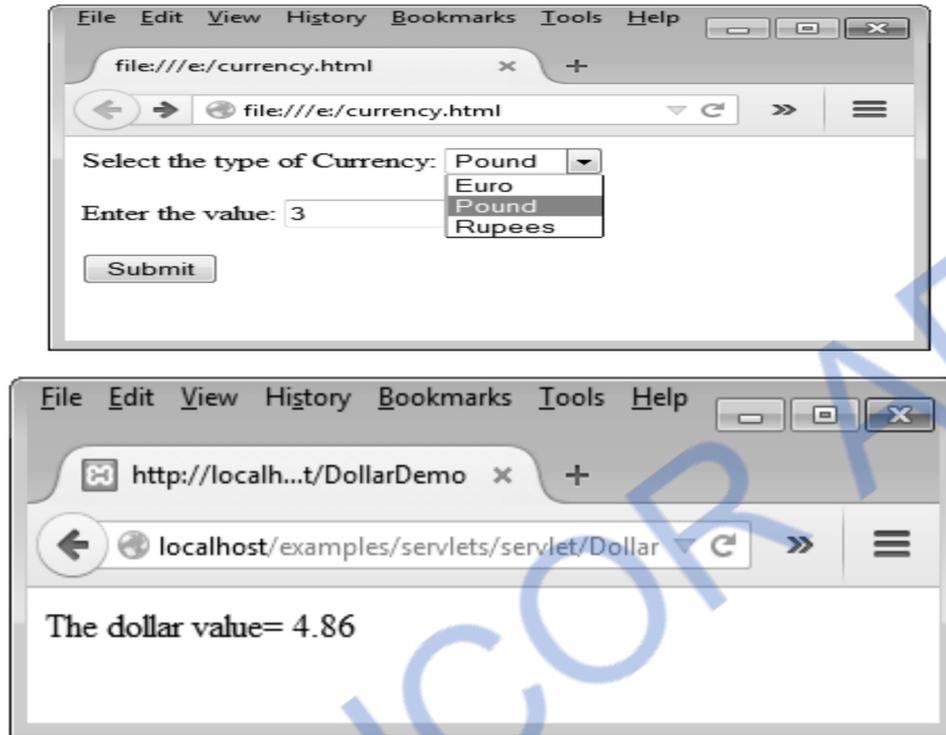
Step 2 : The java servlet for converting the currency value to dollar is as given below -

```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DollarDemo extends HttpServlet {
public void doPost(HttpServletRequest req,HttpServletResponse res)throws
    ServletException,IOException,NullPointerException
{
    res.setContentType("text/html");
    PrintWriter out=res.getWriter();
    String c=req.getParameter("currency");
    double RupeeValue,EuroValue,PoundValue;
    double DollarValue=0;
    String value=req.getParameter("currencyVal");

    if(c.equals("Rupees"))
    {
        RupeeValue=Double.parseDouble(value);
        DollarValue=RupeeValue/61;
    }
    if(c.equals("Euro"))
    {
        EuroValue=Double.parseDouble(value);
        DollarValue=EuroValue/0.77;
    }
    if(c.equals("Pound"))
    {
        PoundValue=Double.parseDouble(value);
        DollarValue=PoundValue* 1.62;
    }
    out.println("<p> The dollar value= "+DollarValue+"</p>");
    out.close();
}
}
```

Output



University Questions

1. What is the difference between the HTTP-GET and HTTP-POST request ?

AU : Dec.-09, Marks 2

2. Write the servlet program that handles the HTTP get request containing data that is supplied by the user as part of the request.

AU : May-08, Marks 8

6.7 Session Management Technique

- When you open some application, use it for some time and then close it. This entire scenario is named as **session**.
- **Session state** is a server-based state mechanism that lets web applications store and retrieve objects of any type for each **unique user session**. That is, each browser session has its own session state stored as a serialized file on the server, which is deserialized and loaded into memory as needed for each request. Refer Fig. 6.7.1.
- Because server storage is a finite resource, objects loaded into memory are released when the request completes, making room for other requests and their session objects. This means there can be more active sessions on disk than in memory at any one time.

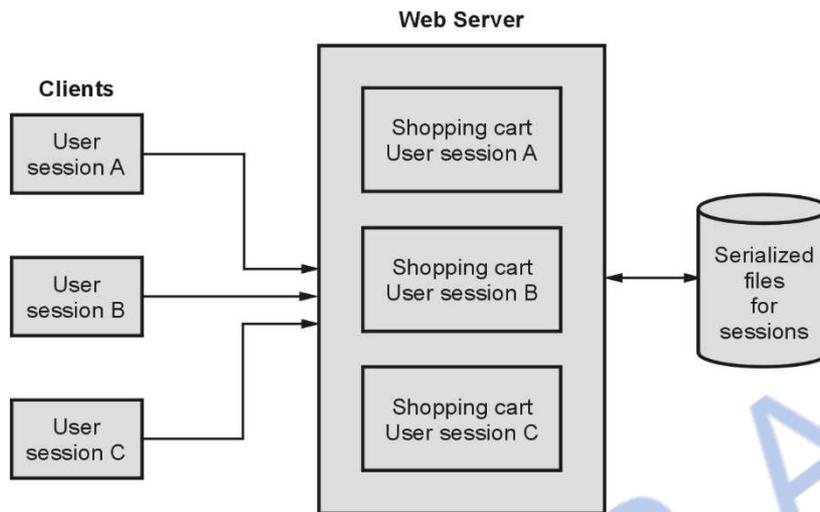


Fig. 6.7.1 Session state

- Sometimes the information about the session is required by the server. This information can be collected during the session. This process is called **session tracking**.
- There are three commonly used techniques used for session tracking and those are
 - Session ID
 - Cookies
 - URL Rewriting

6.8 Session Management using Session ID

AU : May-12,14

- **HTTP** is a **stateless protocol** in which each request is independent of the previous request. And HTTP is a protocol using which user can interact with the server via web browser and it cannot remember previously held communications but sometimes there is serious need to keep track of previous communication sessions. This can be achieved by **session tracking**.
- The **session tracking technique** is a mechanism by which we can keep track of previous sessions between server and the browser.
- For sending all state information to and fro between browser and server, usually an ID is used. This ID is basically a **session-ID**.
- Thus session-ID is passed between the browser and server while processing the information. This method of keeping track of all the information between server and browser using session-ID is called **session tracking**.
- In servlets, for creating the sessions **getSession()** method can be used. This method returns the object which stores the bindings with the names that use this object. And these bindings can be

managed using `getAttribute()`, `setAttribute()`, `removeAttribute()` methods. Actually in session tracking two things are playing an important role -

- 1) One is **HttpServletRequest** interface which supports `getSession()` method.
- 2) The another class is **HttpSession** class which supports the binding managing functions such as `getAttribute()`, `setAttribute()`, `removeAttribute()` and `getAttributeNames()`.

Let us first discuss the servlet program which returns the number of previous sessions established between client and server. Note that any client can communicate with the server using some web browser only!

Ex. 6.8.1 : Write a servlet program to keep track of number of times user is visiting the page. Display the count appropriately.

Sol. : Servlet Program(SessionServletDemo)

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SessionServletDemo extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException
    {
        res.setContentType("text/html");
        HttpSession session=req.getSession();
        String heading;
        Integer cnt=(Integer)session.getAttribute("cnt");
        if(cnt==null)
        {
            cnt=new Integer(0);
            heading="Welcome You are accessing the page for the First Time";
        }
        else
        {
            heading="Welcome once again!";
            cnt=new Integer(cnt.intValue()+1);
        }

session.setAttribute("cnt",cnt);
        PrintWriter out=res.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("</head>");
        out.println("<body>");
```

```

out.println("<center>");
out.println("<h1>"+heading);
out.println("<h2> The number of previous access= "+cnt);
out.println("</center>");
out.println("</body>");
out.println("</html>");
}
}

```

Output

To get an output we will use following steps -

1. Compile the Servlet (i.e. a java program) using **javac**. This will generate a class file.
2. Copy this class file to *C:\your tomcat directory\examples-folder\WEB-INF\classes*
3. Edit the **web.xml** file present at *C:\your tomcat directory\examples-folder\WEB-INF* like this

```

...
<servlet>
  <servlet-name>SessionServletDemo</servlet-name>
  <servlet-class>SessionServletDemo</servlet-class>
</servlet>
...
<servlet-mapping>
  <servlet-name>SessionServletDemo</servlet-name>
  <url-pattern>/servlet/SessionServletDemo</url-pattern>
</servlet-mapping>

```

4. Start tomcat server.
5. Open the web browser and give the URL for **SessionServletDemo**. Following output can be seen.



If you simply refresh your browser either by pressing F5 or clicking the refresh button then you will get following kind of output.



Program Explanation

1. Our class *SessionServletDemo* is inherited from **HttpServlet** class. And inside the **doGet()** method we can create an object *session* of **HttpSession** class. This method returns the current session for the corresponding request.
2. The **getAttribute(string attribute)** returns the value associated with the *attribute*. We can pass this *attribute* as a parameter to this method. If there is no attribute present then this method returns null. The data type of **getAttribute()** method is **void**.
3. The **setAttribute(string attribute, object value)** is a method which assigns the value passed as the object value to the attribute name. The data type of this method is **void**.

University Question

1. What is session ? Explain the session tracking systems used.

AU : May-12,14, Marks 8

6.9 Cookies

AU : May-09, 13, 17, Dec.-16

Definition : Cookies are some little information that can be left on your computer by the other computer when we access an internet.

- Generally this information is left on your computer by some advertising agencies on the internet. Using the information stored in the cookies these advertising agencies can keep track of your internet usage, liking of users.
- For the applications like on-line purchase systems once you enter your personal information such as your name or your e-mail ID then it can be remembered by these systems with the help of cookies.

- Sometimes cookies are very much dangerous because by using information from your local disk some malicious data may be passed to you. So it is upto you how to maintain your own privacy and security.

6.9.1 Servlet Support for Cookies

- The **cookie** class is used to create cookies in servlet.
- **Syntax** : The Syntax for constructor for cookies are
 - Cookie()
 - Cookie(String name, String value)
- Various methods used in Cookie are described in following table

S.N.	Method	Purpose
1.	public string getName()	It returns the name of the cookie.
2.	public String getValue()	It returns the value of the cookie.
3.	public string setName()	It sets or changes the name of the cookie.
4.	public String setValue()	It sets or changes the value of the cookie.
5.	public void addCookie(Cookie c)	The cookie is added in the response object of HttpServletResponse interface.
6.	public Cookie[] get_cookies()	All the cookies can be returned using this method with the help of HttpServletRequest interface.

- In Servlet following steps are used to support for cookies

Step 1 : Creation of Cookies :

The cookie can be created in Servlet and added to response object using **addCookie** method

```
Cookie cookie=new Cookie("user","XYZ");//creating cookie object
```

```
response.addCookie(cookie);//adding cookie in the response
```

Step 2 : Reading Cookies :

The value from the cookie can be obtained using the **getName()** and **getValue()** methods.

```
Cookie[] my_cookies=req.getCookies();
int n=my_cookies.length;
for(int i=0;i<n;i++)
{
String name=my_cookies[i].getName();
String value=my_cookies[i].getValue();
}
```

6.9.2 Examples

Below is simple HTML form in which a servlet is invoked. This servlet creates a cookie by the name **My_Cookie** and stores the value entered by you in the textbox of HTML form. You can further get the information stored in the cookie by another servlet program **getCookieServlet**. Hence we will write three programs -

1. Our normal HTML script in which some value is entered in the textbox.
2. The servlet program named `mycookieservlet` which will set a cookies and take the value entered by you in the HTML form.
3. The another servlet program named `getCookieServlet` which helps us to view the cookie.

HTML Program

```
<html>
<head>
<title>Demo of Cookie</title>
</head>
<body>
<form name="form1" method="post"
action="http://localhost:8080/examples/servlet/mycookieservlet">
<h3> Enter the value for my Cookie: </h3>
<input type="text" name="txt_data" size =30 value=" ">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Servlet Program [mycookieservlet.java]

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class mycookieservlet extends HttpServlet
{
public void doPost(HttpServletRequest req,HttpServletResponse res)
throws ServletException, IOException
{
String txt_data = req.getParameter("txt_data");
// Create cookie.
Cookie cookie = new Cookie("My_Cookie", txt_data);
// Adding cookie to HTTP response.
res.addCookie(cookie);
// Write friendly output to browser.
res.setContentType("text/html");
PrintWriter out = res.getWriter();
```

```
out.println("<h2>MyCookie has been set to : ");
out.println(txt_data);
out.println("<br><br><br>");
out.println("This page shows that the cookie has been added");
out.close();
}
}
```

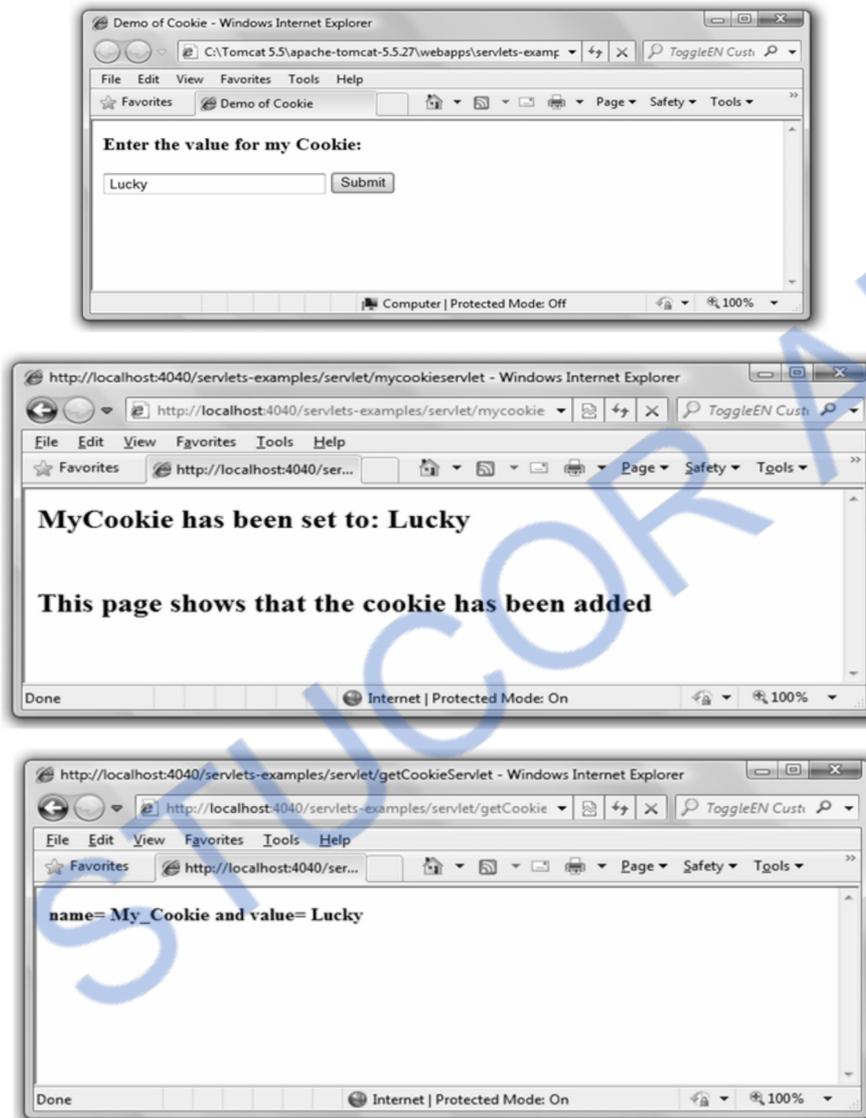
We have first created an object of Cookie class using which the cookie can be added using **addCookie()** method.

Servlet Program[getcookieservlet.java]

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class getCookieServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException
    {
        Cookie[ ] my_cookies=req.getCookies();
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.println("<b>");
        int n=my_cookies.length;
        for(int i=0;i<n;i++)
        {
            String name=my_cookies[i].getName();
            String value=my_cookies[i].getValue();
            out.println("name= "+name);
            out.println("and value= "+value);
        }
        out.close();
    }
}
```

In the above program using **getName()** and **getValue()** functions we can get the name of the cookie as well as the value of the cookie respectively. The output of all the above given three programs is as given below -

Output



Ex. 6.9.1 Write a program that allows the user to select a favorite programming language and post the choice to the servlet. The response is a web page in which user can click a link to view list of book recommendations. The cookies previously stored on the client are read by the servlet and form a web page containing the book recommendation. Use servlet, cookies and HTML.

AU : May-17, Marks 16

Sol. :

Step 1 : Create an HTML page which allows the user to select the favourite programming language. The user can submit his choice by clicking the submit button.

InputForm.html

```

<html>
<body>
<form name="form1" method="post"
action="http://localhost/examples/servlets/servlet/MyProgram">
<h3> Enter the favourite Programming Language: </h3>
<select name="Lang_ch">
    <option value="C++">C++</option>
    <option value="Java">Java</option>
    <option value="PHP">PHP</option>
</select>
<input type="submit" value="Submit">
</form>
</body>
</html>

```

Step 2 : Now create a web servlet program which reads the choice of programming language and set this choice as cookie. The web page containing a hyperlink is created in this servlet program. The code for this Servlet program is as follows –

MyProgram.java

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyProgram extends HttpServlet
{
    public void doPost(HttpServletRequest req,HttpServletResponse res)
    throws ServletException, IOException
    {
        String choice = req.getParameter("Lang_ch");
        // Create cookie.
        Cookie cookie = new Cookie("My_Cookie", choice);
        // Adding cookie to HTTP response.
        res.addCookie(cookie);
        // Write friendly output to browser with web link
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<a href='ReadChoice'><p>Click Here to read about your
        favorite programming language </p></a>");
        out.println("<br><br><br>");
        out.println("This page shows that the cookie has been added");
        out.close();
    }
}

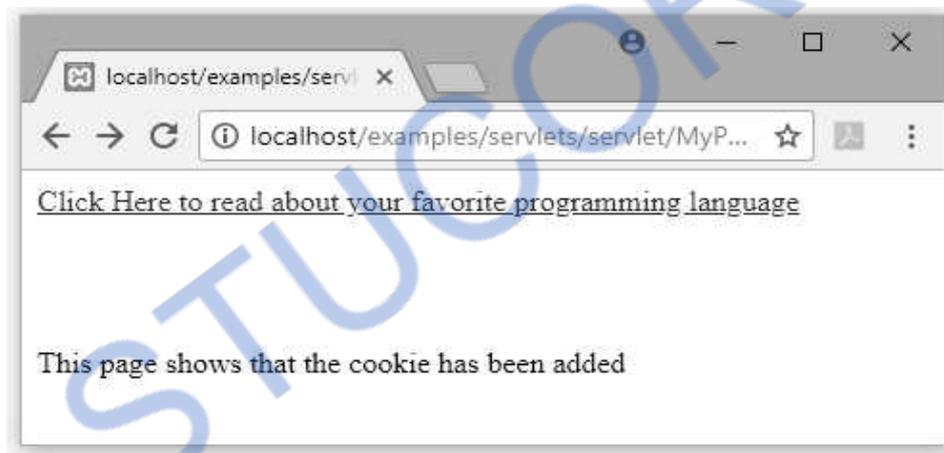
```

Step 3 : On clicking the hyperlink present in the above servlet, we can read the value of cookie(which is actually the name of the favorite programming language). The following program displays the recommended book for the programming language you have chosen. The code is as follows -

ReadChoice.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ReadChoice extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException
    {
        Cookie[] my_cookies=req.getCookies();
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.println("<b>");
        int n=my_cookies.length;
        for(int i=0;i<n;i++)
        {
            String name=my_cookies[i].getName();
            String value=my_cookies[i].getValue();
            if(value.equals("PHP"))
            {
                out.println("<b>Programming Language: </b>"+value);
                out.println("<br/><b>Book Name: </b>"+"PHP and MySQL Web
                Development by Luke Welling and Laura Thompson");
            }
            else if(value.equals("Java"))
            {
                out.println("<b>Programming Language: </b>"+value);
                out.println("<br/><b>Book Name: </b>"+"The Complete
                Reference Java");
            }
            else
            {
                out.println("<b>Programming Language: </b>"+value);
                out.println("<br/><b>Book Name: </b>"+"C++
                Programming Language by Bjarne Stroustrup");
            }
        }
    }
    out.close();
}
```

```
}  
}
```

Output

University Questions

1. How to create cookies and retrieve its value ?
2. Write a servlet to illustrate the principles of cookies and explain ?
3. Explain the use of cookies for tracking requests with a program.

AU : May-09, Marks 6**AU : May-13, Marks 16****AU : Dec.-16, Marks 8****6.10 URL Rewriting****AU : Dec.-18**

- URL rewriting is a process of appending or modifying any URL structure while loading a page.
- As we know that HTTP is a stateless protocol and it can not remember its client. We can send parameter name/value pair from one servlet to another to establish the session or to remember the client.
- The syntax of passing name/value pair using URL rewriting is
url?name1=value1 &name2=value2&??
- When a client logs in the web page, during the logging in session, the web page remembers the user name even if the client navigates from one page to another. This navigation by remembering the user name is done by URL rewriting.

Example Program

Step 1 : Create an HTML page for entering the user name. The code for HTML document is as follows -

input.html

```
<html>
  <body>
    <form action="http://localhost/examples/servlets/servlet/welcome" method="GET">
      Enter Name:
      <input type="text" value="" name="uname">
      <br/><br/>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

Step 2 : Now create another servlet that greets the user by displaying the welcome message

welcome.java

```
import java.io.*;
import javax.servlet.*;
```



```
import javax.servlet.http.*;
public class welcome extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
    {
        try
        {
            res.setContentType("text/html");
            PrintWriter out=res.getWriter();

            String user = req.getParameter("uname");
            out.println("Welcome "+user);
            out.print("<a href='mainPg?uname="+user+"'>Go to Main Page</a>");
            out.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

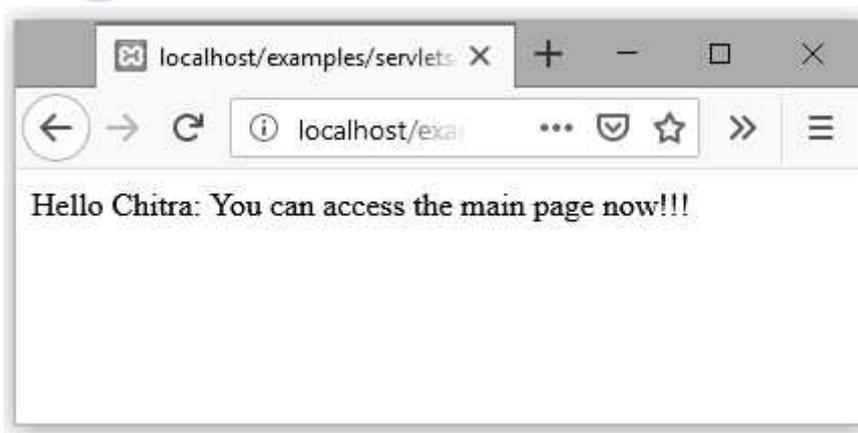
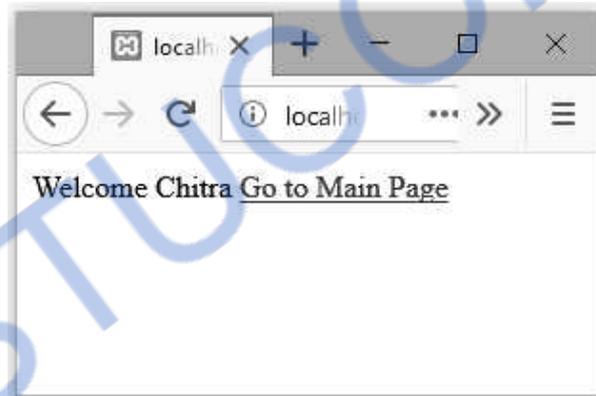
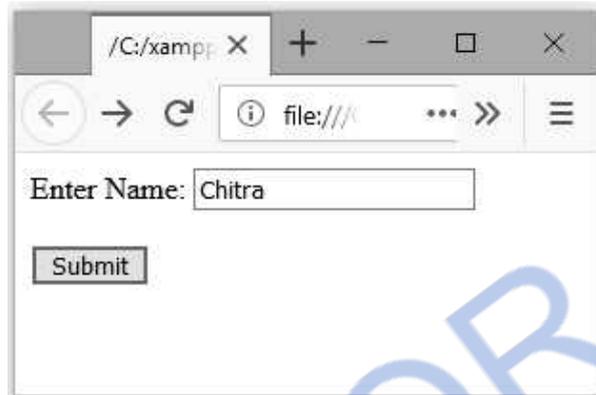
Step 3 : Now using URL rewriting technique, the user name can be sent to another web page. The code for this web page is

mainPg.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class mainPg extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        try
        {
            res.setContentType("text/html");
            PrintWriter out=res.getWriter();

            String user = req.getParameter("uname");
            out.println("Hello "+user+": You can access the main page now!!!");
            out.close();
        }
        catch(Exception e)
```

```
{  
    System.out.println(e);  
}  
}
```

Output

University Question

1. Explain various session management techniques in detail with suitable example

AU : Dec.-18, Marks 15**6.11 Data Storage**

When user demands for some service, then his/her request is stored in some data storage. The server interacts with database or some file system to fulfill the request. That obtained from database systems or file systems need to be stored at some storage repository.

Proper choice of a data storage mechanism and efficient implementation of software interacting with this mechanism is often crucial in developing a responsible and reliable web application.

There are two commonly used approaches for data storage

- (1) A Java object can be stored in a file (Refer section 10.7)
- (2) Java can store the data in a database tables.(Refer section 6.13)

6.12 Servlet and Concurrency

- Multiple requests to the same servlet may be executed at the same time. Hence concurrency container or web server is typically multithreaded.
- A servlet container may send concurrent requests through the service method of the servlet.
- In order to handle the concurrent request the **threads** are used.
- Thread is abstract representation of the process which is used to accomplish certain task. Thread uses the same address space of its process.
- The state of thread contains following information
 - Which statement to be performed next during thread execution. We call the method containing these statements as **current method**.
 - The statement that will be executed when the current method returns, and the statement that will be executed when the method containing that statement returns, and so on for all methods that have not yet returned. We will call these the **active methods** of the thread.
 - The **values of parameters** to the current method and all other active methods of the thread. The combination of this information with the preceding information is sometimes referred to as the **call stack** for the thread.
 - The values of all of the servlet's **local variables**, that is, variables defined within the current method and all other active methods.

- Multiple threads can execute in the same address space concurrently by synchronization method. Servlet uses the **lock mechanism** to synchronize these threads.
- For that purpose we need to add the keyword **synchronized** to the **doGet()** method of servlet.

For example –

```
synchronized public void doGet (HttpServletRequest request, HttpServletResponse response)
Assume count = 10;
```

User 1 Thread	User 2 Thread
<pre><Started> : count ++; //count = 11 <suspended></pre>	
	<pre><started> : count ++; //count = 12 system.out.println(count);//12 <completed></pre>
<pre><resumed> system.out.println(count);//12</pre>	

Fig. 6.12.1 Two threads running concurrently

- The Java run time automatically creates a lock for any object that contains a method with the synchronized keyword in its declaration. When a thread attempts to execute a synchronized method of such an object, the Java VM either decides to suspend execution of the thread or allows the thread to begin executing the method immediately.
- If the thread is allowed to execute immediately, then it is said to **hold the lock** for the object until it completes executing the method, at which point the thread releases the lock.
- At most one thread is allowed to hold the lock for an object at any one time.

6.13 Database and Java Servlets

AU : Dec.-09,12, May-13,16,17

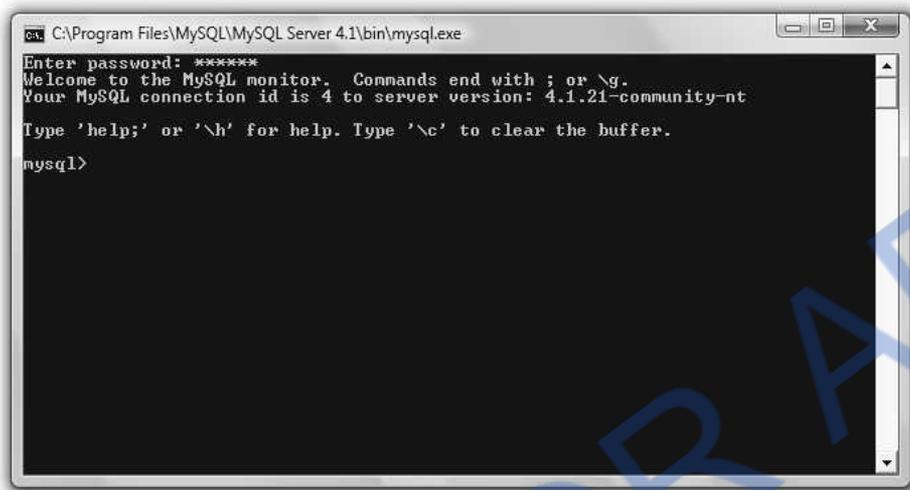
JDBC is a Java API for database access. A servlet can connect to MYSQL and sends the SQL commands to the database and get the required data. Hence we will first get introduced with structured query language SQL.

6.13.1 Structured Query Language using MySQL

The MYSQL is a open source database product which can be downloaded from the web site.



After getting installed on the machine the command prompt window for MYSQL can appear. The screenshot is as follows –



Now let us go through some MYSQL query statements which are required while handling the database.

1. Creating database

```
mysql> CREATE DATABASE mydb;
Query OK, 1 row affected (0.15 sec)
```

2. Displaying all the databases

```
mysql> SHOW DATABASES;
```

```
+-----+
| Database |
+-----+
| mydb     |
| mysql    |
| students |
| test     |
+-----+
```

```
4 rows in set (0.06 sec)
```

3. Selecting particular database

```
mysql> USE MYDB;
Database changed
```

4. Creating table

We must create a table inside a database hence it is a common practice to use create table command after USE database command. While creating a table we must specify the table fields.

```
mysql> CREATE TABLE my_table(id INT(4),name VARCHAR(20));
Query OK, 0 rows affected (0.28 sec)
```

Use of Primary Key : The primary key contains the unique value. The primary key column can not contain NUL value. Each table can have only one primary key. Following is a SQL statement used to create PRIMARY KEY.

```
CREATE TABLE student_table(
roll_no INT(4) NOT NULL AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
address VARACHAR(50) NOT NULL,
PRIMARY KEY(roll_no)
);
```

In above example the roll_no acts as a primary key for the student_table.

5. Displaying a table

After creating the table using SHOW command we can see all the existing tables in the current database.

```
mysql> SHOW TABLES;
+-----+
| Tables_in_mydb |
+-----+
| my_table      |
+-----+
1 row in set (0.00 sec)
```

6. Displaying the table fields

For knowing the various fields of the table we may use following command

```
mysql> DESCRIBE my_table;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(4) | YES  |     | NULL    |      |
| name  | varchar(20) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+
2 rows in set (0.07 sec)
```

7. Inserting values into the table

We can insert only one complete record at a time. It is as shown below -

```
mysql> INSERT INTO my_table
-> VALUES(1,'SHILPA');
Query OK, 1 row affected (0.05 sec)
```

8. Displaying the contents of the table

```
mysql> SELECT * FROM my_table;
```

```
+-----+-----+
| id | name |
+-----+-----+
| 1 | SHILPA |
+-----+-----+
1 row in set (0.06 sec)
```

We can also write SELECT statement for selecting particular row by specifying some condition such as -

```
mysql> SELECT * FROM my_table where id=1;
or
mysql> SELECT * FROM my_table where name='SHILPA';
```

Thus we can insert the rows into the table by repeatedly giving the INSERT command.

If we want to get the records in sorted manner then we use ORDER BY clause

```
mysql> SELECT * FROM my_table;
```

```
+-----+-----+
| id | name |
+-----+-----+
| 1 | SHILPA |
| 2 | SUPRIYA |
| 3 | YOGESH |
| 4 | MONIKA |
+-----+-----+
```

4 rows in set (0.00 sec)

```
mysql> SELECT * FROM my_table ORDER BY name;
```

```
+-----+-----+
| id | name |
+-----+-----+
| 4 | MONIKA |
| 1 | SHILPA |
| 2 | SUPRIYA |
| 3 | YOGESH |
+-----+-----+
```

4 rows in set (0.00 sec)

9. Updating the record

For updating the record in the database following command can be used -

```
mysql> UPDATE my_table
-> SET name='PRIYANKA'
-> WHERE id=4;
```

Query OK, 1 row affected (0.05 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> SELECT * FROM my_table;
```

```

+-----+-----+
| id | name |
+-----+-----+
| 1 | SHILPA |
| 2 | SUPRIYA |
| 3 | YOGESH |
| 4 | PRIYANKA |
+-----+-----+
4 rows in set (0.00 sec)

```

10. Deleting record

For deleting particular record from a database

```

mysql> DELETE FROM my_table
-> WHERE id=3;
Query OK, 1 row affected (0.04 sec)

```

Then use SELECT statement for displaying the contents of the table we use following command

```

mysql> SELECT * FROM my_table;
+-----+-----+
| id | name |
+-----+-----+
| 1 | SHILPA |
| 2 | SUPRIYA |
| 4 | PRIYANKA |
+-----+-----+
3 rows in set (0.00 sec)

```

11. For deleting the table

The table can be deleted using the command

```

mysql> drop table my_table;

```

12. Inner join

This query used to display the result from both the tables when at least one column from both the tables is matching-

For example

Consider table **Customer**

Cust_Id	Name	City
1	Rahul	Bombay
2	Priyanka	Pune
3	Supriya	Banglore

Consider table **Order**

Order_Id	Cust_Id	Order_Number
10	3	1234
20	1	5678
30	1	8900

Then we can make use of inner join query as follows -

```
SELECT Customer.Name, Customer.city, Order.Order_Number
FROM Customer
INNER JOIN Order
ON Customer.Cust_Id=Order.Order_Id
```

Name	City	Order_Number
Rahul	Bombay	5678
Rahul	Bombay	8900

13. Order By Clause

If we want to get the records in sorted manner then we use ORDER BY clause

```
mysql> SELECT * FROM my_table;
```

id	name
1	SHILPA
2	SUPRIYA
3	YOGESH
4	MONIKA

```
mysql> SELECT * FROM my_table ORDER BY name;
```

id	name
4	MONIKA
1	SHILPA
2	SUPRIYA
3	YOGESH

6.13.2 JDBC

- JDBC stands for Java DataBase Connectivity.
- JDBC is nothing but an API (i.e. Application Programming Interface).

- It consists of various classes, interfaces, exceptions using which Java application can send SQL statements to a database. The SQL is a Structured Query Language used for accessing the database.
- JDBC is specially used for having connectivity with the RDBMS packages (such as Oracle or MYSQL) using corresponding JDBC driver.

The **JDBC API** is a set of classes, interfaces and exceptions used for establishing connection with data source. This JDBC API is defined in the **java.sql** and **javax.sql** packages. We use following core JDBC classes and interfaces that belong to **java.sql** package.

- **DriverManager** - When Java application needs connection to the database it invokes the **DriverManager** class. This class then loads JDBC **drivers** in the memory. The DriverManager also attempts to open a connection with the desired database. For JSP-MYSQL connectivity we can get DriverManager class as -

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

- 1) **Connection** - This is an interface which represents connectivity with the data source. The **connection** is used for creating the **Statement** instance.

```
Connection conn=
```

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/databaseName","userName","password");
```

- 2) **Statement** - This interface is used for representing the SQL statements. Some examples of SQL statements are

- SELECT * FROM students_table;
- UPDATE students_table set name='Neel' where roll_no='1';
- DELETE from students_table WHERE roll_no='10';

The code for statement is as follows -

```
Statement stmt=conn.createStatement();
```

```
stmt.executeQuery("select * from mytab");
```

There are two specialised Statement types: **PreparedStatement** and **CallableStatement**. The **PreparedStatement** represents the precompiled SQL statements.

For instance :

```
SELECT * from students_table WHERE name=?
```

The placeholder represented by ? is used in this type of statement. There are special type setter methods that assign the values to the placeholders before the SQL statement is executed.

CallableStatement is represents the stored procedures. These are similar to **PreparedStatement**. In this type of Statements we can assign the methods for the types of output arguments.

- 3) **ResultSet** - This interface is used to represent the database result set. After using SELECT SQL statement, the information obtained from the database can be displayed using **ResultSet**.
- 4) **SQLException** - For handling SQL exceptions this interface is used.

6.13.3 Connectivity between Database and Servlet

Following steps are used to connect JDBC to MYSQL

Step 1 : Import java.sql.* package in the JDBC program

Following line can be included in your JDBC program at the beginning.

```
import java.sql.*
```

Step 2 : Load JDBC Driver.

The JDBC driver for MYSQL can be loaded using following statement

```
Class.forName("com.mysql.jdbc.Driver");
```

Step 3 : Get connection using the Driver Manager

```
conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/my_database","root","password");
```

Step 4 : Create Statement

```
stmt = conn.createStatement();
```

Step 5 : Execute Query

```
String sql = "SELECT Roll,StudName FROM my_table";  
ResultSet rs = stmt.executeQuery(sql);
```

Step 6 : Display the result

Ex. 6.13.1 : Explain the JDBC URL with appropriate examples.

Sol. : JDBC URL with examples : The databases are referenced by Java runtime engine using JDBC URL. The syntax of these URLs depends on the database driver you are using.

The list of common drivers along with JDBC URL is

1. ODBC Driver

Syntax :

```
jdbc:odbc:<data source name>
```

Example :

```
jdbc:odbc:MyDataSource
```

2. MYSQL

Syntax :

```
jdbc:mysql://[host][:port]/[database]
```

Example :

```
jdbc:mysql://localhost:3306/Mydatabase
```

3. ORACLE

Syntax :

```
jdbc:oracle:<drivertype>:<user>/<password>@<database>
```

Example :

```
jdbc:oracle:thin:myuser/mypassword@localhost:mydatabase
```

6.13.4 Servlet-Database Examples

For executing following programs –

- (1) JDK must be installed
- (2) XAMPP must be installed
- (3) Eclipse must be installed. Also add jar file **mysql-connector-java**. Get it downloaded from Internet and save it at suitable location. Then configure Eclipse to use MySQL, for that purpose :
 - (i) Right click on your Project.
 - (ii) Click on Properties->Java Build Path->Libraries.
 - (iii) Click Add External JARs... button and select **mysql-connector-javaXXX.jar file**.

Ex. 6.13.2 Consider a database table with the following schema.(PNR No,status). Write a servlet program to display the status, given the PNR number. **AU : Dec.-09, Marks 10, Dec.-12, Marks 16**

Sol. : Java Program[PNRDemo.java]

```
import java.io.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
public class PNRDemo extends HttpServlet
{
    public void service(HttpServletRequest request,HttpServletResponse response)
    throws IOException, ServletException
    {
        // connecting to database
        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;
        PrintWriter out = response.getWriter();
```

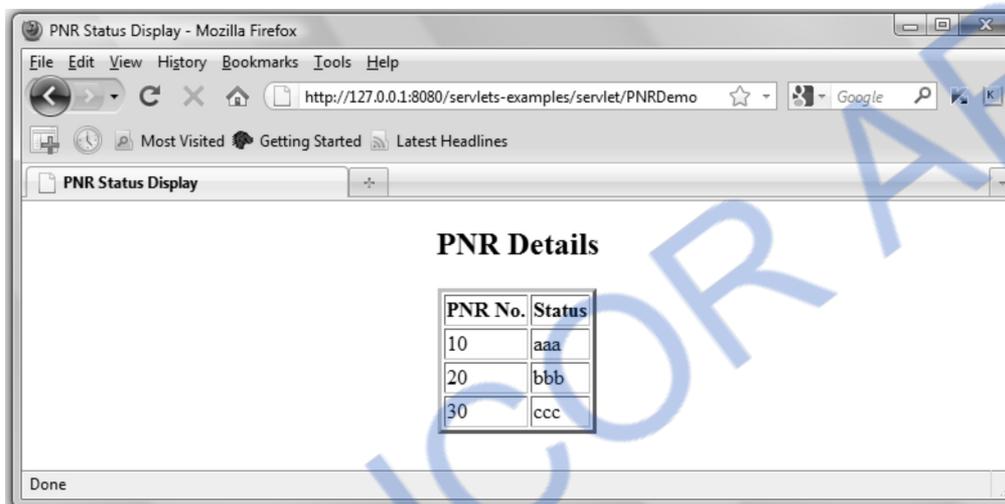
```
try
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    con =DriverManager.getConnection ("jdbc:odbc:PNRDemo","","");
}catch(ClassNotFoundException e){e.printStackTrace();}
catch(SQLException e){e.printStackTrace();}
catch(Exception e){e.printStackTrace();}
try
{
    stmt = con.createStatement();
    rs = stmt.executeQuery("SELECT * FROM PNR_table");
    // displaying records
    response.setContentType("text/html");
    out.println("<html>");
    out.println("<head> <title>PNR Status Display</title> </head>");
    out.println("<body>");
    out.print("<center>");
    out.print("<h2>PNR Details </h2>");
    out.print("<table border=3>");
    out.print("<th>PNR No. </th>");
    out.print("<th>Status</th>");
    while(rs.next())
    {
        out.print("<tr>");
        out.print("<td>");
        out.print(rs.getObject(1).toString());
        out.print("</td>");
        out.print("<td>");
            out.print(rs.getObject(2).toString());
        out.print("</td>");
        out.print("</tr>");
    }
    out.print("</table>");
    out.print("</center>");
    out.println("</body> </html>");
}catch (SQLException e) { }
finally {
    try {
        if(rs != null) {
            rs.close();
            rs = null;
        }
        if(stmt != null) {
            stmt.close();
            stmt = null;
        }
        if(con != null) {
            con.close();
```

```

    con = null;
  }
} catch (SQLException e) {}
}
out.close();
} //end of service function
} //end of class

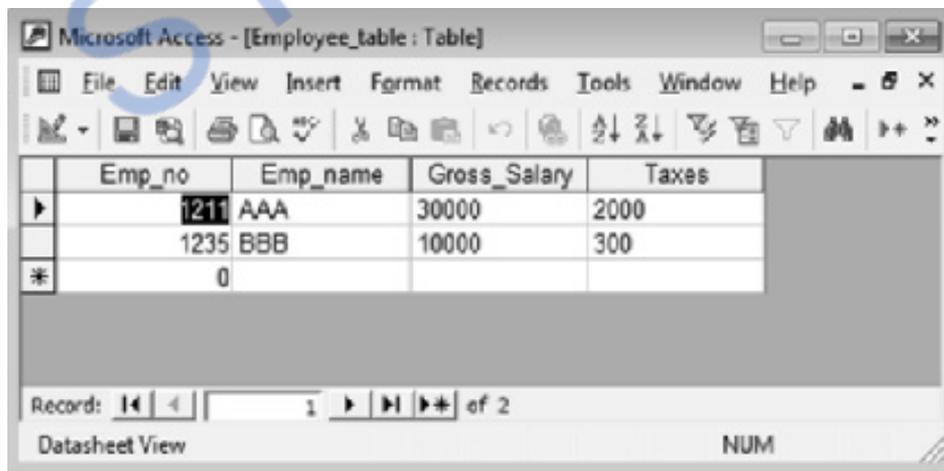
```

Output



Ex. 6.13.3 Write a Java Servlet to display net salary of employee, Use JDBC connectivity to get employee details from database. **AU : May-13, Marks 16**

Sol. : Step 1 : Create a employees database as follows -



Step 2 : The servlet for computing the net salary is as given below -

```
import java.io.*;
```

```
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class empdbase extends HttpServlet
{
    public void service(HttpServletRequest request,HttpServletResponse response)
    throws IOException, ServletException
    {
        // connecting to database
        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;
        PrintWriter out = response.getWriter();
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con =DriverManager.getConnection ("jdbc:odbc:Employees","","");
        }catch(ClassNotFoundException e){e.printStackTrace();}
        catch(SQLException e){e.printStackTrace();}
        catch(Exception e){e.printStackTrace();}
        try
        {
            stmt = con.createStatement();
            rs = stmt.executeQuery("SELECT * FROM Employee_table");
            // displaying records
            response.setContentType("text/html");
            out.println("<html>");
            out.println("<head > <title>Servlet Database Connectivity</title> </head>");
            out.println("<body>");

            out.print("<center>");
            out.print("<h2>Employees Database</h2>");
            out.print("<table border=3>");
            out.print("<th>Emp_ID</th>");
```

```
out.print("<th>Name</th>");
out.print("<th>Gross</th>");
out.print("<th>Taxes</th>");
out.print("<th>Net Salary</th>");
while(rs.next())
{
    int tot=0;
    String gross=rs.getString(3);
    String deductions=rs.getString(4);

    int g=Integer.parseInt(gross);
    int d=Integer.parseInt(deductions);

tot=g-d;//net salary of employee
    out.print("<tr>");
    out.print("<td>");
    out.print(rs.getInt(1));
    out.print("</td>");
    out.print("<td>");
    out.print(rs.getString(2));
out.print("</td>");
out.print("<td>");
    out.print(g);
    out.print("</td>");
    out.print("<td>");
    out.print(d);
    out.print("</td>");
    out.print("<td>");
    out.print(tot);
    out.print("</td>");
    out.print("</tr>");
}
out.print("</table>");
out.print("</center>");
out.println("</body></html>");
}catch (SQLException e) { }
finally {
try {
    if(rs != null) {
        rs.close();
        rs = null;
    }
    if(stmt != null) {
        stmt.close();
    }
}
```

```

        stmt = null;
    }
    if(con != null) {
        con.close();
        con = null;
    }
} catch (SQLException e) {}
}
out.close();
} //end of service function
} //end of class

```

Ex. 6.13.4 Assume that a database has a table Employee with two columns EmployeeID and Name. Assume that the administrator user id and password to access the database table are, scott and tiger. Write a JDBC program that can query and print all the entries in the table Employee. Make the database connection using a type2 driver database.driver and connection string jdbc:db:oci

AU : May-16, Marks 16

Sol. : The JDBC code for displaying the employees in Employee table is as follows -

DisplayEmp.java

```

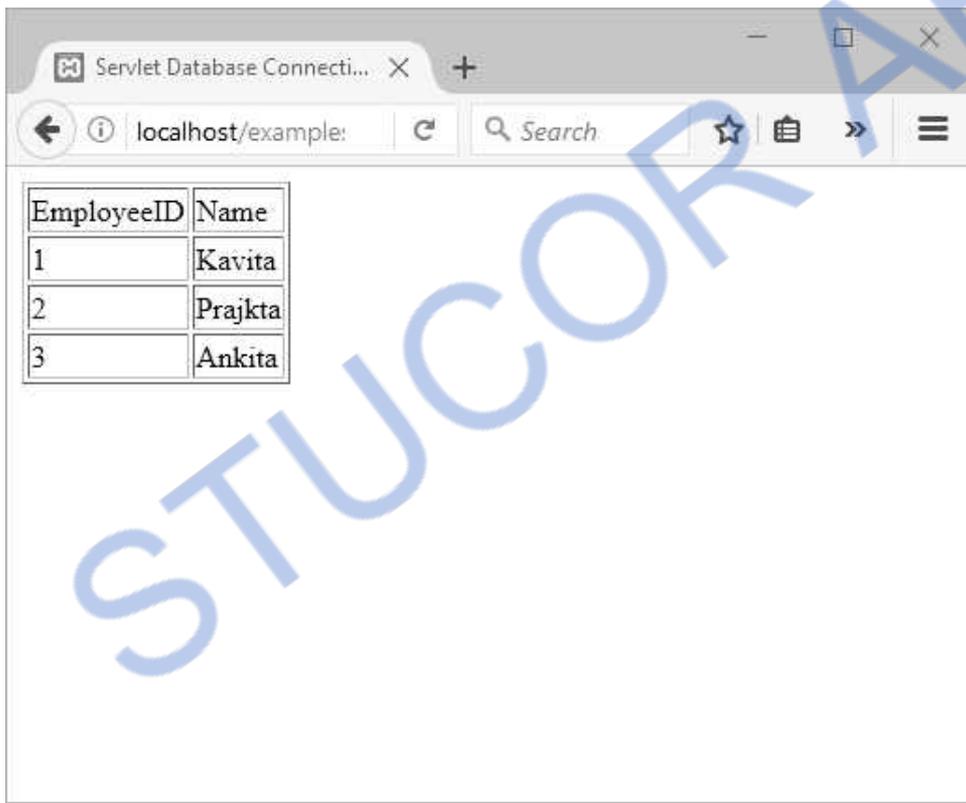
import java.io.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class DisplayEmp extends HttpServlet
{
    public void service(HttpServletRequest request,HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Servlet Database Connectivity</title></head>");
        out.println("<body>");
        // connecting to database
        Connection con = null;
        Statement stmt = null;
        ResultSet rs=null;
        try
        {

```

```
Class.forName("oracle.jdbc.OracleDriver");
String dbURL = "jdbc:oracle:oci:@my_database";
Properties properties = new Properties();
properties.put("user", "scott");
properties.put("password", "tiger");
properties.put("defaultRowPrefetch", "20");
con = DriverManager.getConnection (dbURL,properties);
stmt = con.createStatement();
rs=stmt.executeQuery("select * from Employee");
out.print("<table border=1>");
out.print("<tr><td>EmployeeID</td><td>Name</td></tr>");
while(rs.next())
{
    //Retrieve by column name
    int id = rs.getInt("ID");
    String name = rs.getString("Name");
    out.print("<tr><td>"+id+"</td><td>"+name);
    out.print("</td></tr>");
}
out.println("</table></body></html>");
}catch (SQLException e)
{
    throw new ServletException("Servlet Could not display records.", e);
}
catch (ClassNotFoundException e)
{
    throw new ServletException("JDBC Driver not found.", e);
}
finally
{
    try
    {
        if(rs != null)
        {
            rs.close();
            rs = null;
        }
        if(stmt != null)
        {
            stmt.close();
            stmt = null;
        }
        if(con != null)
```

```
{
    con.close();
    con = null;
}
}
catch (SQLException e) {}
} //end of finally
out.close();
} //end of service function
} //end of class
```

Output

Ex.6.13.5 Write complete query application for books database using JDBC. **AU : May-17, Marks 16**

Sol. :

```
import java.io.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
public class hms extends HttpServlet
```

```
{
public void service(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException
{
// connecting to database
Connection con = null;
Statement stmt = null;
ResultSet rs = null;
PrintWriter out = response.getWriter();
try
{
Class.forName("sun.jdbc.odbc.jdbcodbcDriver");
con = DriverManager.getConnection("jdbc:odbc:books","","");
}catch(ClassNotFoundException e){e.printStackTrace();}
catch(SQLException e){e.printStackTrace();}
catch(Exception e){e.printStackTrace();}
try
{
stmt = con.createStatement();
rs = stmt.executeQuery("SELECT * FROM books_table");
// displaying records
response.setContentType("text/html");
out.println("<html>");
out.println("<head><title>Servlet Database Connectivity</title></head>");
out.println("<body>");
out.print("<center>");
out.print("<h2>Books Database</h2>");
out.print("<table border=3>");
out.print("<th>ISBN</th>");
out.print("<th>Name</th>");
out.print("<th>Author</th>");
out.print("<th>Publication</th>");
out.print("<th>Edition</th>");
while(rs.next())
{
int tot=0;
out.print("<tr><td>");
out.print(rs.getString(1));
out.print("</td><td>");
out.print(rs.getString(2));
out.print("</td><td>");
out.print(rs.getString(3));
out.print("</td><td>");
out.print(rs.getString(4));
```

```
        out.print("</td><td>");
        out.print(rs.getString(5));
        out.print("</td></tr>");
    }
    out.print("</table>");
    out.print("</center>");
    out.println("</body></html>");
} catch (SQLException e) { }
finally {
try {
    if(rs != null) {
        rs.close();
        rs = null;
    }
    if(stmt != null) {
        stmt.close();
        stmt = null;
    }
    if(con != null) {
        con.close();
        con = null;
    }
} catch (SQLException e) {}
}
out.close();
} //end of service function
} //end of class
```

Two Marks Questions with Answers

Q.1 What are servlets ?

AU : Dec.-17

Ans. : Servlets are simple Java programs that run on the server. The servlets are used along with the http protocol. Hence sometimes they are also referred as http servlets. Servlets make use of standard packages such as javax.servlet and javax.servlet.http.

Q.2 What are the uses of servlets ?

Ans. : Following are the uses of servlets -

Servlets can process and store the data submitted by an HTML form.

Servlets are useful for providing the dynamic contents. For example retrieving and updating the databases.

Servlets can be used in the cookies and session tracking. Cookies are small programs which can make use of the information submitted on currently accessed web pages. Similarly session tracking are the useful programs that keeps track of all previously accessed the web pages.

Q.3 Explain the Servlet API life cycle methods in brief.

AU : May-12, 13

Ans. : The **init** method is called when the servlet is loaded in the memory for the first time. Using this method the initialization parameters can also be passed to the servlet. Then using **service()** method the server can invoke the service for particular HTTP request. Finally the server unloads the servlet from the memory using **destroy()** method.

Q.4 Explain the difference between get request type and post request type.

AU : May-12, 13

Ans. : In HTTP-GET request the **doGet** method is used. In HTTP-POST request the **doPost** request is used.

When user submits his request using **doGet** method then the URL string displays the request submitted by the user. But if the **doPost** method is used then URL string does not show the submitted contents.

Q.5 What is the use of web config file ?

Ans. : The web config file specifies the configuration for the web application. The URL for the servlets can be specified using the web config file. It is a XML file usually located in WEB-INF folder.

Q.6 Specify the need for client and server side scripting.

Ans. : The client side scripting is useful for presenting the data to the user or to submit the request to the server. Various scripting languages such as XHTML, JavaScript, DHTML are used as client side scripting languages. The applets can also be embedded in these scripts.

The server side scripting languages are used to process the request submitted by the client. The server side scripting can be done using servlets, JSP, ASP, PHP and so on.

Q.7 Explain cookies.

AU : May-08, 10, 13, Dec.-12

Ans. : Cookies are some little information that can be left on your computer by the other computer when we access an internet. The information is stored in the cookies using the name value pair.

Q.8 What is JDBC ?

AU : Dec.-17

Ans. : The JDBC stands for Java Database Connectivity. JDBC is nothing but an API(Application Programming Interface). It consists of various classes, interfaces, exceptions using which Java application can send SQL statements to a database. JDBC is specially used for the connectivity of Java with RDBMS packages.

Q.9 What is driver manager ?

Ans. : The driver manager is a class that connects the Java application or servlet to the JDBC driver. The driver manager also attempts to open a connection with the desired database using the method `DriverManager.getConnection()`

Q.10 What do you mean by result set ?

Ans. : The Result Set is a set of rows from a database, as well as meta-information about the query such as the column names, and the types and sizes of each column. The **ResultSet** interface is used to represent the database result set.

Q.11 What is the primary purpose of an HTTP get request ?**AU : May-10**

Ans. : The HTTP GET request makes use of doGet method. The user can submit his request using the doGet request. When user submits his request using doGet method the URL string is displayed along with the URL.

Q.12 How can you create JDBC statements ?

Ans. : The JDBC API classes are supported by the Java package java.sql. Hence we must import java.sql.* using following statement -

```
import java.sql.*;
```

We have to use following statement for referring the JDBC-ODBC Bridge.

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

There is a JDBC Driver Manager which connects your Java application or servlet to the JDBC driver. We can try to connect our Java program to our database with JDBC-ODBC Bridge using the **DriverManager.getConnection()** method, in which the first parameter is passed as ***jdbc:odbc:your_database_name***. The JDBC-ODBC driver helps to translate the JDBC function calls into the ODBC function calls. The JDBC for any database can be implemented only when the ODBC driver is available. Note that JDBC-ODBC bridge is implemented in **sun.jdbc.odbc**.

The connection to this database can be done using the JDBC-ODBC driver with the help of following statement in our Java program.

```
DriverManager.getConnection("jdbc:odbc:My_database", " ", " ");
```

Where first parameter specifies the name of the database file, second parameter specifies the user name and the third parameter specifies the password.

Q.13 What are HTTP servlets ?

Ans. : The HTTP servlets are nothing but the servlets only. As servlets are commonly used with the HTTP(Hyper Text Transfer Protocol), they are also called as HTTP servlets.

Q.14 Name the two packages from which most of the servlet classes can be used.

Ans. : The javax.servlet and javax.servlet.http are the two packages from which the classes are used by the servlet.

Q.15 What are HttpServletRequest and HttpServletResponse ?

Ans. : The HttpServletRequest and HttpServletResponse are the two commonly used interfaces from the **javax.servlet.http** package. The HttpServletRequest enables the servlet to read data

from the HTTP request and `HttpServletResponse` enables the servlet to write the data to the HTTP response.

Q.16 List the two commonly used exceptions used by the servlet.

Ans. : The two commonly used basic exceptions of servlet are **IOException** and **ServletException**.

Q.17 Write the code to return the full URL of a document.

AU : Dec.-11

Ans. : The code will be

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class URLReturn extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
    {
        PrintWriter out= res.getWriter();
        String FullURL = req.getRequestURL().toString();
        String queryString = req.getQueryString();
        if (queryString != null)
        {
            FullURL += "?" + queryString;
        }
        out.println(FullURL);
    }
}
```

Q.18 How is session tracking is achieved by the URL rewriting ?

AU : Dec.-11

Ans. : URL rewriting is a technique in which the information is embedded into the URL. When the user requests for some web page via the browser the Server responds to the browser by writing some session id. When the user requests for the another page using the same URL, then the server will come to know that the request is related to the previous page and the task will be considered to be in the same session. When browser gets closed then the session gets lost. In such a case URL will not be available. So when the user opens the browser again then server can not associate the current request with previous session and it will be treated as a new session. Thus using URL the session can be tracked.

Q.19 What is the difference between ServletContext and ServletConfig ?

AU : May-11

Ans. : The **ServletContext** defines a set of methods that a servlet uses to communicate with its servlet container. The **ServletContext** object is contained within the **ServletConfig** object and during the initialization of servlet the web server provides it to servlet.

The ServletConfig object is created after a servlet is instantiated and its default constructor is read. It is created to pass initialization information to the servlet.

Q.20 State the use of ServletContext object.

AU : Dec.-13

Ans. : The servletContext object can be used to communicate with the servlet container. When one ServletContext object is created then it can be accessed by all the servlets in the web container. For instance - if you want to share some data to all the servlets then you can share this data using the ServletContext object.

Q.21 What is a servlet container ? Specify its function.

AU : May-14

Ans. : The servlet container is a part of web server. Its function is to generate the dynamic web pages on the server using the input provided by the user. Thus servlet container is an environment on the web server in which the servlets can be executed to provide information to its user dynamically.

Q.22 What are the two methods used to send a request to a server ?

AU : May-14

Ans. : The HTTP GET and HTTP POST are the two methods used to send a request to a server.

Q.23 Write the code segment to store current server time in session using Java Servlet API

AU : May-16

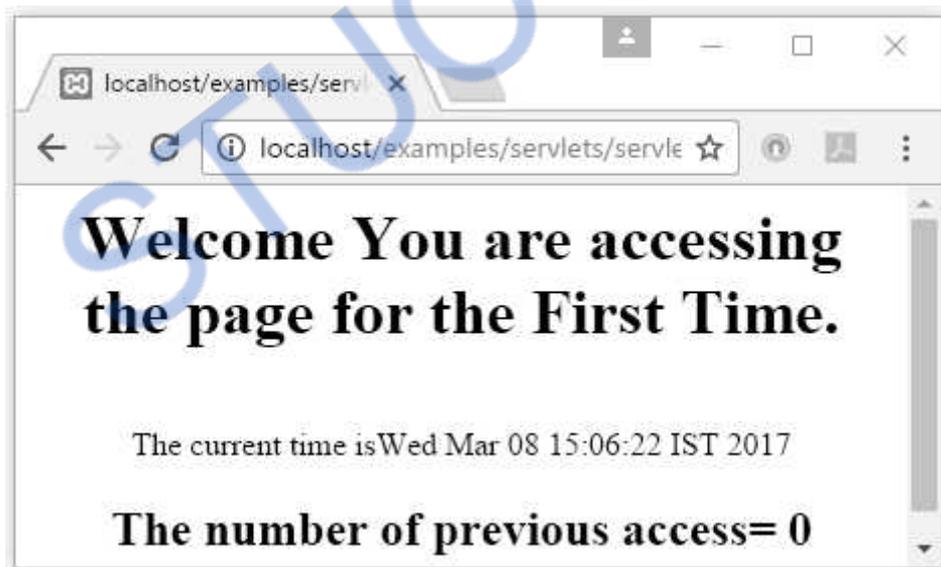
Ans. :

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class SessionServletDemo extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException
    {
        res.setContentType("text/html");
        HttpSession session=req.getSession();
        String heading;
        Date now =new Date();
        Integer cnt=(Integer)session.getAttribute("cnt");
        if(cnt==null)
        {
            cnt=new Integer(0);
            heading="Welcome You are accessing the page for the First Time.";
        }
        else
        {
```

```
heading="Welcome once again!";
cnt=new Integer(cnt.intValue()+1);
}

session.setAttribute("cnt",cnt);
PrintWriter out=res.getWriter();
out.println("<html>");
out.println("<head>");
out.println("</head>");
out.println("<body>");
out.println("<center>");
out.println("<h1>"+heading+"</h1>");
String str=now.toString();
out.println("<br/>The current time is"+str);
out.println("<h2> The number of previous access= "+cnt+"</h2>");
out.println("</center>");
out.println("</body>");
out.println("</html>");
}
}
```

Output



Q.24 Explain the servlet interface and its methods.

AU : Dec.-16

Ans. : This interface defines all the life cycle methods such as **init()**, **destroy()** and **service()**. All the servlets must implement this interface. Some of the methods provided by this interface are -

Method	Description
<code>void init(ServletConfig s)</code>	This method is called for initialising the servlet. The initialisation parameter is obtained from the ServletConfig interface.
<code>void destroy()</code>	This method is called when the servlet has to be unloaded.
<code>ServletConfig getServletConfig()</code>	This method is used to obtain the initialisation parameters.
<code>void Service(ServletRequest req, ServletResponse res)</code>	This method is used to implement the service that a servlet should provide. The clients request is processed and a response is given.



Notes

STUCOR APP

Unit IV

7

Separating Programming and Presentation

Syllabus

Separating Programming and Presentation : JSP Technology Introduction-JSP and Servlets-Running JSP Applications Basic JSP-JavaBeans Classes and JSP-Tag Libraries and Files-Support for the Model-View-Controller Paradigm- Databases and JSP.

Contents

7.1 JSP Technology

7.2 JSP and Servlet

7.3 JSP Document Structure

7.4 Running JSP Application

7.5 Declaration

7.6 Directives

7.7 Comments in JSP

7.8 Scripting Elements

7.9 Actions and Templates

7.10 JavaBean Classes and JSP **May-13,** Marks 10

7.11 Tag Libraries and Files

7.12 Support For Model View Controller Paradigm **May-12,** Marks 8

7.13 Database and JSP

Two Marks Questions with Answers

7.1 JSP Technology

- JSP stands for **Java Server Pages**.
- It is an alternative way than the servlet to build the dynamic web pages. It is built **on top of the servlet**.

7.2 JSP and Servlet

Following are some advantages of Java Server pages -

1. JSP is useful for **server side programming**.
2. JSP can be used **along with servlets**. Hence **business logic** for any application can be developed using JSP.
3. **Dynamic contents** can be handled using JSP because JSP allows scripting and element based programming.
4. JSP allows creating and using our own **custom tag libraries**. Hence any application specific requirements can be satisfied using custom tag libraries. This helps the developer to develop any kind of application.
5. JSP is a **specification and not a product**. Hence developers can develop **variety of applications** and add up to performance and quality of software products. Due to this many companies are ready to invest in JSP technology.
6. JSP is an essential component of J2EE. Hence using JSP it is possible to develop simple as well as complex applications.

Difference between Servlet and JSP

Sr. No.	JSP	Servlet
1.	JSP is a scripting language that generates dynamic content.	Servlets are the Java programs that can be compiled to generate dynamic content.
2.	JSP runs slower than servlet.	Servlet runs faster than JSP.
3.	In Model View Controller JSP acts as view.	In Model View Controller servlet acts as controller.
4.	JSP can build custom tags.	There is no facility of creating custom tags.
5.	JSP can directly call Java beans.	Servlets have no facility of calling Java bean.
6.	It is easier to code in JSP.	The servlets are basically complex Java programs.

7.3 JSP Document Structure

Every JSP document contains four elements -

- 1. Template text :** The XML of HTML code can be directly used in JSP document. This is called the static or fixed part of the document. This markup is called template text. This part is not modified by JSP container.
- 2. Action elements :** The action elements are used to create the documents dynamically. The action elements is written in the form i) Opening tag ii) Action body iii) Closing tag. For example

<code><jsp:include></code>	This tag works as a subroutine. It includes the response from servlet or JSP when request is being processed.
<code><jsp:forward></code>	This tag helps in forwarding the current request to servlet or to JSP page.
<code><jsp:element></code>	This tag generated the XML elements dynamically.
<code><jsp:text></code>	This tag is used to handle template text. When JSP pages are written as XML documents then this tag is used.

- 3. Directives :** JSP directives control the processing of entire JSP Page. It gives direction to the server regarding the processing of a page. The directive elements are enclosed within the `<%@` `%>` delimiters. For example -

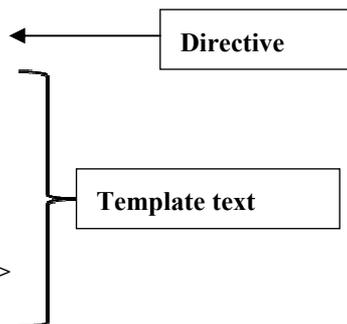
```
<%@page import="java.io.*" %>
<%@page language="java" %>
<%@page contentType="text/html" %>
```

- 4. Scriptlets :** Scriptlets are nothing but java code enclosed within `<%` and `%>` tags. For example we can write following statement in JSP document

```
<p>
    <% out.print("JSP is eqaul to HTML and JAVA "); %> //Scriptlet
</p>
```

Example of JSP document

```
<%@ page language="java" contentType="text/html" %>
<html>
    <head>
        <title>Demo for Template Text</title>
    </head>
    <body bgcolor="gray">
        <h1>Twinkle Twinkle little stars</h1>
        <h2>How I wonder what you are!!!</h2>
        <li>like a diamond in the sky</li>
    <p>
```



```

                <% out.print("JSP is equal to HTML and JAVA "); %>
            </p>
        </body>
    </html>

```



7.4 Running JSP Application

For execution of JSP pages following installations are needed.

- 1. Java Development Kit :** Download and install the latest version of JDK.
- 2. Tomcat WebServer :** The Tomcat is a web container provided by Apache Web server. Now a days XAMPP software package is installed for running JSP, Servlets and PHP programs.

Note that before running any JSP page it is essential to start the tomcat by double clicking the **startup.bat** file present in the tomcat directory.

Ex. 7.4.1 : Write a simple JSP page for displaying the message “This is my first JSP page!!!”

Sol. : We follow these steps to get the JSP page displayed -

Step 1 : First of all open some text editor like Notepad and type the following code.

hello.jsp

```

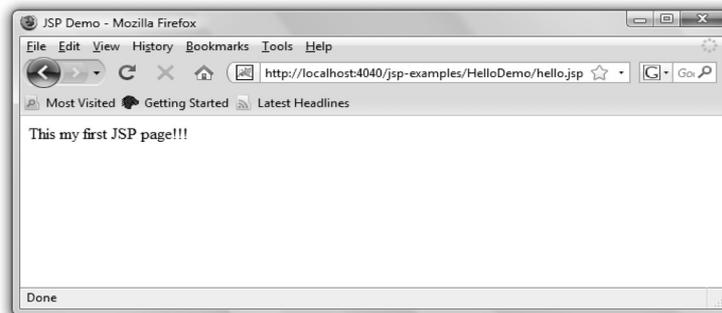
<%@ page language="java" contentType="text/html" %>
<%@ page import ="java.util.*" %>
<html>
<!-- This is basic JSP page -->
<title>JSP Demo </title>
<body>
<!-- Displaying the message on the browser --%>
    <% out.println("This my first JSP page!!!"); %>
</body>
</html>

```

Step 2 : Copy [this file](#) to webapps directory of your tomcat folder.

Step 3 : Start the Tomcat Web server.

Step 4 : Open some suitable web browser like Firefox or Internet Explorer. Type the path for the JSP document and the output will be as follows -



Script explanation

The first line is the **page directive** statement

```
<%@ page language="java" contentType="text/html" %>
```

using this line we are specifying that the language used in this script is Java and the contentType specifies the MIME type which is text/html.

Then on the next line we have imported the Java library package **java.util** using import statement -

```
<%@ page import ="java.util.*" %>
```

We can specify the JSP comments in two ways

```
<!--HTML style comment statement -->
```

```
<%-- JSP comment statement --%>
```

As JSP is commonly used along with HTML, it also supports the HTML comments.

For displaying the message on the web browser we have written the message using the Java statement

```
out.println("This my first JSP page!!!");
```

Normally any Java code must be written within `<%` and `%>`

Then all the corresponding tags are closed.

7.5 Declaration

- The JSP page that we write is turned into class definition. So when we declare a variable or method in JSP inside **Declaration Tag**.
- We can declare static member, instance variable and methods inside **Declaration Tag**.
- Syntax of Declaration Tag

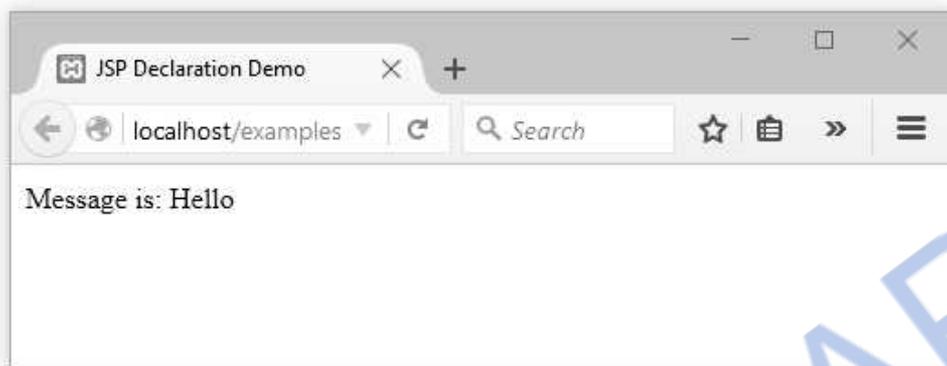
```
<%! Declaration code %>
```

- For example -

```

<html>
  <head>
    <title>JSP Declaration Demo</title>
  </head>
  <%!
    String msg = "Hello";
  %>
  <body>
    Message is:
    <% out.println(msg); %>
  </body>
</html>

```

Output**Explanation of Script :**

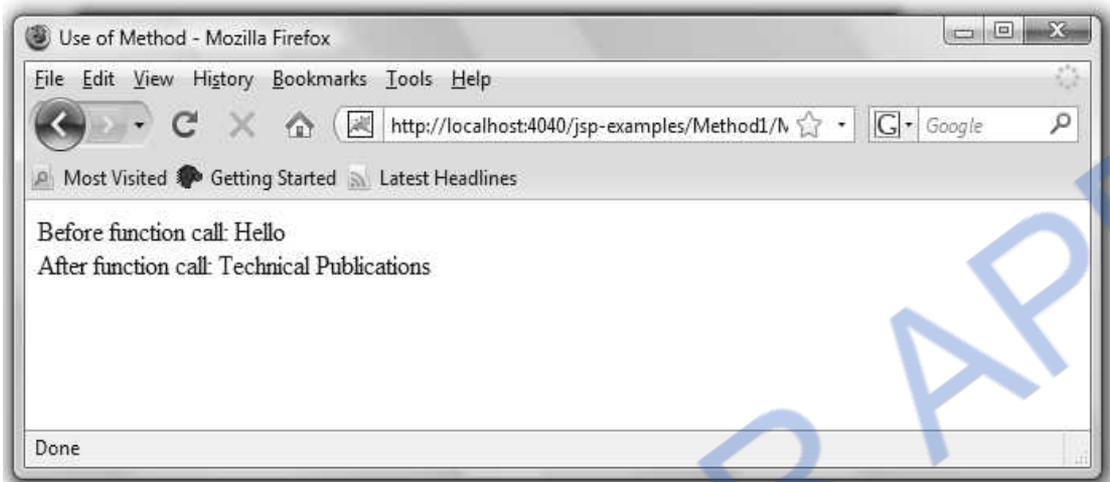
- (1) The above JSP code contains the declaration within `<%! %>` tag.
- (2) The variable **msg** of string type is declared and assigned with value as **“Hello”**

We can declare a function or a method in JSP just similar to variable. Following JSP example illustrates the use of function declaration and definition.

MethodDemo.jsp

```
<%@ page language="java" contentType="text/html" %>
<%!
    String msg="Hello";
%>
<%! public String MyFunction(String msg)
{
    return msg;
}
%>
<html>
<head>
    <title> Use of Method </title>
</head>
<body>
    <% out.print("Before function call: "+msg); %>
<br/>
    After function call: <%= MyFunction("Technical Publications") %>
</body>
</html>
```

Output

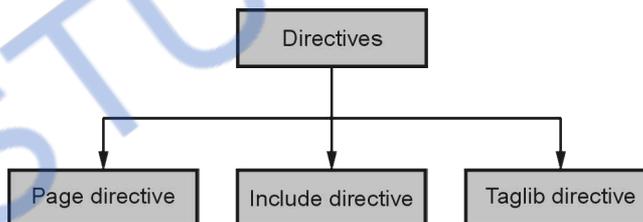


7.6 Directives

JSP directives control the processing of entire JSP Page. It gives direction to the server regarding the processing of a page.

Syntax

```
<%@ directive name [attribute name="value" attribute name="value" .....] %>
```



1. Page directive :

The page directive is used to provide the information about the page. Various attributes of page directive are -

- | | | |
|----------------|--------------------|------------------|
| i) import | ii) language | iii) contentType |
| iv) extends | v) session | vi) buffer |
| vii) autoFlush | viii) isThreadSafe | ix) info |
| x) errorPage | xi) isErrorPage | |

For example -

```
<%@page import="java.io.*" %>
<%@page language="java" %>
<%@page contentType="text/html" %>
```

2. Include directive :

Include directive is used to copy the content of one JSP page to another. It's like including the code of one file into another.

For example

```
<% @include file=calculate.jsp %>
```

3. Taglib directive :

This directive basically allows user to use custom tags in JSP. The custom tags are those which are created by user.

For example -

```
<%@ taglib uri="mytaglib" prefix="sampletag" %>
```

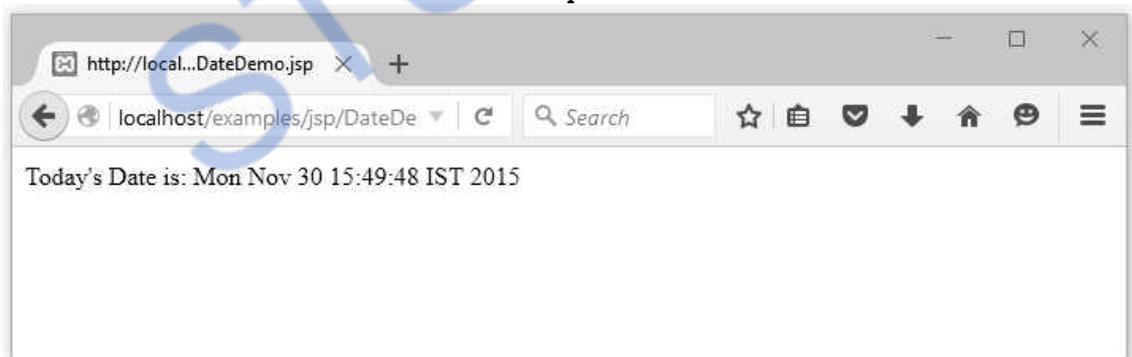
Ex. 7.6.1 : Write a JSP program to display current date and time.

Sol. :

DateDemo.jsp

```
<%@ page language="java" contentType="text/html" %>
<%@ page import = "java.util.*" %>
<html>
<body>Today's Date is:
    <%= new Date().toString() %>
</body>
</html>
```

Output



7.7 Comments in JSP

We can specify the JSP comments in two ways

```
<!--HTML style comment statement -->
```

```
<%-- JSP comment statement --%>
```

As JSP is commonly used along with HTML, it also supports the HTML comments.

7.8 Scripting Elements

There three types of scripting elements in JSP 1. Expression 2. Scriptlet 3. Declaration

1. Expression

The expression tag is used to represent the expression in JSP page. The syntax of writing expression is -

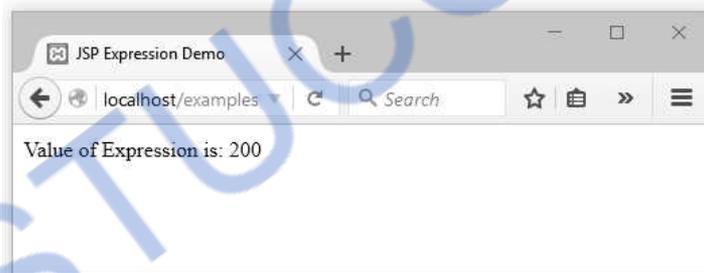
```
<%= Java Expression %>
```

For example -

```
<html>
  <head>
    <title>JSP Expression Demo</title>
  </head>

  <body>
    Value of Expression is :
    <%= (10*20) %>
  </body>
</html>
```

Output



2. Scriptlet

The code that appears between the <% and %> delimiters is called a *scriptlet*. Scriptlets are nothing but java code **enclosed within** <% and %> tags.

The everything other than a JSP statement in the JSP is called template text. For example

Templatetext.jsp

```
<%@ page language="java" contentType="text/html" %>
<html>
  <head>
    <title>Demo for Tempplate Text</title>
  </head>
  <body bgcolor="gray">
    <h1>Twinkle Twinkle little stars</h1>
    <h2>How I wonder what you are!!!</h2>
```

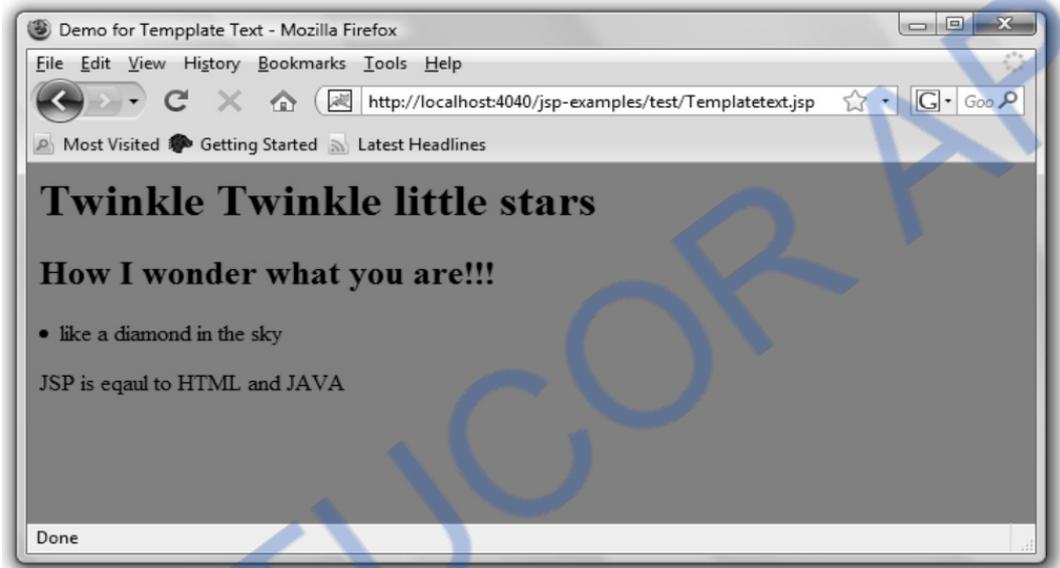
```

<li>like a diamond in the sky</li>
<p>
    <% out.print("JSP is eqaul to HTML and JAVA "); %>
</p>
</body>
</html>

```

In the above code the boldfaced code is a template text. The output for the above code will be

Output



3. Declaration

We can declare a variable or method in JSP inside declaration tag. We can declare static member, instance variable and methods inside declaration tag. Syntax of declaration tag -

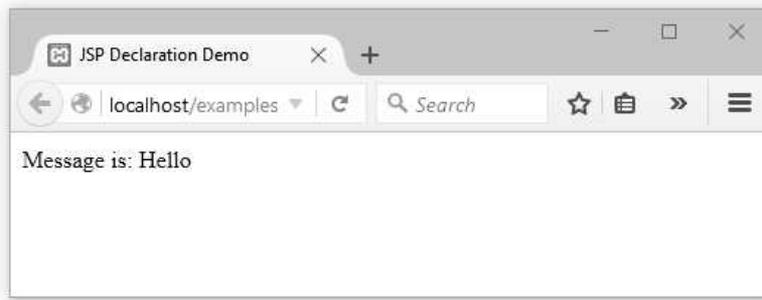
```
<%! Declaration code %>
```

For example -

```

<html>
<head>
    <title>JSP Declaration Demo</title>
</head>
<%!
    String msg = "Hello";
%>
<body>
    Message is:
    <% out.println(msg); %>
</body>
</html>

```

Output

The above JSP code contains the declaration within `<%! %>` tag.

Ex. 7.8.1 : Develop a simple JSP application that accepts user's age and displays a message according to the following rules :

- i) Age < 15 : Message : You are a kid!
- ii) Age between 16 and 40 : You are young!
- iii) Age above 40 : You are old!

Sol. : For developing this application we will create an HTML form for inputting the age value and then invoke the JSP page on which the appropriate messages are given based on the age.

Step 1 :**InputAge.html**

```
<html>
<head>
</head>
<body>
<form method="get" action="AgeMessage.jsp">
<strong>Enter Age:</strong>
<br/>
<input type="text" name="Age">
<br/>
<input type="submit" value="Enter">
</form>
</body>
</html>
```

Step 2 : The JSP code is as follows -

AgeMessage.jsp

```
<html>
<body>
<%@ page language="java" contentType="text/html" %>
<%@page import="java.io.*" %>
```

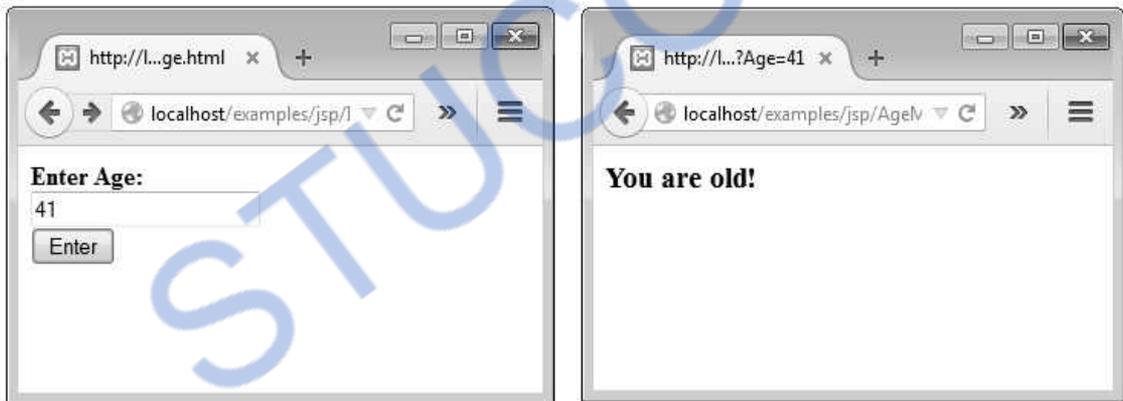
```

<%
String msg="";
String age;
int age_val;
age=request.getParameter("Age");
age_val=Integer.parseInt(age);
if(age_val<15)
    msg+="You are a kid!";
if((age_val>16)&&(age_val<40))
    msg+="You are young!";
if(age_val>40)
    msg+="You are old!";
out.println("<h3>" +msg+"</h3>");

%>
</body>
</html>

```

Step 3 : The output will be as follows -



7.9 Actions and Templates

- The standard actions are those actions that can be defined by the JSP specification itself.
- Following is a list of standard actions

Action Element	Description
<jsp:useBean>	This tag is used to instantiate the object of java bean.
<jsp:setProperty>	This tag is used to set the property value for the java bean.
<jsp:getProperty>	This tag is used to get the property value from the java bean.

<jsp:include>	This tag works as a subroutine. It includes the response from servlet or JSP when request is being processed.
<jsp:param>	For adding the specific parameter to the request this tag is used. Normally this tag is used inside <jsp:include> or <jsp:forward>
<jsp:forward>	This tag helps in forwarding the current request to servlet or to JSP page.
<jsp:plugin>	This tag is used to generate HTML code and to embed the applet into it.
<jsp:attribute>	This tag sets the value of the action attribute
<jsp:element>	This tag generated the XML elements dynamically.
<jsp:text>	This tag is used to handle template text. When JSP pages are written as XML documents then this tag is used.
<jsp:body>	This tag is used to set the body element.

Let us now discuss the standard actions by creating appropriate JSP pages.

Include action

Using the <jsp:include> action we can include different files in current jsp pages.

For example -

Include_action.jsp

```
<html>
  <head>
    <title> Include action Demo </title>
  </head>
  <body>
    <h1>This page contains include action</h1>
    <jsp:include page="test_include.html" />
    <jsp:include page="test_jsp.jsp" />
  </body>
</html>
```

Now the static html page can be

test_include.html

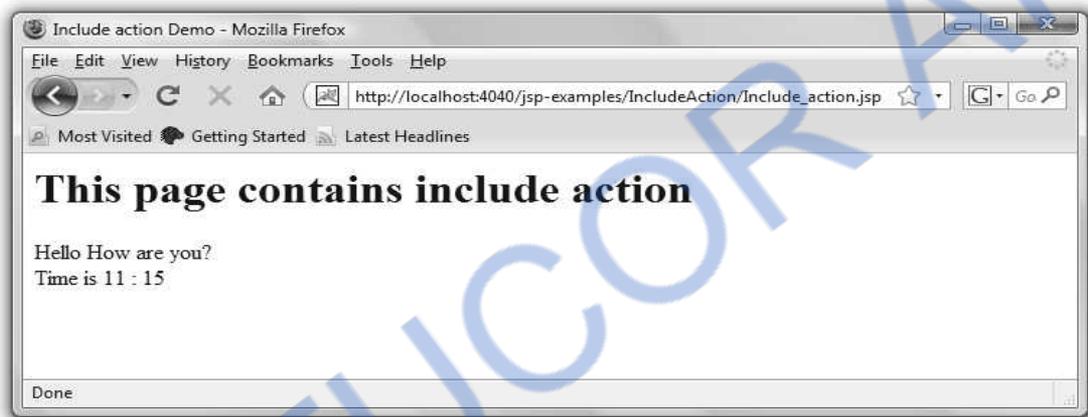
```
<html>
<body>
  Hello How are you?
</body>
</html>
```

Now the JSP page for displaying dynamic content is

test_jsp.jsp

```
<html>
  <body>
    <br>Time is <%= new java.util.Date().getHours()%>
    : <%= new java.util.Date().getMinutes()%>
  </body>
</html>
```

For getting the output, open the web browser and open our main JSP page **Include_action.jsp**, it is as follows-



Forward action

- Using forward action we can forward the current request to corresponding JSP page or servlet.
- Here is a simple demonstration in which we have created two web pages the first one is a main JSP page named as first.jsp in which we are using JSP action forward. And forwarding the current request to an HTML page which is named as Second.html

first.jsp

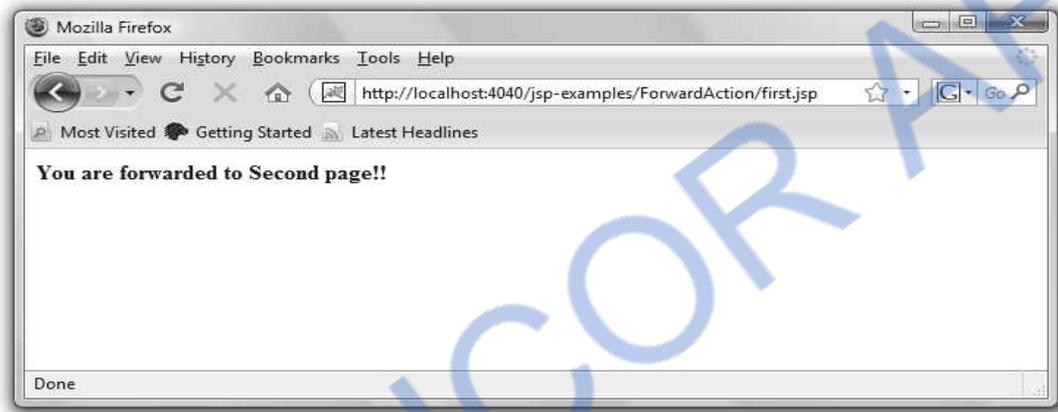
```
<html>
  <head>
    <title> Forward action Demo </title>
  </head>
  <body>
    <h1>This page contains forward action</h1>
    <jsp:forward page="Second.html" />
  </body>
</html>
```

Second.html

```
<html>
<body>
    <strong>You are forwarded to Second page!!</strong>
</body>
</html>
```

For getting the output open the web page and open the main JSP page that is **first.jsp** and you will get the following output -

Output



(3) Plugin Action

- The plugin action is used to embed an applet or a bean.
- In this section we will discuss how to embed an applet in the JSP page using **jsp:plugin**. We will follow these steps to use plugin action.

Step 1: Let us write an applet program as follows -

ColorDemo.java

```
import java.awt.*;
import java.applet.*;
public class ColorDemo extends Applet
{
    public void paint(Graphics g)
    {
        setBackground(Color.cyan);
        setForeground(Color.red);
        g.drawString("Its a colorful Applet",50,30);
        Color newColor=new Color(255,255,0);
        //creating red+green=yellow color
        g.setColor(newColor);
        g.drawString("Its a colorful Applet",50,70);
    }
}
```

```
}  
}
```

compile this code using the command

```
D:\> javac ColorDemo.java
```

Then corresponding class file i.e. ColorDemo.class gets generated. Now create a folder named **PluginAction** in your tomcat directory in the subfolder webapps/jsp-examples. Copy this class path in the **PluginAction** directory.

Step 2 : Now create a JSP page in **PluginAction** folder in which we are using the jsp action plugin. It is as follows -

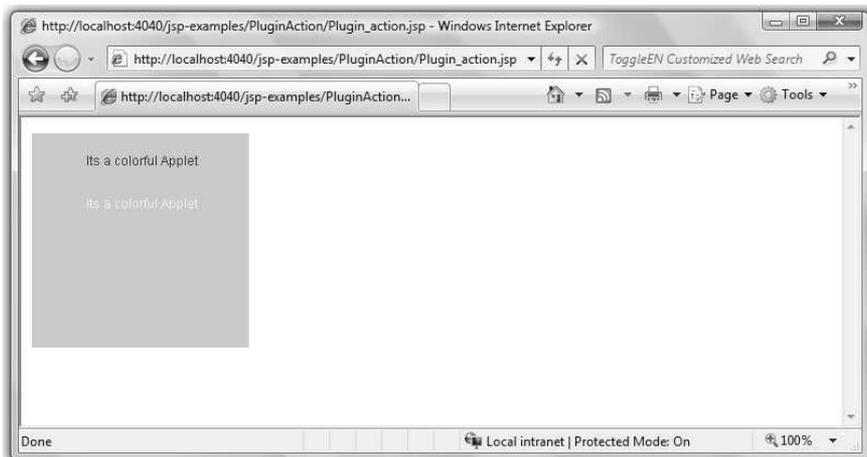
Plugin_action.jsp

```
<jsp:plugin type="applet" code="ColorDemo" codebase=  
"/jsp-examples/PluginAction" width = "190" height = "190">  
  <jsp:fallback>  
    <strong>Sorry!!! Applet can not be loaded.</strong>  
  </jsp:fallback>  
</jsp:plugin>
```

Note that in the above given JSP page we have used `<jsp:plugin>` where **type** is **applet** because we want to insert applet in JSP page. The type can also be **bean** for inserting a bean component. Then **code** denotes the name of the applet class. We have created applet class in step 1 and its name is **ColorDemo** hence we put the same name for the **code**. Then comes a **codebase** whose value is the directory name in which java class is located. Here the class file refers to the applet class file which we have created in Step 1.

Step 3 : For getting the output, open the web browser and open the jsp file Plugin_action.jsp and you can see that the desired applet is getting loaded -

Output



7.10 JavaBean Classes and JSP

- Java beans are reusable components. We can use simple Java bean in the JSP.
- Beans are used in the JSP pages as the instance of class. We must specify the scope of the bean in the JSP page. Here scope of the bean means the range time span of the bean for its existence in JSP.
- Bean is basically Java class with **getXX** and **setXX** methods

Characteristics for JavaBean Class

- (1) Java Bean class contains a default **no argument constructor**. This constructor must have the access specifier as **public**.
- (2) It should be serializable and should **implement serializable interface**.
- (3) It can have any number of properties which can be read or written. For these properties we write **getter and setter methods**.

JavaBean Properties

A JavaBean Property is a named attribute that can be accessed using the object.

The JavaBean properties can be accessed through two methods namely

- (1) **getPropertyName()**: This method is called **accessor** method. It returns the property value. This method is written in some special way. For example – If the property is **rollNo** then the method name for **rollNo** is **getRollNo()**
- (2) **setPropertyName()**: This is called **mutator** method. This method is generally used to assign the value to the properties. This method is written in some special way. For example – If the property is **rollNo** then method name will be **setRollNo()**

The read-only attribute is for **getPropertyName()** method and write-only attribute is for **setPropertyName()** method.

Example: Bean program using JSP for Employee Information.

Step 1: Create a Bean program in some package. I have created a folder(as a package) named **BeanProg**. The code for JavaBean class is written as follows –

Employee.java

```
package BeanProg;
public class Employee implements java.io.Serializable {
    private String empName=null;
    private int empNo=0;
    public Employee() { //no argument constructor
    }
    public String getEmpName() {
        return empName;
    }
}
```

```
}  
public int getEmpNo() {  
    return empNo;  
}  
public void setEmpName(String empName) {  
    this.empName=empName;  
}  
public void setEmpNo(int empNo) {  
    this.empNo=empNo;  
}  
}
```

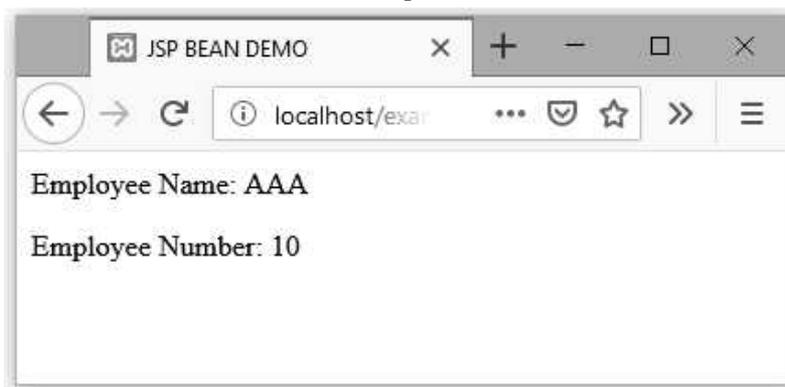
Step 2: The JSP code can be written as follows –

EmpBeanDemo.jsp

```
<html>  
  <head>  
    <title> JSP BEAN DEMO </title>  
  </head>  
  <body>  
    <jsp:useBean id="empID" class="BeanProg.Employee">  
      <jsp:setProperty name = "empID" property = "empName" value = "AAA"/>  
      <jsp:setProperty name = "empID" property = "empNo" value = '10'/>  
    </jsp:useBean>  
    <p>Employee Name:  
    <jsp:getProperty name = "empID" property = "empName"/>  
    </p>  
    <p>Employee Number:  
    <jsp:getProperty name = "empID" property = "empNo"/>  
    </p>  
  </body>  
</html>
```

Step 3: Start tomcat and open the JSP page in Web browser you will get the output as follows –

Output



Ex. 7.10.1 Write a client server JSP program to find the simple interest and display the result in the client.

AU : May-13, Marks 10

Sol. : This program is created using following steps.

Step 1 : Create an simple HTML form for inputting the values of P, N and R. The HTML document for this is as given below

Input.html

```
<html>
<head><title>Input Form</title></head>
<body>
Enter following values:<br/>
<form method="post" action="interest.jsp">
Amount:<input type="text" name="amount" value="" size="10"/>
Period:<input type="text" name="period" value="" size="3"/>
Rate of Interest:<input type="text" name="rate" value="" size="3"/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Step 2 : Create JSP page which acts as a client to receive the values for P,N and R. These values will be sent to the server Java program for calculating the simple interest. The calculated interest value is displayed by this same JSP page. The code for this is as follows -

Interest.jsp

```
<html>
<head>
<title> Simple Interest Calculation Demo </title>
</head>
<body>
<jsp:useBean id="bean_id" class="interestcalcDemo.InterestDemo" scope="session" />
<% String s_p=request.getParameter("amount");
int p=Integer.parseInt(s_p);
bean_id.setP(p);

String s_n=request.getParameter("period");
int n=Integer.parseInt(s_n);
bean_id.setN(n);

String s_r=request.getParameter("rate");
int r=Integer.parseInt(s_r);
bean_id.setR(r);
%>
```

```

</form>
Amount = <%= p%><br/>
Period = <%= n%><br/>
Rate = <%= r%><br/>
<strong>Interest is now <%= bean_id.getI() %><br/></strong>
</body>
</html>
    
```

Obtaining interest value from server

Step 3 : Following is a Java bean program which acts as a Server. It calculated the simple interest using the formula $(p*n*r)/100$. This value is then returned to the client JSP program created in Step 2. The code for server java program is as follows -

InterestDemo.java

```

package interestcalcDemo;

public class InterestDemo

{
    public int p,n,r;

    public InterestDemo()
    {
        p=0;n=0;r=0;
    }

    public int getI()
    {
        int i=(p*n*r)/100;
        return i;
    }

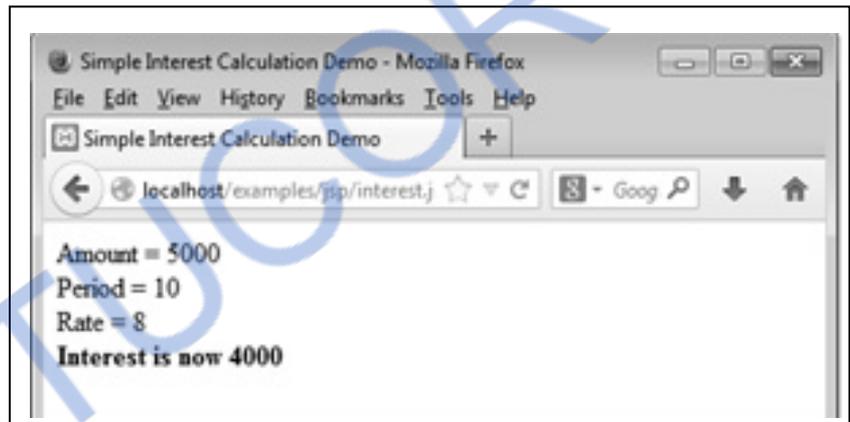
    public void setP(int p)
    {
        this.p=p;
    }

    public int getP()
    {
        return p;
    }

    public void setN(int n)
    {
        this.n=n;
    }

    public int getN()
    {
        return n;
    }
}
    
```

Output



```

public void setR(int r)
{
    this.r=r;
}
public int getR()
{
    return r;
}
}

```

7.11 Tag Libraries and Files

- In JSP the tag library is specified by JSTL.
- JSTL stands for **JSP Standard Tag Library**.
- **J2EE** is typically used for server side programming using Java and JSTL is a component of J2EE web application development platform.
- The JSTL makes use of JSP along with a tag library.
- This tag library is useful for performing some common tasks such as condition execution, loop execution, data procession and so on.
- JSTL allows the programmer to embed the logic in JSP page without using JAVA code.
- The tag library makes use of some standard set of tags. We will discuss the use of core tag in JSP

7.11.1 Core Tags

- The core tag is used nearly in all the web applications.
- This tag is useful for performing basic input and output, for looping operation or for evaluating expressions. Various tags used in core library are -

1. <c:out>
2. <c:set>
3. <c:if>
- 4.<c:choose>
- 5.<c:when>
6. <c:otherwise>
7. <c:forEach>
8. <c:forTokens>
9. <c:url>
- 10.<c:redirect>
- 11.<c:param>
- 12.<c:import>

Let us understand the use of core tag with the help of programming examples.

Use of <c:out>

First.jsp

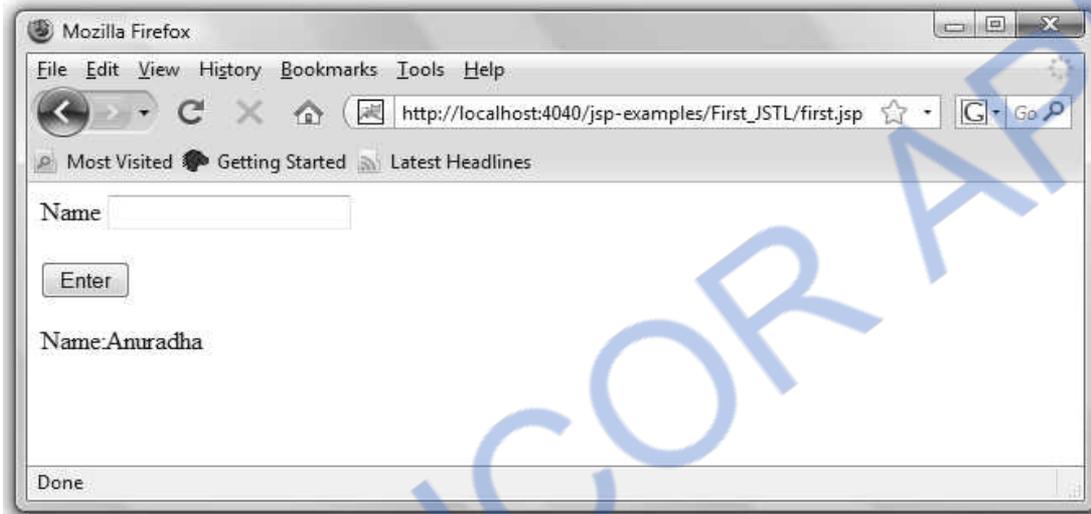
```

<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body>
<form method=post action="first.jsp">

```

```
Name <input type=text name="text1" value=""> <br> <br>
<input type=submit value="Enter">
</form>
Name:<c:out value="{param.text1}" /> <br>
</body>
</html>
```

Output



Script explanation

In above written JSP code

- (1) we have used **taglib** directive with prefix **c**.
- (2) Note that we must specify the **uri="http://java.sun.com/jsp/jstl/core"** for every core tag and prefix **"c"**.
- (3) We have used **<c:out>** for outputting the value written in the text box of the form.

Use of <cset>

Second.jsp

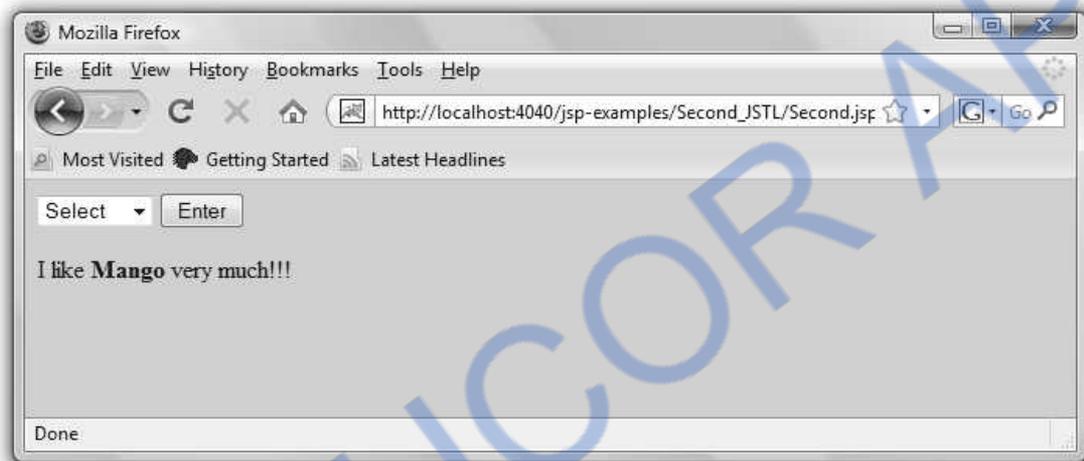
```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor=pink>
<form method=post action=Second.jsp>
  <select name="combo1">
    <option value="">Select
    <option value="Apple">Apple
    <option value="Mango">Mango
    <option value="Banana">Banana
    <option value="Guava">Guava
```

```

</select>
<input type=submit value="Enter">
</form>
<c:set var="fruit" value="{param.combo1}" />
I like <strong> <c:out value="{fruit}" /> </strong> very much!!!
<br>
</body>
</html>

```

Output



Use of <c:if>

Third.jsp

```

<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor=pink>
<form method=post action=Third.jsp>
  <select name="combo1">
    <option value="">Select
    <option value="Apple">Apple
    <option value="Mango">Mango
    <option value="Banana">Banana
    <option value="Guava">Guava
  </select>
  <input type=submit value="Enter">
</form>
<c:set var="fruit" value="{param.combo1}" />
<c:if test="{fruit} == 'Apple'" >
I like <strong> <c:out value="{fruit}" /> </strong> very much!!!
</c:if>

```

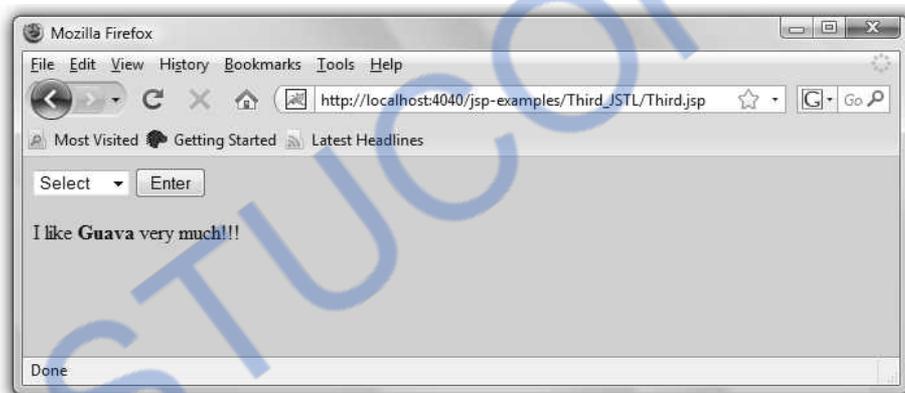
```
<c:if test="{fruit == 'Mango'}" >
I like <strong><c:out value="{fruit}" /></strong> very much!!!
</c:if>
```

```
<c:if test="{fruit == 'Banana'}" >
I like <strong><c:out value="{fruit}" /></strong> very much!!!
</c:if>
```

```
<c:if test="{fruit == 'Guava'}" >
I like <strong><c:out value="{fruit}" /></strong> very much!!!
</c:if>
```

```
<br>
</body>
</html>
```

Output



Use of <c:choose>, <c:when> <c:otherwise>

We can have select any item using <c:choose> tag. The <c:choose> contains the <c:when test> and <c:otherwise> tags for specifying the selection criteria. In the following example we will see how to make use of these tags. For a <c:when> tag the **test** is a standard variable being used.

Forth.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor=pink>
<form method=post action=Forth.jsp>
  <select name="combo1">
    <option value="">Select
    <option value="Apple">Apple
    <option value="Mango">Mango
```

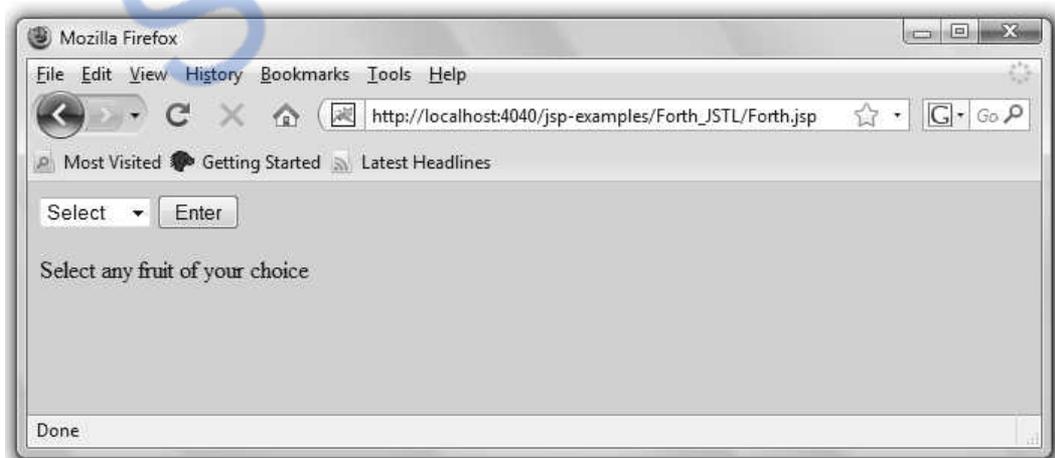
```
<option value="Banana">Banana
<option value="Guava">Guava
</select>
<input type="submit" value="Enter">
</form>
<c:set var="fruit" value="{param.combo1}" />
<c:choose>
  <c:when test="{fruit == 'Apple'}" >
    I like <strong><c:out value="{fruit}" /></strong> very much!!!
  </c:when>

  <c:when test="{fruit == 'Mango'}" >
    I like <strong><c:out value="{fruit}" /></strong> very much!!!
  </c:when>

  <c:when test="{fruit == 'Banana'}" >
    I like <strong><c:out value="{fruit}" /></strong> very much!!!
  </c:when>

  <c:when test="{fruit == 'Guava'}" >
    I like <strong><c:out value="{fruit}" /></strong> very much!!!
  </c:when>
  <c:otherwise>
    Select any fruit of your choice
  </c:otherwise>
</c:choose>
<br>
</body>
</html>
```

Output



Use of <c:forEach>

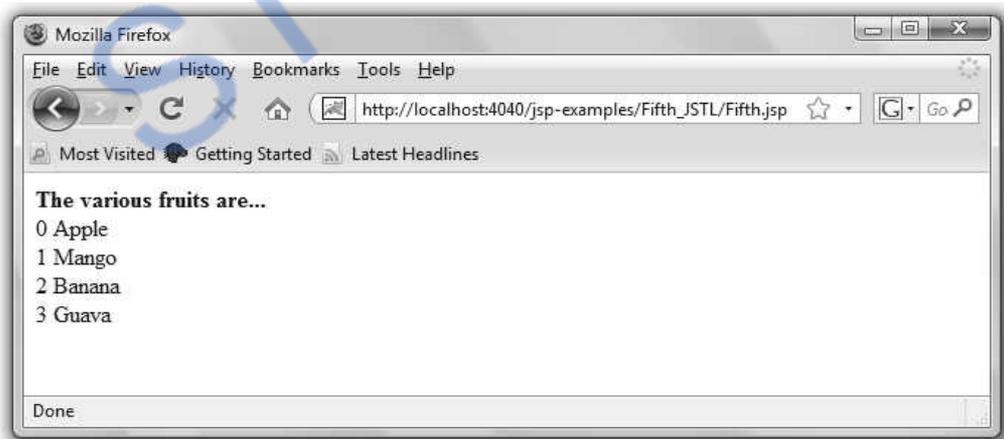
This tag is specially used to denote the looping condition. Following code shows its demonstration.

Fifth.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body>
<% pageContext.setAttribute("Fruits",new String[] {"Apple","Mango","Banana","Guava"} );%>
<b>The various fruits are...</b>
<c:forEach var="i" items="{Fruits}" varStatus="n">
<br/>
  <c:out value="{n.index}" />
  <c:out value="{n.current}" />
</c:forEach>
</body>
</html>
```

In above code we have used <c:forEach> tag in which **items** denote the current String, **var** is a symbolic name for array **varStatus** is the symbolic name for current status. Using **index** we can display the index of each item in an array. By default array starts from index 0 and **current** is used to display the current element.

The output will be

Output

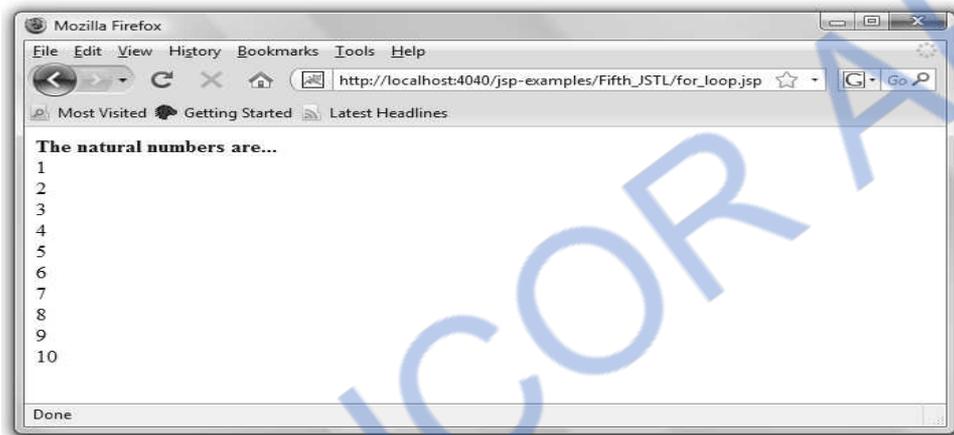
Let us see one more program which makes use of the tag <c:forEach>

for_loop.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<html>
  <body>
    <b>The natural numbers are...</b>
    <c:forEach var="i" begin="1" end="10">
      <br/>
      <c:out value="${i}" />
    </c:forEach>
  </body>
</html>
```

Output



Use of <c:url>

As the name suggests this tag is used to navigate to the desired page using the URL. Here is its demonstration -

urldemo.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body>
    <c:import url="MyPage.html" />
    <c:out value="Thank you for using this demo..." />
  </body>
</html>
```

MyPage.html

```
<html>
<body>
<h1>The visitors of this site are always wonderful persons!!</h1>
</body>
</html>
```

Output

To make use of `<c:redirect>` tag just make slight change in the above JSP page as follows –

Use of `<c:redirect>`

As the name suggests this tag is used to navigate to the desired page using the URL. Here is its demonstration -

urldemo.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body>
    <c:import url="MyPage.html" />
    <c:out value="Thank you for using this demo..." />
  </body>
</html>
```

MyPage.html

```
<html>
<body>
<h1>The visitors of this site are always wonderful persons!!</h1>
</body>
</html>
```

Output

To make use of `<c:redirect>` tag just make slight change in the above JSP page as follows -

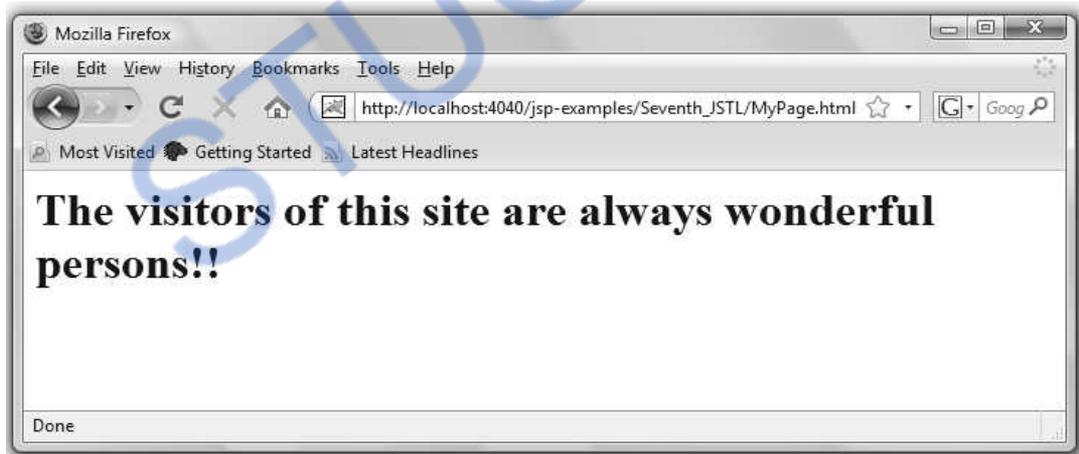
urldemo.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body>
    <c:redirect url="http://localhost:4040/jsp-examples/Seventh_JSTL/MyPage.html" />
    <c:out value="Thank you for using this demo..." />
  </body>
</html>
```

MyPage.html

```
<html>
  <body>
    <h1>The visitors of this site are always wonderful persons!!</h1>
  </body>
</html>
```

The **Output** itself is self explanatory. Note that the string specified using `<c:out>` tag i.e. "Thank you for using this demo..." will not be displayed at all. That means using `<c:redirect>` we get navigated to designated page from the source web page and contents of source web page will not be displayed. Here is your output



Use of `<c:param>`

This tag is useful to send some parameter to another page. We have written one main JSP page on which we have given a redirecting link to the NewPage.jsp and while navigating to this web page we have sent a parameter MyName by assigning some name value to it.

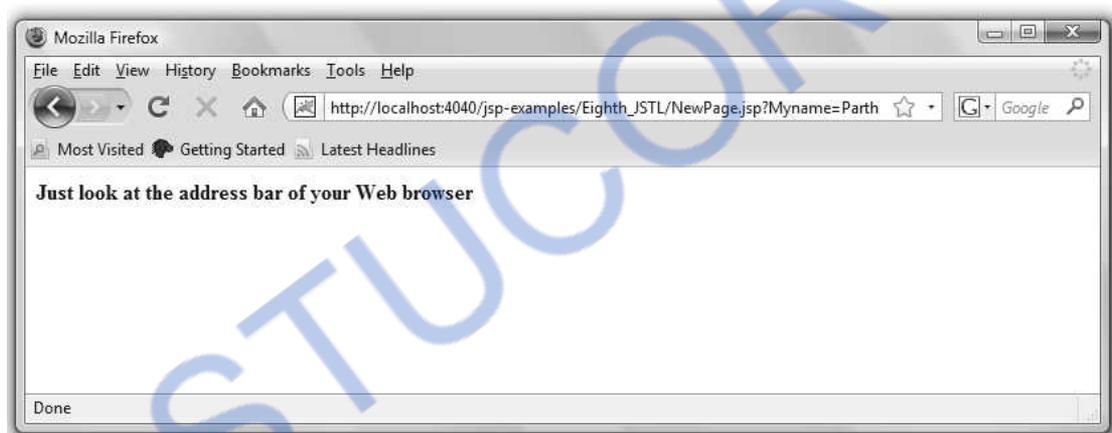
ParamDemo.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:redirect url="http://localhost:4040/jsp-examples/Eighth_JSTL/NewPage.jsp" >
<c:param name="MyName" value="Parth" />
</c:redirect>
```

NewPage.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body>
    <strong> Just look at the address bar of your Web browser </strong>
    <c:out value="{param.MyName}" />
  </body>
</html>
```

Output



Use of <c:import>

This tag is very much similar to <c:url>

ImportDemo.jsp

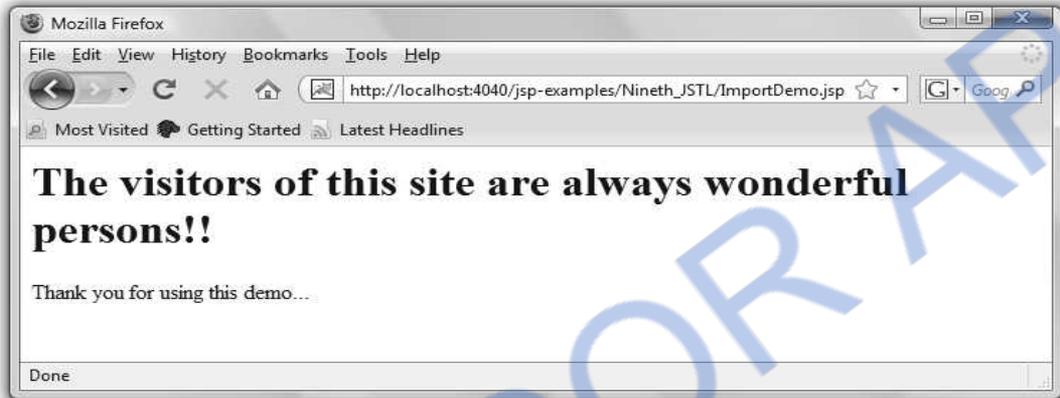
```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body>
  <c:import url="http://localhost:4040/jsp-examples/Nineth_JSTL/MyPage.html" />
  <c:out value="Thank you for using this demo..." />
</body>
</html>
```

MyPage.html

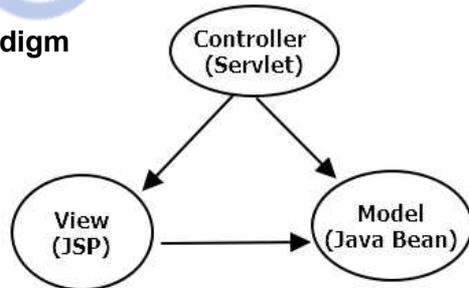
```

<html>
  <body>
    <h1>The visitors of this site are always wonderful persons!!</h1>
  </body>
</html>

```

Output**7.12 Support For Model View Controller Paradigm**

- The design model of JSP application is called MVC model.
- The MVC stands for Model-View-Controller.
- The basic idea in MVC design model is to separate out design logic into three parts - modelling, viewing and controlling.
- Any server application is classified in three parts such as business logic, presentation and request processing.
- The **business logic** means the coding logic applied for manipulation of application data.
- The **presentation** refers to the code written for look and feel of the web page. For example: background color, font style, font size, placing of controls such as text boxes, command buttons and so on.
- The **request processing** is nothing but a combination of business logic and presentation. The request processing is always done in order to generate the response.
- According to the MVC design model, the Model corresponds to business logic, View corresponds to presentation and Controller corresponds to request processing.

**Fig. 7.12.1 : Model View Controller**

Advantages of using MVC design Model

1. The use of MVC architecture allows the developer to keep the separation between business logic, presentation and request processing.
2. Due to this separation it becomes easy to make **changes** in presentation without disturbing the business logic. The changes in presentation are often required for accommodating the new presentation interfaces.
3. Each to maintain if the application is complex and large.
4. Navigation control is centralized.

University Question

1. Explain the model view controller architecture pattern in detail.

AU : May-12, Marks 8

7.13 Database and JSP

While accessing JSP database from JSP page we should have some database package installed. In this section we will consider the connectivity of JSP with the MYSQL database. We will follow following steps to get access the database using JSP page with some assumptions -

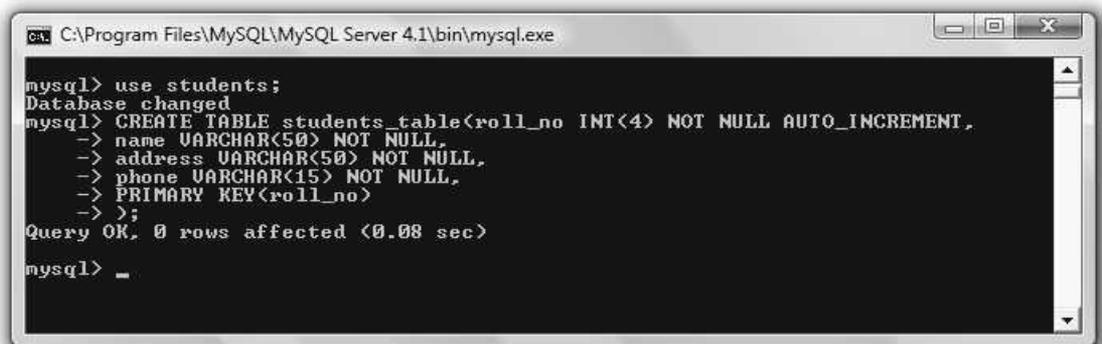
Assumptions

- TOMCAT web server is installed.
- MYSQL server is installed.
- JDK is installed.

Step 1 : Create a database named students in MYSQL using following command.

```
mysql>CREATE DATABASE students;
```

Then create a table named students_table in the students database as follows -



```
C:\Program Files\MySQL\MySQL Server 4.1\bin>mysql.exe
mysql> use students;
Database changed
mysql> CREATE TABLE students_table(roll_no INT(4) NOT NULL AUTO_INCREMENT,
-> name VARCHAR(50) NOT NULL,
-> address VARCHAR(50) NOT NULL,
-> phone VARCHAR(15) NOT NULL,
-> PRIMARY KEY(roll_no)
-> );
Query OK, 0 rows affected (0.08 sec)
mysql> _
```

If you would like to see the structure of created table then give following command -

```

mysql> describe students_table;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra           |
+-----+-----+-----+-----+-----+-----+
| roll_no | int(4) |      | PRI | NULL    | auto_increment |
| name    | varchar(50) |      |      |          |                 |
| address | varchar(50) |      |      |          |                 |
| phone   | varchar(15) |      |      |          |                 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.03 sec)

mysql>

```

Step 2 : For establishing the connectivity between JSP and MYSQL using JDBC driver what we need is to download the MYSQL JDBC connector. Download this connector from <https://dev.mysql.com/downloads/connector/j/3.1.html>

Copy the jar file named **mysql-connector-java-3.1.14-bin.jar** to the C:/your_tomcat_directoy/common/lib.

Step 3 : Start tomcat server.

Step 4 : The code for JSP is as follows -

StudentForm.jsp

```

<%@ page language="java" import="java.sql.*" %>
<%@ page import="java.io.*" %>
    <%
        Connection conn=null;
        ResultSet rs=null;
        Statement stmt=null;
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        conn=
        DriverManager.getConnection("jdbc:mysql://localhost:3306/students","root","system");
        out.write("Connected to mysql!!!");
        stmt=conn.createStatement();
        if(request.getParameter("action") != null)
        {
            String Studname=request.getParameter("Studname");
            String Studaddress=request.getParameter("Studaddress");
            String Studphone=request.getParameter("Studphone");
            stmt.executeUpdate("insert into students_table(name,address,phone) values('
            "+Studname+"','"+Studaddress+"','"+Studphone+"' )");
            rs=stmt.executeQuery("select * from students_table");
        }
    %>
    <html>
    <body>

```

Diagram annotations for the code above:

- 1: A bracket groups the initialization of `conn`, `rs`, and `stmt`.
- 2: An arrow points to the `Class.forName` call.
- 3: A callout box points to the `DriverManager.getConnection` call.
- 4: A bracket groups the retrieval of `Studname`, `Studaddress`, and `Studphone` parameters.
- 5: A callout box points to the `stmt.executeUpdate` call.

```

<center>
  <h2>Student List</h2>
  <table border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td> <b>Roll No</b></td>
    <td> <b>Student Name</b></td>
    <td> <b>Address</b></td>
    <td> <b>Phone</b></td>
  </tr>
  <%
  int num=1;
  while(rs.next()){
  %>
  <tr>
    <td> <%=num%></td>
    <td> <%=rs.getString("name")%></td>
    <td> <%=rs.getString("address")%></td>
    <td> <%=rs.getString("phone")%></td>
  </tr>
  <%
  num++;
  }
  rs.close();
  stmt.close();
  conn.close();
  %>
  </table>
</center>
</body>
</html>
<%}else{%>
<html>
<head>
<title>Student Registration Demo</title>
<script language="javascript">
  function validation(Form_obj)
  {
  if(Form_obj.Studname.value.length==0){
  alert("Please,fill up the remaining information!!");

  Form_obj.Studname.focus();
  return false;
  }
  if(Form_obj.Studaddress.value.length==0){
  alert("Please,fill up the remaining information!!");
  Form_obj.Studaddress.focus();
  return false;

```

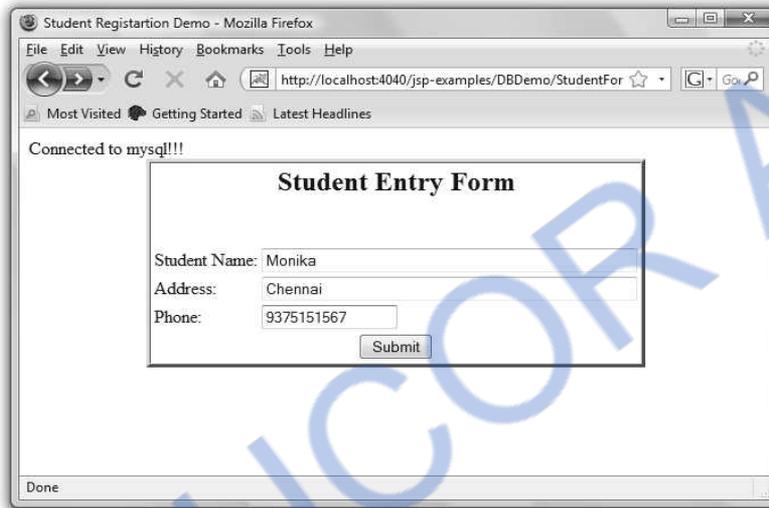
```

    }
    if(Form_obj.Studphone.value.length==0){
    alert("Please,fill up the remaining information!!");
    Form_obj.Studphone.focus();
    return false;
    }
    return true;
    }
    </script>
</head>
<body>
    <center>
    <form action="StudentForm.jsp" method="post"
    name="entry" onSubmit="return validation(this)">
        <input type="hidden" value="list" name="action">
        <table border="3" cellpadding="0" cellspacing="0">
            <tr>
            <td>
            <table>
                <tr>
                    <td colspan="2" align="center">
                        <h2>Student Entry Form</h2></td>
                </tr>
                <tr>
                    <td colspan="2">&nbsp;</td>
                </tr>
                <tr>
                    <td>Student Name:</td>
                    <td><input name="Studname"
                    type="text" size="50"></td>
                </tr>
                <tr>
                    <td>Address:</td>
                    <td><input name="Studaddress" type="text"
                    size="50"></td>
                </tr>
                <tr>
                    <td>Phone:</td>
                    <td><input name="Studphone"
                    type="text" size="15"></td>
                </tr>
                <tr>
                    <td colspan="2" align="center">
                        <input type="submit" value="Submit"></td>
                </tr>
            </table>
        </td>
        </tr>
        </table>
    </center>

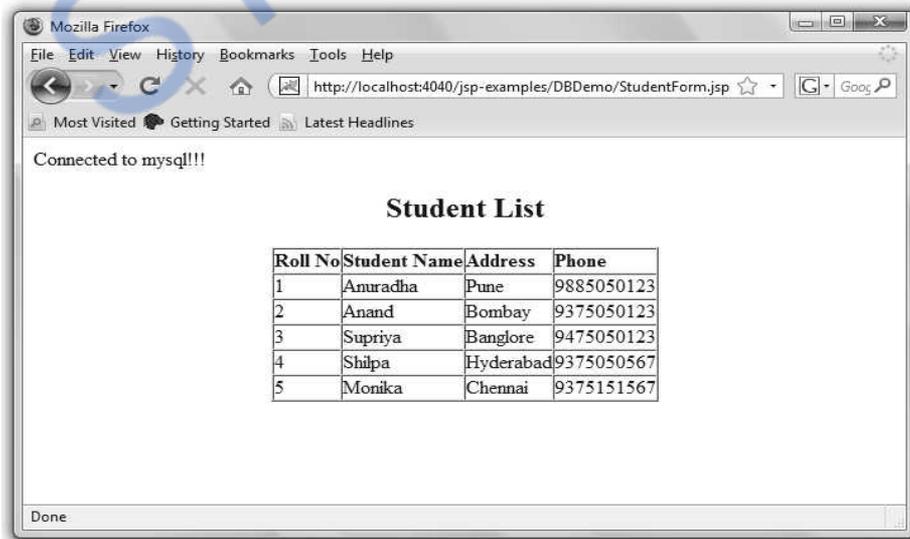
```

```
</td>
</tr>
</table>
</form>
</center>
</body>
</html>
<%}%>
```

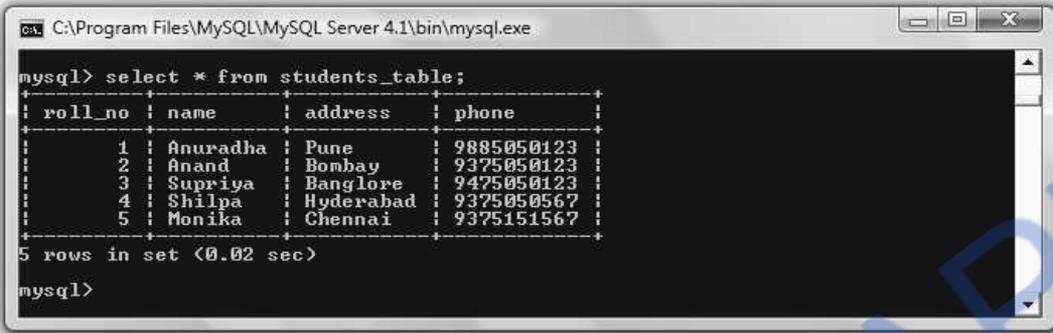
Output



Just click the Submit button and you will get the display of all the records from the students_table.



We can verify this table using backend database MYSQL as follows -



```

C:\Program Files\MySQL\MySQL Server 4.1\bin>mysql.exe
mysql> select * from students_table;
+----+-----+-----+-----+
| roll_no | name   | address | phone |
+----+-----+-----+-----+
| 1       | Anuradha | Pune   | 9885050123 |
| 2       | Anand   | Bombay | 9375050123 |
| 3       | Supriya | Banglore | 9475050123 |
| 4       | Shilpa  | Hyderabad | 9375050567 |
| 5       | Monika  | Chennai | 9375151567 |
+----+-----+-----+-----+
5 rows in set (0.02 sec)

mysql>

```

Script explanation

In above code, I have marked some steps by numbering them out. Let us understand these steps -

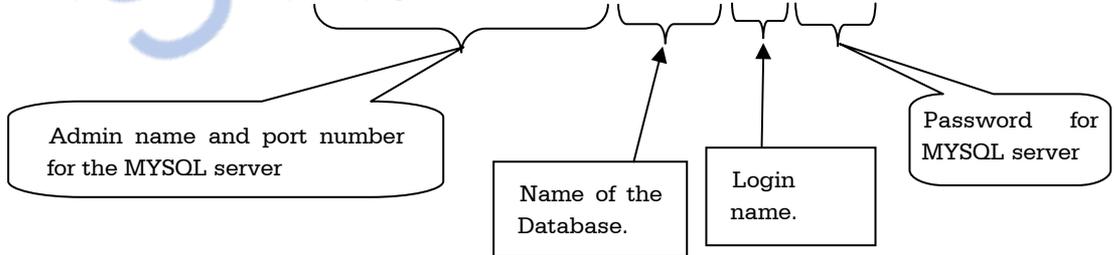
1. In this step required JDBC classes and interfaces such as **Connection**, **ResultSet**, **Statement** are initialised. These classes and interfaces are defined in the package **java.sql.***. That is why we must import them hence at the beginning of the program we did this job!

2. Here the object of the JDBC driver is instantiated. We have already downloaded the MYSQL JDBC connector (*as described in Step 2*). Hence the we specify it as

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

3. Then using **DriverManager** class we are getting connected to the MYSQL database. Note that this **DriverManager** class is defined in **java.sql.*** package. Using **getConnection** method we are actually getting connected to the MYSQL database.

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/students","root","system");
```



Note that the above given parameters in **getConnection** method may vary as per your database name, login name for your SQL server and the password. Then, a **createStatement** is used. It creates an object using which SQL statements can be sent to database.

4. Now we are retrieving the fields from the **form** (*this form is created on the same JSP page*) using **request.getParameter** method.

5. We then insert the values in the table using Insert query using **executeUpdate** method.

After inserting the values the table in the database can be displayed using the **Select *** query. For this purpose we have used **executeQuery** method.

- An HTML form is created in which various input fields such as students' name, address and phone number are given. User can enter the required values in these fields. These values can be further used by **Insert** query.

The above code also has a provision of validating the user input.

Ex. 7.13.1: Write a program that lists all books and the authors for those books from technical library database [Assume database schema].

Sol. : Assumption :

- The database named **books** is created.
- The table in the books database is **books_info** which is as given below -

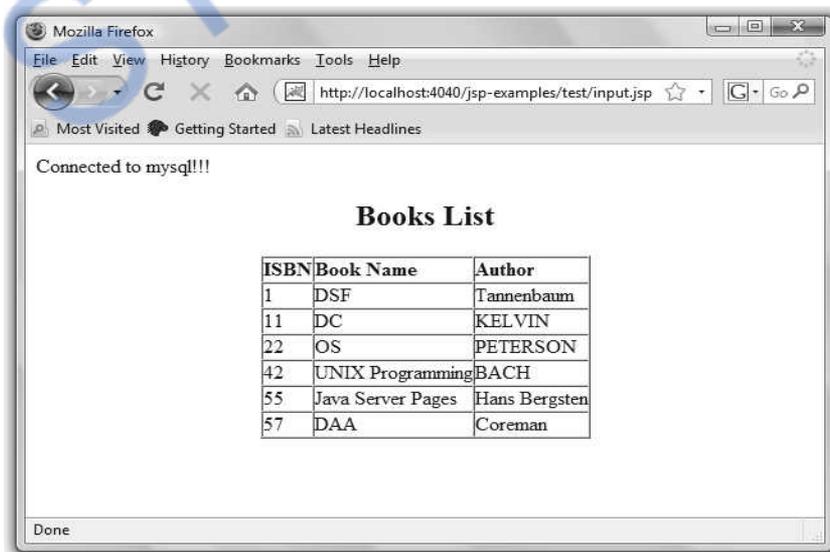
ISBN	name	author
1	DSF	Tannebaum
11	DC	Kelvin
22	OS	Peterson
42	Unix Programming	Bach
55	Java Server Pages	Hans Bergsten
57	DAA	Coreman

input.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.sql.Connection" %>
<%@ page import="java.sql.DriverManager" %>
<%@ page import="java.sql.ResultSet"%>
<%@ page import="java.sql.ResultSetMetaData" %>
<%@ page import="java.sql.Statement" %>
<%
    Connection conn=null;
    ResultSet rs=null;
    Statement stmt=null;
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/books","root","system");
    out.write("Connected to mysql!!!");
    stmt=conn.createStatement();
    rs=stmt.executeQuery("select * from books_info");
%>
<html>
```

```
<body>
<center>
<h2>Books List</h2>
<table border="1" cellspacing="0" cellpadding="0">
<tr>
<td><b>ISBN</b></td>
<td><b>Book Name</b></td>
<td><b>Author</b></td>
</tr>
<%
while(rs.next())
{
%>
<tr>
<td><%=rs.getString("ISBN")%></td>
<td><%=rs.getString("name")%></td>
<td><%=rs.getString("author")%></td>
</tr>
<%
}
rs.close();
stmt.close();
conn.close();
%>
</table>
</center>
</body>
</html>
```

Output



Two Marks Questions with Answers**Q.1 List any three advantages of Java Servlet over JSP.****AU : Dec.-08****Ans. :** Following are the three advantages of Java Servlet -

1. Java servlets contain pure Java code and there is no mixing of HTML tags.
2. Java servlets execute faster than JSP.
3. Servlet can act as a model-view and controller all at the same time. It can handle different dynamic pages.

Q.2 What are the advantages of JSP ?**OR Give the advantage of using JSP for server side programming .****AU : May-12****Ans. :** Following are some advantages of JSP –

1. JSP can be used along with servlets. Hence business logic for any application can be developed using JSP.
2. Dynamic contents can be handled using JSP because JSP allows scripting and element based programming.
3. JSP allows creating and using our own custom tag libraries. Hence any application specific requirements can be satisfied using custom tag libraries. This helps the developer to develop any kind of application.
4. JSP is a specification and not a product. Hence developers can develop variety of applications and add up to performance and quality of software products. Due to this many companies are ready to invest in JSP technology.

Q.3 List the scripting components of JSP.**AU : May-10****Ans. :** There are two types of scripting elements- JSP element and template text. The template text can be scripting code such as HTML, WML, XML or simple plain text. Various JSP elements can be action tags, custom tags, JSTL elements. The JSP elements are responsible for generating dynamic contents.**Q.4 What is MVC ?****Ans. :** MVC stands for Model View Controller. The basic idea in MVC design model is to separate out design logic into three parts - modelling, viewing and controlling.**Q.5 What is the advantage of using MVC model ?****Ans. :** The use of MVC architecture allows the developer to keep the separation between business logic, presentation and request processing. Due to this separation it becomes easy to

make changes in presentation without disturbing the business logic. The changes in presentation are often required for accommodating the new presentation interfaces.

Q.6 What are the types of directive in JSP ?

AU : May-08

Ans. : There are three types of directives in JSP.

1. page directive : Using page directive we can specify the language used by the JSP. For example : `<%@page language="java"%>`
2. include directive : This directive is used to include a file in the JSP page. For example : `<%@include file="/dingdong.jsp"%>`
3. taglib directive : The taglib is used to use the custom tags in the JSP pages. For example : `<%@taglib uri="/WEB-INF/Mytag.tld" prefix="my" %>`

Q.7 What are the standard action elements in JSP ?

Ans. : The standard actions are those actions that can be defined by the JSP specification itself.

For example –

Action Element	Description
<code><jsp:useBean></code>	This tag is used to instantiate the object of java bean.
<code><jsp:setProperty></code>	This tag is used to set the property value for the java bean.
<code><jsp:getProperty></code>	This tag is used to get the property value from the java bean.
<code><jsp:include></code>	This tag works as a subroutine. It includes the response from servlet or JSP when request is being processed.

Q.8 What is the use of ‘param’ variable in JSP ?

AU : Dec.-09

Ans. : For adding the specific parameter to the request the param tag is used. Normally this tag is used inside `<jsp:include>` or `<jsp:forward>`

Q.9 Explain in brief about Java scriplet.

AU : May-11

Ans. : Java scriplets allow the developer to embed the Java code in JSP page. The syntax of scriplet can be -

```
<% any Java code %>
```

The tags `<%` and `%>` are used for embedding the Java code.



Q.10 Write about javax.servlet.jsp.tagext package.

Ans. : It supports the classes and interfaces that are required for Java Server Pages tags libraries. Various classes and interfaces from the tag library are

Classes	Description
TagSupport	When a tag has to be implemented then we can extend the new tag handler class using this base class.
TagInfo	It provide the tag information for the tag in a tag library.
BodyTagSupport	When we want to create tag handlers for defining the body tag then this base class is used.

Interface	Description
Tag	This is the simple interface for defining the tag handler which does not want to manipulate its body.

The JSP tag library contains -

1. Various tag handler classes handling the request.
2. Tag library descriptors in .tld files.
3. Additional classes and interfaces.

Q.11 List methods in Request object.

Ans. : Various methods that are defined by the Request object are enlisted below -

Methods	Meaning
getAttribute(String att)	It returns the value of the attribute.
getAttributeNames()	For retrieving the names of the all attributes that are associated with the current session, this method is used.
getCookies()	It returns all the cookies sent by the client during the request.
getHeader(String header)	It returns the value of the requested header.
getHeaderNames()	It returns the header named of the requested header.
getMethod()	It returns the HTTP methods mentioned during the request. The HTTP methods are GET and POST.
getParameter(String param)	It returns the value of requested parameter.

getParameterNames()	It returns the names of all the parameters that are associated with the current session.
getQueryString	For returning the query string used during the request, this method is invoked.
setAttribute	For setting the object to the named attribute this method is used.

Q.12 Write two differences between JSP and servlet.

AU: May-13

Ans. :

Sr. No.	JSP	Servlet
1.	JSP is an HTML in which the Java code is embedded.	Servlets are basically Java programs in which the HTML is embedded.
2.	JSP is a scripting language which generates the dynamic contents.	Servlets are Java programs that are compiled to generate dynamic web contents.
3.	In Model View Controller architecture(MVC) the JSP acts as a View.	In Model View Controller architecture(MVC) the servlet acts as a controller.
4.	JSP makes use of custom tags that can call the Java beans directly.	Servlet does not support for the custom tags.

Q.13 What are the advantages of JSP over Servlets ?

AU : Dec.-18

- Ans. :**
- i) JSP can build custom tags.
 - ii) JSP can call Java beans.

□□□

Notes

STUCOR APP

Unit IV

8	Representing Web Data:XML
----------	----------------------------------

Syllabus

H Representing Web Data : XML-Documents and Vocabularies-Versions and Declaration-Namespaces- DOM based XML processing Event-oriented Parsing : SAX-Transforming XML Documents-Selecting XML Data : XPATH-Template based Transformations : XSLT-Displaying XML Documents in Browsers.

Contents

8.1	Introduction	
8.2	Documents and Vocabularies	
8.3	XML Versions and XML Declaration	
8.4	Namespace.....	May-10, 12 Marks 8
8.5	Document Type Definition (DTD).....	May-11, Marks 10
8.6	Introduction to DOM and SAX.....	Dec.-15, May-16, 17, Marks 8
8.7	DOM based XML Processing	
8.8	Event-Oriented Parsing : SAX.....	Dec.-15,May-16, 17, Marks 8
8.9	XSLT : Displaying XML Documents in Browsers	
8.10	Displaying XML Documents in Browser using CSS	
	Two Marks Questions with Answers	

8.1 Introduction

- XML stands for eXtensible Markup Language.
- This scripting language is similar to HTML. That means, this scripting language contains various tags. But these tags are not predefined tags, in-fact user can define his own tags.
- Thus HTML is designed for **representation of data** on the web page whereas the XML is designed for **transport or to store data**.

Uses of XML

1. XML is used to display the meta contents i.e. XML describes the content of the document.
2. XML is useful in exchanging data between the applications.
3. The data can be extracted from database and can be used in more than one application. Different applications can perform different tasks on this data.

Advantages of XML

1. XML document is human readable and we can edit any XML document in simple text editors.
2. The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl.
3. Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.

Goals of XML

Following are the goals of XML -

1. User must be able to define and use his own tag. This allows us to restrict the use of the set of tags defined by proprietary vendors.
2. Allow user to build his own tag library based on his web requirement.
3. Allow user to define the formatting rules for the user defined tags.
4. XML must support for storage or transport of data.

Features of XML

Following are some features which are most suitable for creating web related applications.

- XML is EXtesible Markup Language intended for transport or storage of the data.
- The most important feature of XML is that user is able to **define and use his own tag**.
- XML contains only data and does not contain any formatting information. Hence document developers **can decide how to display the data**.

- XML permits the document writer to create the tags for any type of information. Due to this virtually any kind of information can be such as mathematical formulae, chemical structures, or some other kind of data can be described using XML.
- Searching sorting, rendering or manipulating the XML document is possible using Extended Stylesheet Language (XSL).
- The XML document can be validated using some external tools.
- Some commonly used web browsers like Internet Explorer and Firefox Mozilla provide support to the tags in XML. Hence XML is not at all vendor specific or browser specific.

Difference between XML and HTML

Sr. No.	HTML	XML
1.	HTML stands for Hypertext Markup Language.	XML stands for eXtensible Markup Language.
2.	HTML is designed to display data with the focus on look and feel of data.	XML is used to transport and store data, with focus on what data is.
3.	HTML is case insensitive.	XML is case sensitive.
4.	HTML has predefined tags.	XML has custom tags can be defined and the tags are invented by the author of the XML document.
5.	As HTML is for displaying the data it is static.	As XML is for carrying the data it is dynamic.
6.	It can not preserve white space.	It can preserve the white space.

8.2 Documents and Vocabularies

8.2.1 Elements and Attributes

- Various building blocks of XML are -

1. Elements

The basic entity is **element**. The elements are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

2. Attribute

The attributes are generally used to specify the values of the element. These are specified within the double quotes.

For example -

```
<flag type="True">
```

The type attribute of the element flag is having the value True.

3. CDATA

CDATA stands for Character Data. This character data will be parsed by the parser.

4. PCDATA

It stands for Parsed Character Data (i.e. text).

8.2.1.1 Declaring Elements

- The element type data can be declared using the **ELEMENT**.
- The syntax for defining the ELEMENT is,

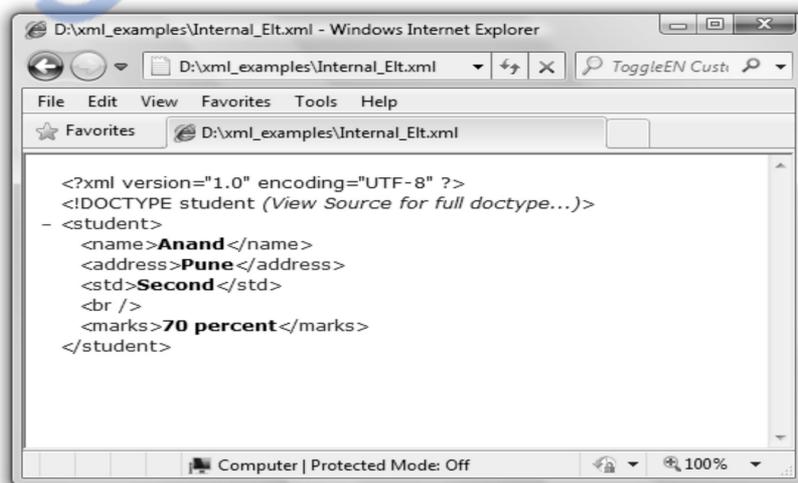
```
<!ELEMENTelt_name (elt1,elt2,..)>
```

XML Document [Internal_Elt.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,std,br,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENTstd (#PCDATA)>
<!ELEMENTbr EMPTY>
<!ELEMENT marks (#PCDATA)>
]>
<student>
  <name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <br />
  <marks>70 percent</marks>
</student>
```

Here we are defining
the DTD internally

Output



- The ELEMENT must be defined in the beginning of the XML file. It must be enclosed within `<!DOCTYPEroot_element>`
- There are various ways by which the actual element can be **defined**. It is enlisted in the following table -

Category	Syntax	Description
Empty	<code><!ELEMENT elt_nameEMPTY></code>	Empty elements can be declared.
Character data	<code><!ELEMENT elt_name (#PCDATA)></code>	Character data can be declared.
Any content	<code><!ELEMENT elt_nameANY></code>	Any combination of data can be declared.
Sequence	<code><!ELEMENTelt_name(elt1,elt2,...)></code>	Sequence of elements that must appear in specific order.

- Sometimes we can define the ELEMENT along with the iterator characters. The iterator characters are the special character which are associated with some meaning.
- Some commonly used iterator characters are as given below -

Iterator character	Meaning	Example
?	The preceding choice appears zero or once.	<code><!ELEMENT(marks?)></code> The child element marks can appear for zero or one time.
*	The preceding choice appears for any number of time(zero or more times).	<code><!ELEMENT (person*)></code> The person can appear for any number of times.
+	The preceding choice appears for at least once (one or more times).	<code><!ELEMENT (person+)></code> The person can appear for at least once.

8.2.1.2 Declaring Attributes

- The attribute type declaration is done using the string ATTLIST.
- The attribute type specifies the type of data which can be used for specifying the attribute. The syntax for specifying the attribute declaration is as follows -
`<!ATTLISTelement_nameattribute_nameattribute_typedefault_value>`

- For example

```
<!ATTLIST address phone CDATA #REQUIRED>
```

↑ Element_name
↑ Attribute_name
↑ Attribute_type
↑ Default_value

- The XML document containing attributes is as given below -

XML Document [Internal_Attr.xml]

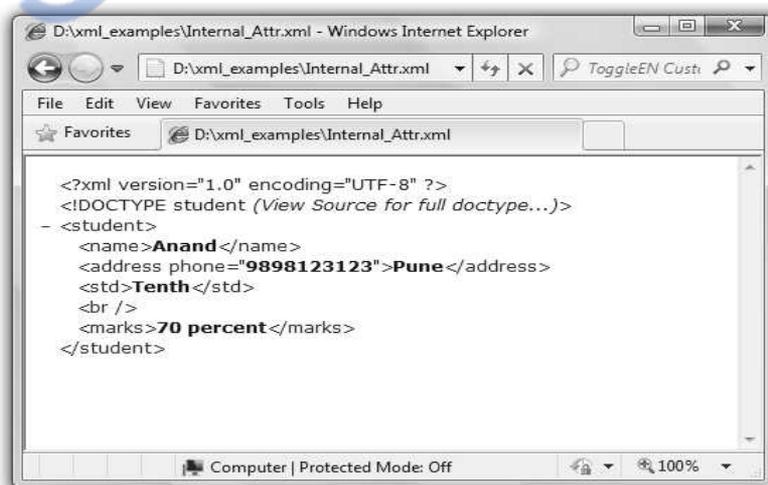
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENTstd (#PCDATA)>
<!ELEMENT marks (#PCDATA)>

<!ATTLIST address phone CDATA #REQUIRED>

]>
<student>
  <name>Anand</name>
  <address phone="9898123123">Pune</address>
  <std>Tenth</std>
  <br />
  <marks>70 percent</marks>
</student>
```

Using ATTLIST the attribute can be defined.

Output



- Following are some **attribute type** –

Attribute_Type	Syntax	Description
Character data	CDATA	Attribute value is character data.
Enumerated	(str1 str2 ...)	Attribute value must be one from an enumerated list.
Identifier	ID	Attribute value is a unique id.
Identifier reference	IDREF	Attribute value is the id of another element.
Identifier reference list	IDREFS	Attribute value is a list of other ids.
Name token	NMTOKEN	Attribute value is a valid XML name.
Name token list	NMTOKENS	Attribute value is a list of valid XML names.

The default_values are -

Default_Value	Meaning
Value	Default value can be defined. For example - <!ATTLIST Index flag="0"> In XML document <Index flag="1" /> If the flag is not assigned with any value then by default it will be 0.
#REQUIRED	Required attribute forces the attribute to be present even if there is no default value.
#IMPLIED	The default value is not provided and attribute can be optionally used.
#FIXED value	The attribute value is fixed and cannot be changed.

8.2.1.3 Declaring Entities

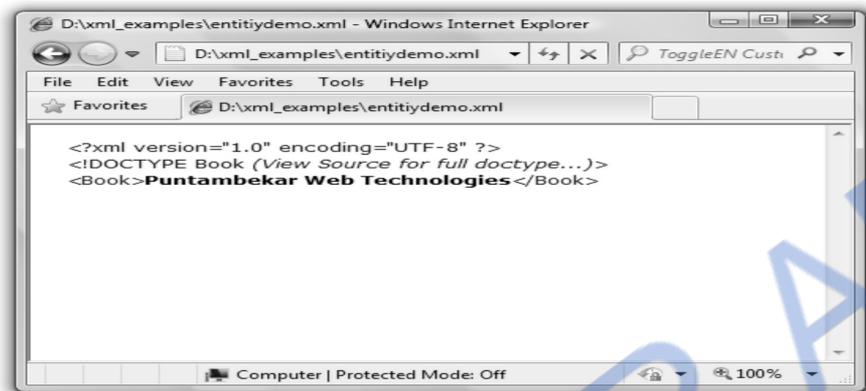
- The attribute type declaration is done using the string **ENTITY**.
- Inside the XML file at the beginning the entity can be defined using the <!DOCTYPE ...> and with the help of string ENTITY. Following XML document demonstrates it -

XML Document [EntityDemo.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Book[
<!ENTITY writer "Puntambekar">
<!ENTITY nbsp "&#160;">
<!ENTITY title "Web Technologies">
```

|>

<Book>&writer;&nbsp;&title;</Book>

Output

Note that we can define the predefined entities ** ** along with the other entity declaration.

The entity can be used using following syntax

&name_of_entity;

In above script we have used two entities namely **writer** and **title**. For displaying the space within these two entities we have used an entity **nbsp** and for defining it we have used **#160**.

University Question

1. What are the features of elements and attributes in XML ?

8.2.2 Rules

- **Rules that must be followed while writing XML -**

Here are some rules that must be followed while writing the XML programs -

1. XML is a case sensitive. For example -

```
<hobby>I like drawing</Hobby> or
<Hobby>I like drawing</hobby>
```

is not allowed because the words **hobby** and **Hobby** are treated differently.

2. In XML each start tag must have matching end tag. For example-

```
<staring> XML is funny to write
<simple> It is simple to implement </simple>
```

That means a closing tag is a must.

3. The elements in XML must be properly nested. For example-

```
<one><two>Hello how are you?</one></two> is wrong but
<one><two>Hello how are you?</two></one> is correct
```

8.3 XML Versions and XML Declaration

- Every XML document normally begin with following code

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

This indicates that the version is 1.0, and the encoding is **UTF-8**

- XML was developed under the World Wide Web Consortium as a language for **representing documents**. The XML is developed in 1996 and the first working draft was published in this year. The XML 1.0 was officially adopted as a W3C recommendation in 1998. According to this recommendation every XML document must begin with a special tag called **XML declaration** and this declaration can be

```
<?xml version "1.0">
```

- There is a potential impact on the character set of an XML document. Hence it is necessary to specify the character encoding that the XML document uses. There are two types of encodings UTF-16 or UTF-8.
- The default encoding of XML document is UTF-16 if the document begins with the two byte character 0xfeff and if document begins with any other character then the encoding should be UTF-8. All the software that read XML documents must support these encoding.
- If other than these default encoding is used then one must specify the encoding using the encoding declarations -

```
<encoding="ISO-8859-1">
```

Specification of such encoding is called **encoding declarations**.

8.4 Namespace

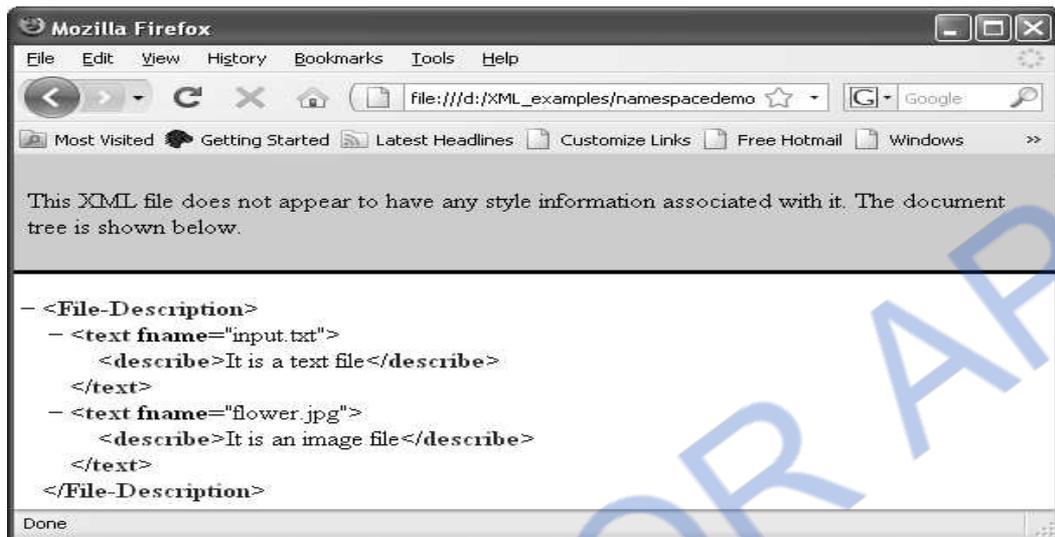
AU : May-10,12, Marks 8

- Sometimes we need to create two different elements by the same name. The xml document allows us to create **different elements** which are having the **common name**. This technique is known as **namespace**.
- In some web documents it becomes necessary to have the same name for two different elements. Here different elements mean the elements which are intended for different purposes. In such a case support for namespace technique is very much helpful.
- **For example :** Consider the following xml document -

namespacedemo.xml

```
<File-Description>
  <text fname="input.txt">
    <describe>It is a text file</describe>
  </text>
  <text fname="flower.jpg">
    <describe>It is an image file</describe>
  </text>
</File-Description>
```

The above document does not produce any error although the element **text** is used for two different attribute values. The output will be,



University Questions

1. Why should namespaces be used in XML ? Explain with an example.
2. Explain the role of XML name space with examples.

AU : May-10, Marks 8

AU : May-12, Marks 8

8.5 Document Type Definition (DTD)

AU : May-11, Marks 10

- The document type definition is used to define the basic building block of any xml document.
- Using DTD we can specify the various elements types, attributes and their relationship with one another.
- Basically DTD is used to specify the set of rules for structuring data in any XML file.
- **For example :** If we want to put some information about some students in XML file then, generally we use tag **student** followed by his/her **name, address, standard and marks**. That means we are actually specifying the manner by which the information should be arranged in the XML file. And for this purpose the Document Type Definition is used.
- Various building blocks of XML are –

1. Elements

The basic entity is **element**. The elements are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

2. Attribute

The attributes are generally used to specify the values of the element. These are specified within the double quotes.

For example -

```
<flag type="True">
```

The type attribute of the element flag is having the value *True*.

3. CDATA

CDATA stands for character data. This character data will be parsed by the parser.

4. PCDATA

It stands for Parsed Character **Data** (i.e. text). Any parsable character data should not contain the markup characters. The markup characters are < or > or &. If we want to use less than, greater than or ampersand characters then make use of < , > or &

- There are two ways by which DTD can be defined.

Internal DTD

Consider following xml document –

XML Document [DTDDemo1.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
]>
<student>
<name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <marks>70 percent</marks>
</student>
```

Here we are defining the DTD internally

Output

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <student>
  <name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <marks>70 percent</marks>
</student>
```

External DTD

In this type, an external DTD file is created and its name must be specified in the corresponding XML file. Following XML document illustrates the use of external DTD.

Step 1 : Creation of DTD file [*student.dtd*]

Open some suitable text editor or a notepad. Type following code into it -

```
<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
```

Now save this file as *student.dtd*

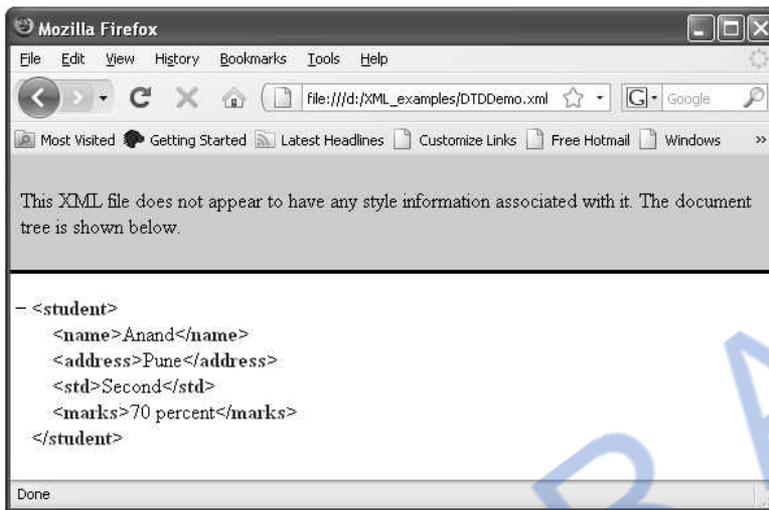
Step 2 : Creation of XML document -**XML Document [*DTDDemo.xml*]**

Now create a XML document as follows -

```
<?xml version="1.0"?>
<!DOCTYPE student SYSTEM "student.dtd">
<student>
  <name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <marks>70 percent</marks>
</student>
```

The external DTD file is created

Step 3 : Using some web browser open the XML document.

Output

Ex. 8.5.1 : Create a DTD for a catalog of four stroke motorbikes, where each motor bike has the following child elements -make, model, year, color, engine, chassis number and accessories. The engine element has the child elements engine number, number of cylinders, type of fuel. The accessories element has the attributes like disc brake, auto-start and radio, each of which is required and has the possible values yes and no.

Sol. :

automobile.dtd

```

<!ELEMENT Catalog (MotorBike)* >
<!ELEMENT MotorBike (make, model, year,color, engine,chasis-num,accessories ) >
<!ELEMENT make (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT color (#PCDATA)>
<!ELEMENT engine (engin-num,cylinders-num,fuel-type)>
<!ELEMENT chasis-num (#PCDATA)>
<!ELEMENT accessories (#PCDATA)>
<!ATTLIST accessories disc-brake (yes|no) #REQUIRED
                auto-start (yes|no) #REQUIRED
                radio (yes|no) #REQUIRED>

```

Ex. 8.5.2 : Consider a hospital management system. Write a DTD program to consolidate and show the bill to be paid by the inpatients (Assume your own data).

AU : May-11, Marks 10

Sol. : Patients.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Patients [

```

```
<!ELEMENT Patients (Patient)*>
<ELEMENT Patient (Name, Address,Test,Date,Doctor,Bill)>
<ELEMENT Name (#PCDATA)>
<ELEMENT Address (#PCDATA)>
<ELEMENT Test (#PCDATA)>
<ELEMENT Date (#PCDATA)>
<ELEMENT Doctor (#PCDATA)>
<ELEMENT Bill (#PCDATA)>
]>
<Patients>
  <Patient>
    <Name>Anuradha</Name>
    <Address>Pune</Address>
    <Test>ECG</Test>
    <Date>1-1-2010</Date>
    <Doctor>Mr.XYZ</Doctor>
    <Bill>Rs.500</Bill>
  </Patient>
  <Patient>
    <Name>Archana</Name>
    <Address>Mumbai</Address>
    <Test>Blood Test</Test>
    <Date>10-3-2010</Date>
    <Doctor>Mr.PQR</Doctor>
    <Bill>Rs.100</Bill>
  </Patient>
</Patients>
```

Merits of DTD

1. DTDs are used to define the structural components of XML document
2. These are relatively simple and compact.
3. DTDs can be defined inline and hence can be embedded directly in the XML document.

Demerits of DTD

1. The DTDs are very basic and hence cannot be much specific for complex documents.
2. The language that DTD uses is not an XML document. Hence various frameworks used by XML cannot be supported by the DTDs.
3. The DTD cannot define the type of data contained within the XML document. Hence using DTD we cannot specify whether the element is numeric, or string or of date type.
4. There are some XML processor which do not understand DTDs.
5. The DTDs are not aware of namespace concept.

8.6 Introduction to DOM and SAX**AU : Dec.-15, May-16, 17, Marks 8**

The primary goal of any XML processor is to parse the given XML document. Java has a rich source of in-built APIs for parsing the given XML document. It is parsed in two ways -

- Parsing using DOM(tree based)
- Parsing using SAX(event based)

8.6.1 Difference between DOM and SAX

Sr. No.	DOM	SAX
1.	DOM is a tree based parsing method used to parse the given XML document.	SAX is an event based parsing method used to parse the given XML document.
2.	In this method, the entire XML document is stored in the memory before actual processing. Hence it requires more memory.	In this method, the parsing is done by generating the sequence of events or it calls handler functions.
3.	The DOM approach is useful for smaller applications because it is simpler to use but it is certainly not used for larger XML documents because it will then require larger amount of memory.	Although SAX development is much more challenging, it is useful for parsing the large XML document because the approach is event based, xml gets parsed node by node and does not require large amount of memory.
4.	We can insert or delete a node.	We can insert or delete a node.
5.	Traversing is done in any direction in DOM approach.	Top to bottom traversing is done in this approach.

Features of XML Parsers

1. XML parser is a library that provides methods (or interfaces) for client applications to work with XML documents.
2. It validates the document.
3. It checks for well formedness.
4. DOM is a tree based parsing method.
5. SAX is event based parsing method.
6. In DOM approach the entire XML document is stored in the memory before actual processing.
7. In SAX the sequence of events or handler functions are called for parsing the XML document.
8. In DOM , the XML document is traversed in any direction and in SAX the XML document is traversed from top to down.

8.7 DOM based XML Processing

Using DOM API the XML document can be processed. The DOM API is defined in the package `org.w3c.dom.*`. Using this API a DOM object tree is created from the input XML document and this tree helps in parsing the XML document.

Ex. 8.7.1 : Write XML document for checking Well Formedness of XML document using DOM API.

Sol. :

Java Program [parsing_DOMDemo.java]

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
```

} 1

```
public class Parsing_DOMDemo
{
```

```
    static public void main(String[] arg)
    {
```

```
        try
        {
```

```
            System.out.print("Enter the name of XML document ");
            BufferedReader input = new BufferedReader(new
            InputStreamReader(System.in));
            String file_name = input.readLine();
            File fp = new File(file_name);
            if(fp.exists())
            {
```

} 2

```
                try
                {
```

```
                    DocumentBuilderFactory Factory_obj =
                    DocumentBuilderFactory.newInstance();
                    DocumentBuilder builder = Factory_obj.newDocumentBuilder();
                    InputSource ip_src = new InputSource(file_name);
                    Document doc = builder.parse(ip_src);
                    System.out.println(file_name + " is well-formed!");
```

} 3

```
                }
```

```
            catch (Exception e)
```

```
            {
```

```
                System.out.println(file_name + " isn't well-formed!");
                System.exit(1);
```

} 4

```
            }
```

```
        }
    }
}
```

```
        {
            System.out.print("File not found!");
        }
    }
    catch(IOException ex)
    {
        ex.printStackTrace();
    }
}
}
```

Program Explanation

1. In above Java program we have to import some useful packages initially. Those are described as follows -
 - The **java.io** package provides the interface for performing simple input and output operations.
 - The **javax.xml.parsers.*** package provides the classes allowing the processing of XML documents. It supports various classes such as **DocumentBuilder** and **DocumentBuilderFactory**.
 - The package **org.w3c.dom.*** provides the interfaces for Document Object Model which is a component API of JAVA API for XML processing.
 - The package **org.xml.sax.*** provides the classes and interfaces for the Simple API for XML (SAX) which is a component API of JAVA API.
2. Reading the name of XML document using the command prompt. Using the input stream for the **BufferedReader** class we can read the contents from the command prompt.
3. **DocumentBuilderFactory** is a factory API an application can obtain parser. This parser basically produces DOM object tree from given XML document. Using the object of **DocumentBuilderFactory** an instance for **DocumentBuilder** is created. The object of **DocumentBuilder** is used to invoke a method **parse**. This method takes as XML document as an input, parses it. If the XML document is well formed then appropriate message will be displayed. In an XML document if every starting tag has an ending tag then that document is said to be well formed otherwise it is not.
4. In the **try** block we are calling the method **parse** for parsing the XML document. If the XML document is not well formed then the control of the program will go to **catch** block.

We will consider following XML document for executing above program -

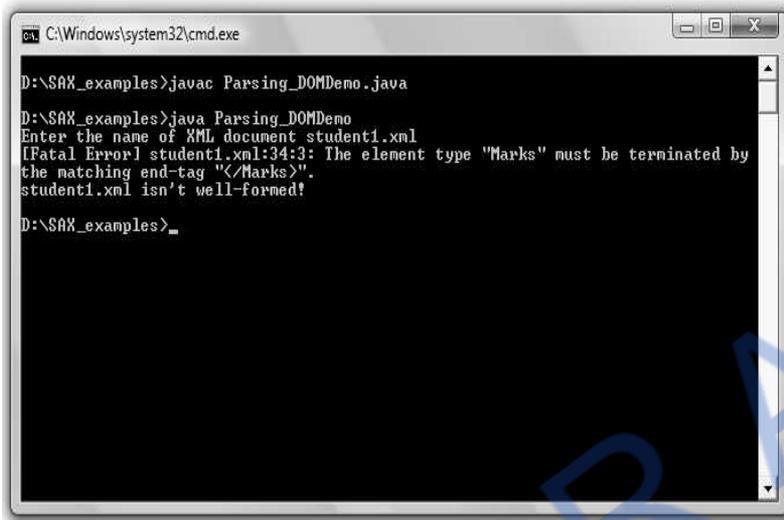
student1.xml

```
<?xml version="1.0" ?>
<student>
  <Roll_No>10</Roll_No>
  <Personal_Info>
    <Name>Parth</Name>
    <Address>Pune</Address>
    <Phone>1234567890</Phone>
  </Personal_Info>
  <class>Second</class>
  <Subject>Mathematics</Subject>
  <Marks>100
  <Roll_No>20</Roll_No>
  <Personal_Info>
    <Name>Anuradha</Name>
    <Address>Banglore</Address>
    <Phone>90901233</Phone>
  </Personal_Info>
  <class>Fifth</class>
  <Subject>English</Subject>
  <Marks>90</Marks>
  <Roll_No>30</Roll_No>
  <Personal_Info>
    <Name>Anand</Name>
    <Address>Mumbai</Address>
    <Phone>90901256</Phone>
  </Personal_Info>
  <class>Fifth</class>
  <Subject>English</Subject>
  <Marks>90</Marks>
</student>
```

Purposely made this statement like this! (It is not well formed)

For getting the output go to the command prompt and give the commands as shown in the following screenshot.

Output



```

C:\Windows\system32\cmd.exe

D:\SAX_examples>javac Parsing_DOMDemo.java

D:\SAX_examples>java Parsing_DOMDemo
Enter the name of XML document student1.xml
[Fatal Error] student1.xml:34:3: The element type "Marks" must be terminated by
the matching end-tag "</Marks>".
student1.xml isn't well-formed!

D:\SAX_examples>_

```

Naturally if we correct the statement like this,

```
<Marks>100</Marks>
```

Then we will get the **output** as follows -

```
D:\SAX_examples>javac Parsing_DOMDemo.java
```

```
D:\SAX_examples>java Parsing_DOMDemo
Enter the name of XML document student1.xml
student1.xml is well-formed!
D:\SAX_examples>
```

8.8 Event-Oriented Parsing : SAX

AU : Dec.-15, May-16,17, Marks 8

Now we will discuss simple Java programs that can be built to parse the given XML document using the event based API i.e. SAX

Ex. 8.8.1 : Write XML document for checking Well Formedness of XML document using SAX API.

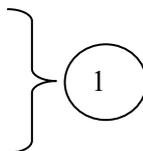
Sol. :

Java Program [Parsing_SAXDemo.java]

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
```

```
public class Parsing_SAXDemo
{
```

```
    public static void main(String[] args) throws IOException
    {
```



```

try
{
    System.out.print("Enter the name of XML document ");
    BufferedReader input = new BufferedReader(new
InputStreamReader(System.in));
    String file_name = input.readLine();
    File fp = new File(file_name);
    if (fp.exists())
    {
        try
        {
            XMLReader reader = XMLReaderFactory.createXMLReader();
            reader.parse(file_name);
            System.out.println(file_name + " is well-formed.");
        }
        catch (Exception e)
        {
            System.out.println(file_name + " is not well-formed.");
            System.exit(1);
        }
    }
    else
    {
        System.out.println("File is not present: " + file_name);
    }
}
catch (IOException ex){ex.printStackTrace();}
}

```

Program Explanation (The numbered steps are explained as follows)

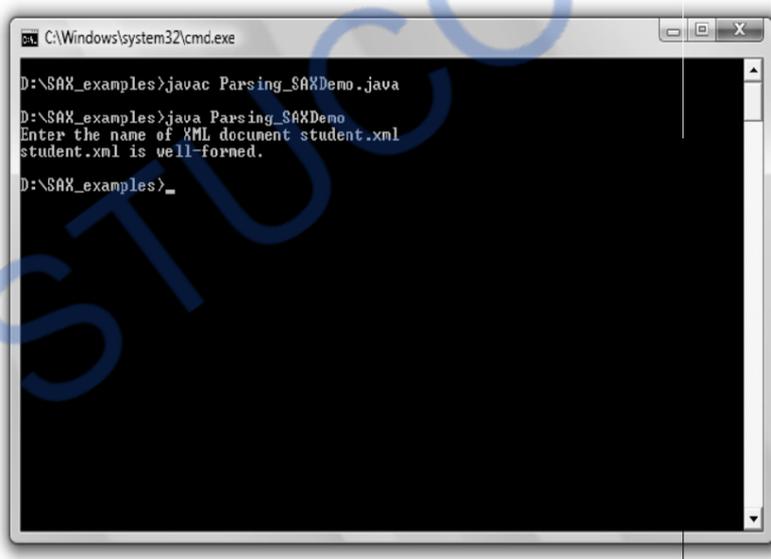
- In above Java program we have to import some useful packages initially. Those are described as follows -
 - The **java.io** package provides the interface for performing simple input and output operations.
 - The package **org.xml.sax.*** provides the classes and interfaces for the Simple API for XML (SAX) which is a component API of JAVA API.
 - The package **org.xml.sax.helpers.*** provides the helper classes for Simple API for XML(i.e. SAX) which is a component of JAVA API for processing XML.
- Reading the name of XML document using the command prompt. Using the input stream for the **BufferedReader** class we can read the contents from the command prompt.
- The **XMLReaderFactory** helps in creating an XML reader. This reader then parses xml document using the appropriate callbacks. Using **createXMLReader** an object **reader** is created using which we can call the method **parse** for parsing the XML document.

4. In the **try** block we are calling the method **parser** for parsing the XML document. If the XML document is not well formed then the control of the program will go to **catch** block. A sample XML document that can be parsed using above program is as given below -

student.xml

```
<?xml version="1.0" ?>
<student>
  <Roll_No>10</Roll_No>
  <Personal_Info>
    <Name>Parth</Name>
    <Address>Pune</Address>
    <Phone>1234567890</Phone>
  </Personal_Info>
  <class>Second</class>
  <Subject>Mathematics</Subject>
  <Marks>100</Marks>
</student>
```

Output



```
C:\Windows\system32\cmd.exe
D:\SAX_examples>javac Parsing_SAXDemo.java
D:\SAX_examples>java Parsing_SAXDemo
Enter the name of XML document student.xml
student.xml is well-formed.
D:\SAX_examples>_
```

University Questions

1. Explain XML parsers and validation.
2. List the essential features of XML parsers.
3. Discuss the merits and demerits of DOM and SAX parsers with neat examples.

AU : Dec.-15, Marks 8

AU : May-16, Marks 8

AU : May-17, Marks 8

8.9 XSLT : Displaying XML Documents in Browsers

- The XSLT stands for **XSL Transformations** and XSL stands for eXtensible Stylesheet Language.
- An XSLT is a W3C recommendation.
- The XSLT is used for defining the XML document transformations and presentations. Hence XSL decides how an XML document should look on the web browser.
- XSL consists of three parts :
 1. XSLT is a language for **transforming XML documents**.
 2. XPath is a language for **navigating in XML documents**. In other words we can reach to any node of XML document using XPath.
 3. XSL-FO is a language for formatting XML documents for **displaying it in desired manner**.

8.9.1 Transforming XML into XSLT

- Using XSLT we can decide the way by which we want the element to get displayed.
- XSLT uses **XPath** to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents. We can sort these elements, hide or display particular element and can apply some decision making logic on these elements.
- XSLT processors **take two input documents** one is **XML document** and another is **XSLT document**. The XSLT document is nothing but a program and XML document is nothing but the input data, thus this program works on the XML input data. Then some or whole part of the XML document is selected, modified and merged with XSLT program document in order to produce another document.
- This newly produced document is provided as input to the XSLT processor which in turn produce another document called the **XSL document**.
- The XSL document is used along with the application so that particular application can be displayed on the web browser in some desired manner.

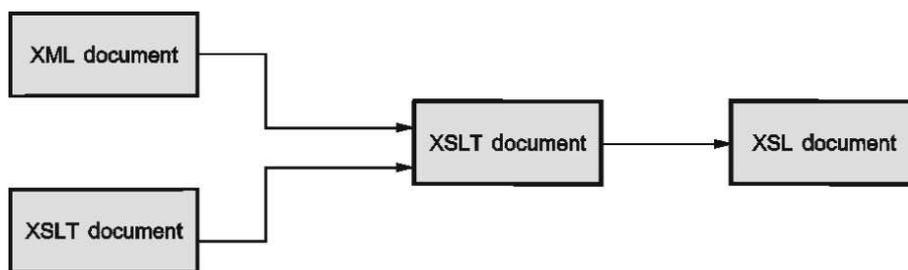


Fig. 8.9.1 Processing of XSLT document

- The XSLT document makes use of **templates** using which particular code can be described in XML document.
- XML document and then particular code in XSLT is executed when the match is found.
- XSLT processor processes the XML document sequentially by reading each line one by one.
- The XSLT model can be described as **template driven** or **data driven** model.

8.9.2 XSL Elements

1. The <xsl:template>

- The <xsl:template> is for building the templates.
- The **match** attribute is associated with the template element. The match attribute can also be used to define a template for the entire XML document. The **match="/"** defines the whole document.
- In the following example we store the XML elements in the tabular form using the <xsl:template> and <match>

Step 1 : Write a simple XML document. One sample example is as below -

Open Notepad and write the following code. Save it by the name SimpleXml.xml

XML Document[SimpleXml.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="SimpleXml.xsl"?>
<Student >
<Person-Details >
    <name>Anand</name>
    <address>Pune</address>
    <std>Second</std>
    <marks>70 percent</marks>
</Person-Details >
    <Person-Details >
    <name>Anuradha</name>
    <address>Chennai</address>
    <std>Second</std>
    <marks>90 percent</marks>
    </Person-Details >
<Person-Details >
    <name>Archana</name>
    <address>Mumbai</address>
    <std>Forth</std>
    <marks>80 percent</marks>
</Person-Details >
<Person-Details >
```

We have defined this file in step 2

```

    <name>Monika</name>
    <address>Delhi</address>
    <std>Tenth</std>
    <marks>77 percent</marks>
</Person-Details>
</Student>

```

Step 2 : Now create a XSL stylesheet. Note that this document must have file name extension as **.xsl**. Open notepad and save following code in it.

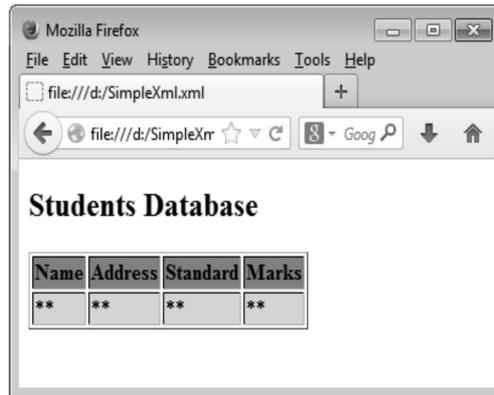
XSLT Document [SimpleXml.xsl]

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<tr bgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
<tr bgcolor="pink">
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Step 3 : Open the web browser and the output will be as follows -



2. The <xsl:value-of>

The **value-of** is used to extract the value of xml element and add it to output stream of the XSL transformation. Following example illustrates this

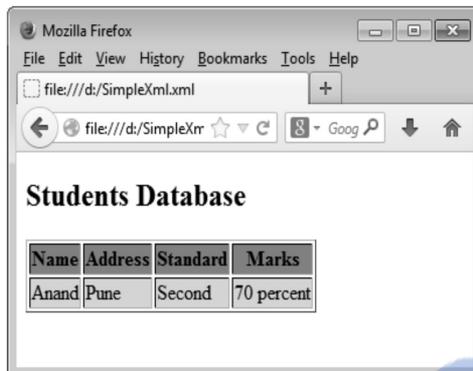
Step 1 : Keep the same xml[**SimpleXml.xml**] file which you have created in section 8.9.2 (1) Step 1.

Step 2 : Create the xsl file as follows -

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<tr bgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
<tr bgcolor="pink">
<td><xsl:value-of select="Student/Person-Details/name"/></td>
<td><xsl:value-of select="Student/Person-Details/address"/></td>
<td><xsl:value-of select="Student/Person-Details/std"/></td>
<td><xsl:value-of select="Student/Person-Details/marks"/></td>
</tr>
</table>
</body>
</html>
```

```
</xsl:template>
</xsl:stylesheet>
```

Step 3 : Open the web browser and the output will be as follows -



3. The <xsl:for-each>

The **for-each** element allows to traverse through each element of the node

Step 1 : Keep the same xml[SimpleXml.xml] file which you have created in section 8.9.2(1)
Step 1.

Step 2 : Now create .xsl document illustrating <xsl:for-each>. It is as follows -

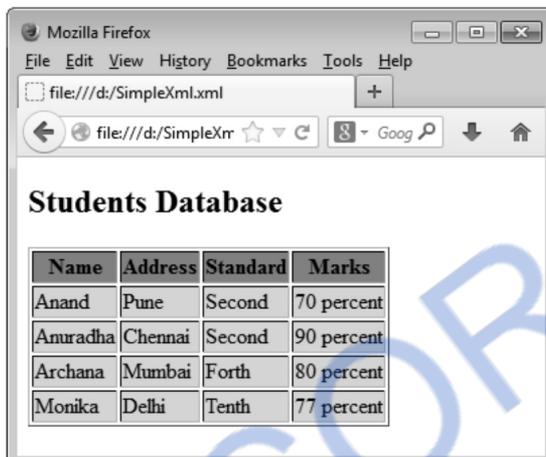
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<tr bgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
<xsl:for-each select="Student/Person-Details">
<tr bgcolor="pink">
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="address"/></td>
<td><xsl:value-of select="std"/></td>
<td><xsl:value-of select="marks"/></td>
</tr>
</xsl:for-each>
```

```

</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Step 3 : Open the web browser and the output will be as follows -



4. The <xsl:sort>

We can also display the sorted data using <xsl:sort>.

Step 1 : Keep the same xml[SimpleXml.xml] file which you have created in section 8.9.2 (1) Step 1.

Step 2 : Create the .xsl file as follows –

SimpleXml.xsl

```

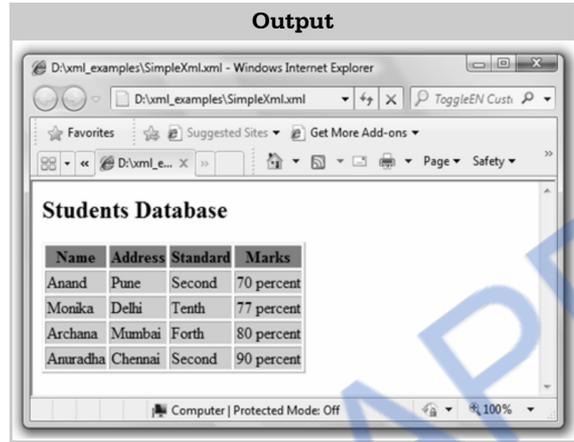
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<tr bgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>

```

```

<xsl:for-each          select="Student/Person-
Details">
<xsl:sort select="marks"/>
<trbgcolor="pink">
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="address"/></td>
<td><xsl:value-of select="std"/></td>
<td><xsl:value-of select="marks"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```



5. The <xsl:if>

For selection specific record based on some condition the <xsl:if> is used.

Step 1 : Keep the same xml[SimpleXml.xml] file which you have created in section 8.9.2(1) Step 1.

Step 2 : Create the .xsl file as follows –

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<trbgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
<xsl:for-each select="Student/Person-Details">
<xsl:if test="marks>77">
<trbgcolor="pink">
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="address"/></td>
<td><xsl:value-of select="std"/></td>
<td><xsl:value-of select="marks"/></td>
</tr>
</xsl:if>

```

```

</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

6. The <xsl:choose>

The <xsl:choose> element is used along with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.

Step 1 : Keep the same xml[SimpleXml.xml] file which you have created in section 8.9.2 (1) Step 1.

Step 2 : Create the .xsl file as follows –

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<tr bgcolor="gray">
<th>Name</th>
<th>Marks</th>
</tr>
<xsl:for-each select="Student/Person-Details">
<tr>
<td><xsl:value-of select="name"/></td>
<xsl:choose>
<xsl:when test="marks &lt; 77">
<td bgcolor="pink">
<xsl:value-of select="marks"/></td>
</xsl:when>
<xsl:otherwise>
<td><xsl:value-of select="marks"/></td>
</xsl:otherwise>
</xsl:choose>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

8.10 Displaying XML Documents in Browser using CSS

- Cascading Style Sheet(CSS) is used to apply the style to HTML elements. Similarly we can apply the style to XML elements using CSS style
- The document structure can be specified in DTD file.
- The style sheet can be specified in a separate CSS file.
- **For example :** Following is a web application that represent the book database displaying Book Title, Author Name, publication, Edition,ISBN, Price in different fonts and color

Step 1 : Create a DTD file name *Catalog.dtd*

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Catalog (Book)* >
<!ELEMENT Book ( Title, Author, Publication, Edition, ISBN, Price ) >
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Publication (#PCDATA)>
<!ELEMENT Edition (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
```

Step 2 : Create a CSS file and name it as *library.css*

```
Catalog
{
font-family:arial;
color:red;
font-size:16pt
}
Book
{
display:block;
font-family:times new roman;
color:blue;
font-size:14pt
}
Title
{
font-family:arial;
color:maroon;
font-size:12pt
}
Author
{
font-family:arial;
color:magenta;
```

```

}
Publication,Edition,ISBN,Price
{
display:block;
font-family:arial;
color:black;
font-size:10pt;
margin-left: 20pt
}

```

Step 3 : Finally create a XML document and name it as CSSDemo.css as follows -

XML Document

```

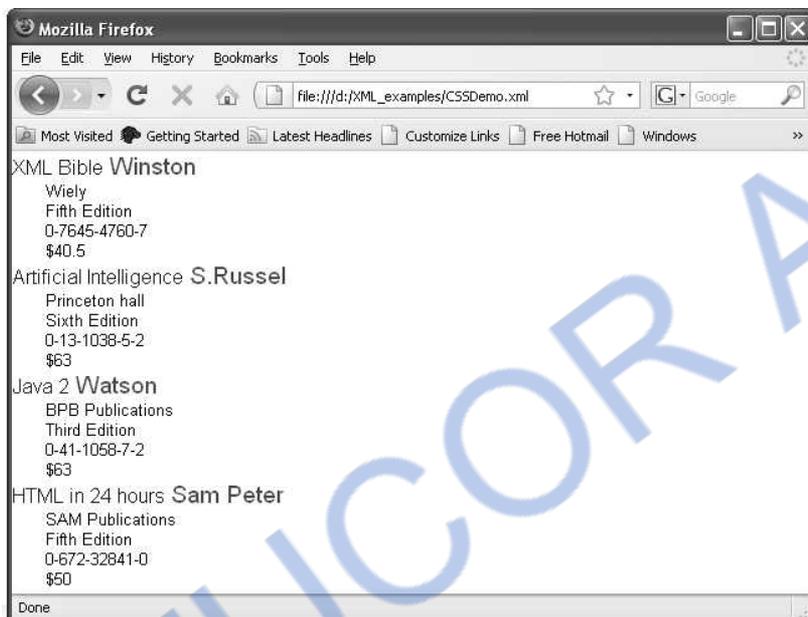
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="library.css"?>
<!DOCTYPE Catalog SYSTEM "Catalog.dtd">
<Catalog>
  <Book>
    <Title>XML Bible</Title>
    <Author>Winston</Author>
    <Publication>Wiely</Publication>
    <Edition>Fifth Edition</Edition>
    <ISBN>0-7645-4760-7</ISBN>
    <Price>$40.5</Price>
  </Book>
  <Book>
    <Title>Artificial Intelligence</Title>
    <Author>S.Russel</Author>
    <Publication>Princeton hall</Publication>
    <Edition>Sixth Edition</Edition>
    <ISBN>0-13-1038-5-2</ISBN>
    <Price>$63</Price>
  </Book>
  <Book>
    <Title>Java 2</Title>
    <Author>Watson</Author>
    <Publication>BPB Publications</Publication>
    <Edition>Third Edition</Edition>
    <ISBN>0-41-1058-7-2</ISBN>
    <Price>$63</Price>
  </Book>
  <Book>
    <Title>HTML in 24 hours</Title>
    <Author>Sam Peter</Author>
    <Publication>SAM Publications</Publication>
    <Edition>Fifth Edition</Edition>
    <ISBN>0-672-32841-0</ISBN>
  </Book>

```

Here both CSS and DTD files are used externally

```
<Price>$50</Price>
</Book>
</Catalog>
```

Step 4 : Finally execute the XML document on the web browser and you can see following sort of output -



Two Marks Questions with Answers

Q.1 What is XML?

Ans. : The XML stands for eXtensible Markup Language. The XML is designed to transport or to store the data.

Q.2 What is the difference between XML and HTML?

Ans. : The difference between HTML and XML is that HTML is designed for representation of data on the web page whereas XML is designed transport or to store the data. The HTML has its predefined tags whereas the XML does not have predefined tags.

Q.3 What are goals of XML?

Ans. : Following are the goals of XML -

1. User must be able to define and use his own tag. This allows us to restrict the use of the set of tags defined by proprietary vendors.
2. Allow user to build his own tag library based on his web requirement.
3. Allow user to define the formatting rules for the user defined tags.

4. XML must support for storage or transport of data.

Q.4 Explain the following terms related to XML :

i) XML parsing ii) XML browsers iii) XML editors iv) XML validators

AU : May-17

Ans. :

i) XML parsing : This is a technique in which the XML parser reads the XML file into the memory and converts into XML DOM object.

ii) XML browser : The XML browser visualizes the elements in XML document. The user can browse through or can associate external interfaces with the XML elements. For examples the XML file can be viewed using Internet Explorer or Firefox browser.

iii) XML editors : XML editors allow the developer to write error free XML document by editing it. The plain text editor like notepad can also be used for editing the XML document. But there are some graphical XML editors like XML Spy or oXygen that present the content to the user in a more user-friendly format.

iv) XML validators : The validation of XML document is done by XML parser. It can be done using two approaches.

i) Checking the well formedness of XML document - That means if any bracket is missing or additional is tested by checking the well formedness of the XML document.

ii) Checking the validity of XML elements.

Q.5 Discuss the important features of XML which makes it more suitable for creating web related services.

Ans. : Following are some features which are most suitable for creating web related applications.

1. XML is EXtesible Markup Language intended for transport or storage of the data.
2. The most important feature of XML is that user is able to define and use his own tag. This allows us to restrict the use of the set of tags defined by proprietary vendors. On the other hand other markup languages like HTML or SGML requires a predefined set of tags.
3. XML contains only data and does not contain any formatting information. Hence document developers can decide how to display the data.
4. XML permits the document writer to create the tags for any type of information. Due to this virtually any kind of information can be such as mathematical formulae, chemical structures, or some other kind of data can be described using XML.
5. Searching sorting, rendering or manipulating the XML document is possible using eXtended Stylesheet Language (XSL).

6. The XML document can be validated using some external tools.
7. Some commonly used web browsers like Internet Explorer and Firefox Mozilla provide support to the tags in XML. Hence XML is not at all vendor specific or browser specific.

Q.6 What are the advantages of XML ?**AU : May-17****Ans. :**

1. XML document is human readable and we can edit any XML document in simple text editors.
2. The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl.
3. Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.

Q.7 What is XML element ?**Ans. :** The basic entity in XML is element. The elements are used to define tags.

The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

The syntax of writing any element for opening tag is

```
<element name>
```

The syntax of writing any closing element for closing tag is

```
< /element name >
```

Every XML must have a single root element.

Q.8 Why DTD's ?**AU : May-09**

Ans. : The document type definition is used to define the basic building block of any xml document. Using DTD we can specify the various elements types, attributes and their relationship with one another. Basically DTD is used to specify the set of rules for structuring data in any XML file.

For example : If we want to put some information about some students in XML file then, generally we use tag student followed by his/her name, address, standard and marks. That means we are actually specifying the manner by which the information should be arranged in the XML file. And for this purpose the Document Type Definition is used.

Q.9 What is meant by namespace ?**AU : Dec.-09, 12, May-11**

Ans. : The xml document allows us to create different elements which are having the common name. This technique is known as namespace.

For example : Consider the following XML document

XML Document

```
<File-Description>
<text fname="input.txt">
  <describe>It is a text file</describe>
</text>
<text fname="flower.jpg">
  <describe>It is an image file</describe>
</text>
</File-Description>
```

The above document does not produce any error although the element text is used for two different attribute values.

Q.10 What is CDATA ?

Ans. : The CDATA stands for Character Data. The character data will be parsed by the parser.

Q.11 What is PCDATA ?

Ans. : It stands for Parsed Character Data (i.e. text). Any Parsable character data should not contain the markup characters. The markup characters are < or > or &. If we want to use less than, greater than or ampersand characters then make use of < , > or &

Q.12 What are the merits of DTD ?

Ans. :

1. DTDs are used to define the structural components of XML document
2. These are relatively simple and compact.
3. DTDs can be defined inline and hence can be embedded directly in the XML document.

Q.13 What are the demerits of DTD ?

Ans. :

1. The DTDs are very basic and hence cannot be much specific for complex documents.
2. The language that DTD uses is not an XML document. Hence various frameworks used by XML cannot be supported by the DTDs.
3. The DTD cannot define the type of data contained within the XML document. Hence using DTD we cannot specify whether the element is numeric, or string or of date type.
4. There are some XML processor which do not understand DTDs.
5. The DTDs are not aware of namespace concept.

Q.14 What is XSL and why it is used ?

AU : May-08

Ans. : XSL stands for eXtensible Stylesheet Language. The XSL decided how an XML document should look on web browser.

Q.15 What is the difference between DOM and SAX ? [Refer section 8.6.1]

AU : May-13

Q.16 What are the two methods of parsing the XML document ?

Ans. : The XML document is parsed using two approaches -

- 1) Tree based
- 2) Event based.

The tree based parsing is done using DOM and th Event based parsing is done using SAX.

Q.17 What is the importance of SAX ?

Ans. : The SAX stands for Simple API for XML. It allows to access the information of XML document using sequence of events. This API is basically used for parsing the XML document.

Q.18 How is XML parsing done with SAX ?

AU : Dec.-11

Ans. : The XML parsing can be done with SAX using following steps -

A Java program can make use of following steps.

- 1) Using the **BufferedReader** class the contents of XML document are read.
- 2) Then using **XMLReadFactory** XML reader is created. This reader parses the XML document using appropriate callbacks.
- 3) Using **createXMLReader** an object **reader** is created.
- 4) With the help of this object the method parse is invoked for parsing the XML document.

Q.19 What does XSLT mean ?

AU : May-12

Ans. : The XSLT stands for eXtensible Stylesheet Language Transformation. It is a W3C recommendation.

Q.20 What is the purpose of XSLT ?

AU : Dec.-11

Ans. : The purpose of XSLT is to define the XML document transformations and presentations. Thus using XSLT the XML document can be transformed into XHTML documents or some other XML document.

Q.21 What is Xpath ?

AU : Dec.-12

Ans. : The XSLT uses the Xpath to navigate in XML document. In other words we can reach to any node of XML document using XPath.

Q.22 What is XSL-FO ?

Ans. : The XSL-FO is a language for formatting XML document for displaying it in desired manner.

Q.23 What is the use of CSS for XML document ?

Ans. : In XML user defined tags are used. The cascading style sheet can be embedded within the XML document using which the desired style can be applied to the user defined tags. Thus the XML document can be presented in the desired style using the CSS.

Q.24 What is XML Parse Tree ?

AU : Dec.-15

Ans. : The XML document form a tree structure that starts from root node to branches.

The XML parse tree is a DOM model in which everything from XML is treated as node.

This tree is created for parsing the XML document.

Various properties such as nodeName, nodeValue, parentNode, childNodes and so on are used while parsing the XML document.

Q.25 Why is XSLT an important tool in development of web applications ?

AU : May-16

Ans. : The XSLT stands for XSL Transformations. It is used for defining the XSL document transformations and presentations. Thus XSLT decides how an XML document should look on web browser.

Q.26 What is XML parser ?

AU : Dec 17

Ans. : XML parser is a parser that is designed to read XML and check the well formedness of the document. The XML parser also validates the document.

Q.27 What is Well formed XML document ?

AU : May-18

Ans. : Refer section 8.2.2.



Notes

STUCOR APP

Unit V

9

AJAX

Syllabus

AJAX : Ajax Client Server Architecture-XML Http Request Object-Call Back Methods.

Contents

9.1	AJAX Client Server Architecture	Dec.-15,16, May-16	Marks 16
9.2	XMLHttpRequest Object	May -17,	Marks 10
9.3	Call Back Methods		
9.4	Coding AJAX Script		
9.5	Two Marks Questions with Answers		

9.1 AJAX Client Server Architecture

AU : Dec.-15, 16, May-16, Marks 16

9.1.1 Introduction to AJAX

- AJAX is a **Asynchronous JavaScript and XML**
- Here
 - **Asynchronous** means, the execution of script does not disturb the user's work.
 - **JavaScript** because, it makes use of Javascripting to do the actual work.
 - **XML** because along with JavaScript the XML is also supported to perform the given task in AJAX.
- It is not a new programming language but it is a kind of web document which adopts certain standards.
- AJAX allows the developer to exchange the data with the server and updates the part of web document without reloading the web page.

9.1.2 Architecture

- When user makes a request, the browser creates a object for the **HttpRequest** and a request is made to the server over an internet.
- The Server processes this request and sends the required data to the browser.
- At the browser side the returned data is processed using JavaScript and the web document gets updated accordingly by sending the appropriate response.

Following Fig. 9.1.1 illustrates this working.

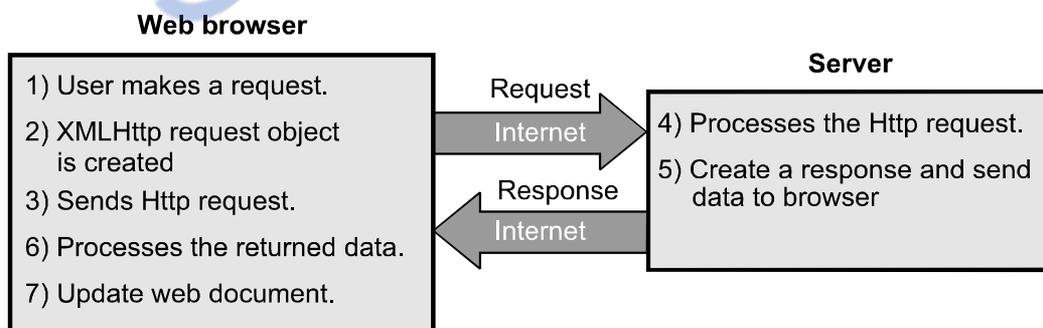


Fig. 9.1.1 Working of AJAX

Similarities and Differences

Sr.No.	Traditional Web Application Architecture	Ajax Based Web Application Architecture
1.	The web applications architectures are based on HTTP request response protocol.	
2.	At the client side the browser client has only one component and that is – user interface.	At the client side the browser client has user interface and AJAX Engine due to which the client gets quick response from the server.
3.	The architecture is based on client server communication.	
4.	It makes use of synchronous communication. The client has to wait to get the response from the server then only client can make the request to the server. It is start-stop-start-stop kind of communication.	By adding a new layer of AJAX Engine at the client side, it eliminates the start-stop-start-stop nature of communication and makes the communication asynchronous.
5.	With traditional web application architecture user cannot get rich user interface experience.	With AJAX architecture user gets rich user interface experience.
6.	It is less responsive.	It is more responsive.
7.	Building web application is simple.	The development time is more while designing the Ajax based web application.
8.	Limited use of JavaScript, CSS and XML technologies. In fact - Most user actions in the interface trigger an HTTP request back to a web server.	Extensive use of JavaScript, CSS and XML - i) standards-based presentation using XHTML and CSS; ii) Dynamic display and interaction using the Document Object Model(DOM); iii) Data interchange and manipulation using XML and XSLT; iv) Asynchronous data retrieval using XMLHttpRequest; v) JavaScript binding everything together

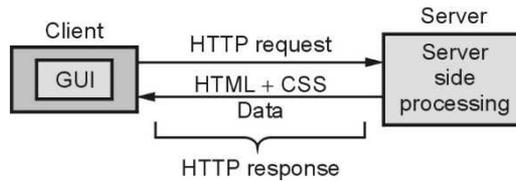


Fig. 9.1.2 Traditional web application architecture

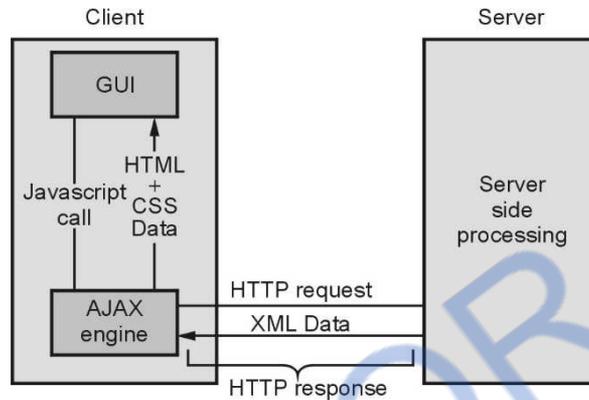


Fig. 9.1.3 AJAX web application architecture

University Questions

1. Discuss Ajax client server architecture in detail. **AU : Dec.-15, Marks 16**
2. Compare and contrast the traditional web application architecture and AJAX based web application architecture. **AU : May-16, Marks 16**
3. Explain the Ajax client server architecture in detail with a diagram. **AU : Dec.-16, Marks 8**

9.2 XMLHttpRequest Object

AU : May-17, Marks 10

- XMLHttpRequest object is an important element in AJAX.
- XMLHttpRequest is an API which is used by JavaScript, VBScript and some other scripting languages.
- The methods of XMLHttpRequest are used in transferring data between a Web browser and a web server.
- It is because of XMLHttpRequest object, to update parts of web page without reloading the whole page.

Syntax

```
Variable = new XMLHttpRequest();
```

Various Methods of XMLHttpRequest object

Method	Purpose
new XMLHttpRequest()	Creates new XMLHttpRequest object.
Open(method, URL, async,user,pwd)	To make request this method is used. method: The request can be GET or POST URL: The location of file async: true(for asynchronous) and false(synchronous) user: user name(optional) pwd:password(optional)
send()	sends the request to the server.
send(string)	sends the request to the server.
setRequestHeader()	Adds the label- value pair to the header.

Various Properties of XMLHttpRequest object

Property	Description
readyState	Specifies the ready state of XMLHttpRequest 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
onreadystatechange	When readystate property changes the onreadystatechange event listener will be automatically invoked
responseText	Returns the response data as a string
status	Returns the status number. Following is the meaning of various numbers returned by the status property 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status text as "OK", or "Forbidden"

Ex. 9.2.1 Create an XMLHttpRequest to retrieve data from an XML file and display the data in an HTML table. The data to be retrieved is a collection of stationary items stored in an XML file.

AU : May -17, Marks 10

Sol. : Step 1 : Create an xml file storing the stationary items. The code is as follows -

```

/ELEMENTS>
<STATIONARY>
<ITEM>Notebook</ITEM>
<PRICE>40</PRICE>
</STATIONARY>
<STATIONARY>
<ITEM>Punching Machine</ITEM>
<PRICE>60</PRICE>
</STATIONARY>
<STATIONARY>
<ITEM>Pencil Box</ITEM>
<PRICE>40</PRICE>
</STATIONARY>
<STATIONARY>
<ITEM>Eraser</ITEM>
<PRICE>5</PRICE>
</STATIONARY>
</ELEMENTS>

```

Step 2 : The XML file displaying the stationary items in a tabular form is as given below –

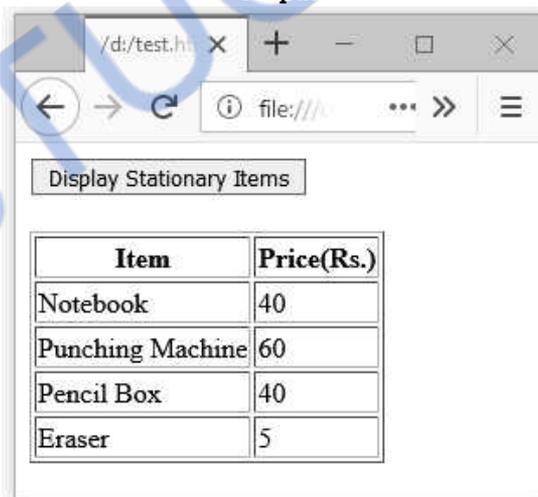
```

test.html
<!DOCTYPE html>
<html>
<body>
<button type="button" onclick="loadXMLDoc()">Display Stationary Items</button>
<br><br>
<table id="demo" border="1"></table>
<script>
function loadXMLDoc()
{
  if(window.XMLHttpRequest)
  {
    xmlhttp = new XMLHttpRequest();
  }
  else
  {
    xmlhttp = new ActiveXObject("Microsoft.req");//to execute on Inter. Explorer
  }
  xmlhttp.onreadystatechange = function()
  {
    if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
    }
  }
}

```

```
}
};
    xmlhttp.open("GET","stationary.xml",true);
    xmlhttp.send();
}
function myFunction(xml)
{
    var i;
    var xmlDoc = xml.responseXML;
    var table="<tr><th>Item</th><th>Price(Rs.)</th></tr>";
    var x = xmlDoc.getElementsByTagName("STATIONARY");
    for (i = 0; i <x.length; i++)
    {
        table += "<tr><td>" +
        x[i].getElementsByTagName("ITEM")[0].childNodes[0].nodeValue +
        "</td><td>" +
        x[i].getElementsByTagName("PRICE")[0].childNodes[0].nodeValue +
        "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
}
</script>
</body>
</html>
```

Output



9.3 Call Back Methods

- Callback functions are used to handle responses from the server in Ajax. A callback function can be either a named function or an anonymous function.

- We can set the value of event handler `onreadystatechange` equal to anonymous function as follows –

```
req.onreadystatechange=function()
{
  if (req.readyState==4 && req.status==200)
  {
    document.getElementById("myID").innerHTML=req.responseText;
  }
}
```

- We can write the call back function as named function using following steps –

Step 1 : Create XMLHttpRequest object in global space

```
var req = new XMLHttpRequest();
```

Step 2 : Write a function which acts as a callback function

```
function myCallBkFun()
{
  if (req.readyState==4 && req.status==200)
  {
    Cosole.log(req.responseText);
  }
}
```

Step 3 : Now we can set the value of `onreadystatechange` property to the name of the call back function.

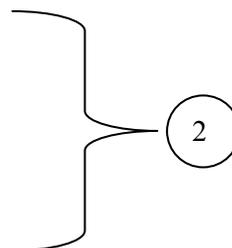
//assigning call back function to onreadystatechange property

```
req.onreadystatechange=myCallBkFun;
```

9.4 Coding AJAX Script

Let understand how AJAX works with the help of programming example

```
<html>
<head>
<script type="text/javascript">
function MyFun()
{
  if (window.XMLHttpRequest)
  {
    req=new XMLHttpRequest();
  }
  else
  {
    req=new ActiveXObject("Microsoft.req");
  }
}
```



```

req.onreadystatechange=function()
{
  if (req.readyState==4 && req.status==200)
  {
    document.getElementById("myID").innerHTML=req.responseText;
  }
}
req.open("GET","newdata.txt",true);
req.send();
}
</script>
</head>
<body>
<div id="myID">This text can be changed</div>

<button type="button" onclick="MyFun()">Change</button>
</body>
</html>

```

Script Explanation (Above numbered steps are explained here)

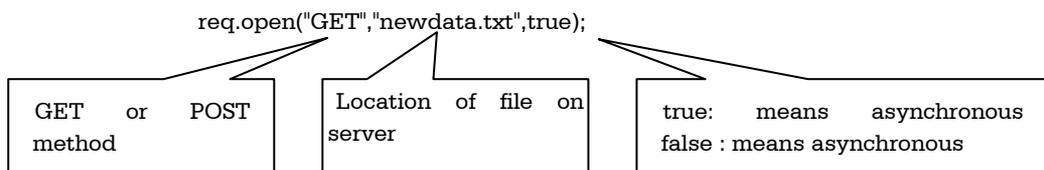
In above script, we have written some text which can be replaced by some another text on button click.

1. On button click a function **MyFun** is invoked. Thus client triggers the event.
2. **XMLHttpRequest** object is used to exchange data with a server. This object allows the user to change/update the parts of the web page without reloading it fully. The modern web browsers such as IE7+, FireFox,Chrome have built in **XMLHttpRequest** but old web browsers make use of **ActiveXObject**.
3. When a request to a server is sent, then **onreadystatechange** event is triggered.

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

The DOM is modified and response is modified using the statement
`document.getElementById("myID").innerHTML=req.responseText;`

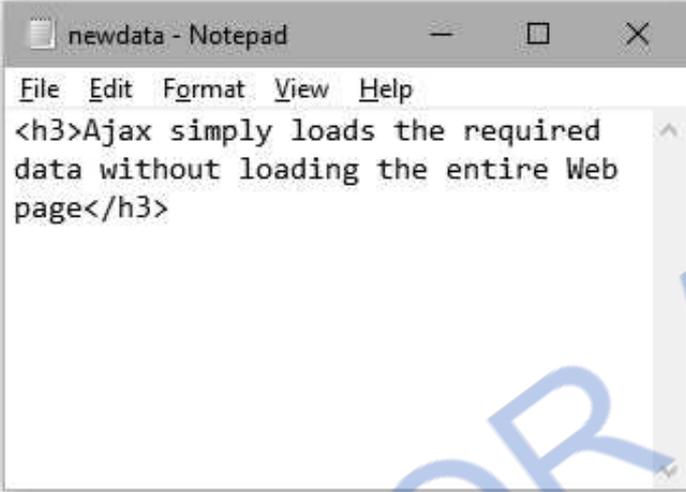
4. The request can be sent to the server by using two functions **open()** and **send()**.



Asynchronous communication allows fast processing of the data.

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

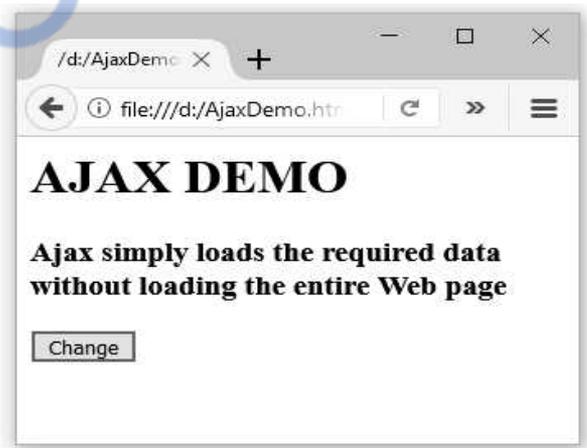
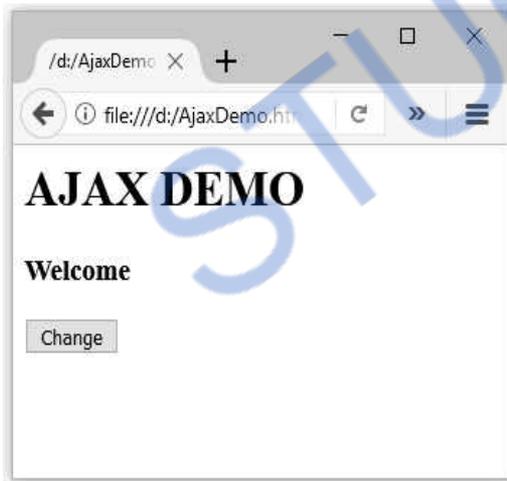
newdata.txt



```
File Edit Format View Help
<h3>Ajax simply loads the required
data without loading the entire Web
page</h3>
```

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

Output



Ex. 9.4.1 : Write AJAX script to obtain the student information stored in XML document. The information should be displayed on clicking the button. It should be displayed in tabular form.

Sol. :

Step 1 : Create an XML file for storing the student information. The XML file is as follows

Student.xml

```
<Student >
  <student_data>
    <Name>AAA</Name>
    <Marks>45</Marks>
  </student_data>
  <student_data>
    <Name>BBB</Name>
    <Marks>55</Marks>
  </student_data>
  <student_data>
    <Name>CCC</Name>
    <Marks>67</Marks>
  </student_data>
  <student_data>
    <Name>DDD</Name>
    <Marks>84</Marks>
  </student_data>
</Student>
```

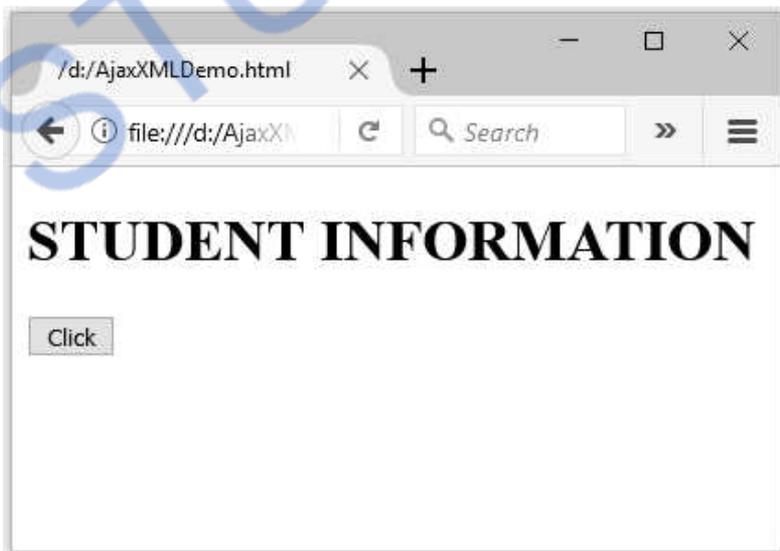
Step 2 : Create a AJAX script as follows –

AjaxXMLDemo.html

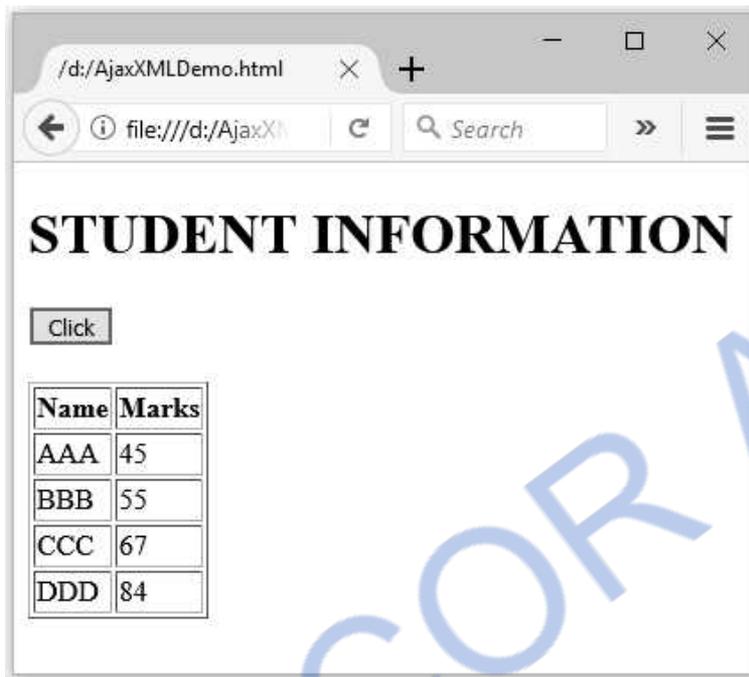
```
<!DOCTYPE html>
<html>
<body>
<h1>STUDENT INFORMATION</h1>
<button type="button" onclick="MyFun()">Click</button>
<br><br>
<table border="1" id="demo"> </table>
<script>
function MyFun()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function()
  {
    if (this.readyState == 4 && this.status == 200)
    {
      Load_XML_File(this);
    }
  };
}
```

```
xhttp.open("GET", "Student.xml", true);
xhttp.send();
}
function Load_XML_File(xml)
{
    var i;
    var xmlDoc = xml.responseXML;
    var table="<tr><th>Name</th><th>Marks</th></tr>";
    var x = xmlDoc.getElementsByTagName("student_data");
    for (i = 0; i <x.length; i++)
    {
        table += "<tr><td>" +
        x[i].getElementsByTagName("Name")[0].childNodes[0].nodeValue +
        "</td><td>" +
        x[i].getElementsByTagName("Marks")[0].childNodes[0].nodeValue +
        "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
}
</script>
</body>
</html>
```

Step 3 : Open the web browser and the output will be as follows –



On clicking the button we will get



Two Marks Questions with Answers

Q.1 What is AJAX ?

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

Q.2 What is Http Request Object ?

Ans. : XMLHttpRequest is an API which is used by JavaScript, VBScript and some other scripting languages.

The methods of XMLHttpRequest are used in transferring data between a Web browser and a web server.

Q.3 What is call back method in AJAX ?

Ans. : Callback functions are used to handle responses from the server in Ajax. A callback function can be either a named function or an anonymous function.

Q.4 What is the use of XMLHttpRequest object ?

AU : Dec.-17

Ans. : The XMLHttpRequest object is a Javascript object. It provides an easy way to retrieve data from a URL without having to do a full page refresh.

Q.5 What are the features of XMLHttpRequest object ?

Ans. : Following are the features of XMLHttpRequest object -

1. Requests the data from the server after the page gets loaded.
2. Receives the data from the server after the page gets loaded.
3. Sends the data to the server behind the scene.
4. Update the web page without reloading the page.

Q.6 What is the use of the property readyState in XMLHttpRequest object ?

Ans. : This property is used to retrieve the current state of request operation.

Q.7 List any four merits for usage of AJAX in web service.**AU : Dec.-15**

Ans. :

- 1) AJAX allows easier and quicker interaction between user and web service as the pages are not reloaded for the contents to be displayed.
- 2) It allows easier navigation to users in comparison to using traditional back and forward button on a browser.
- 3) Traffic to and from the server is reduced a considerable amount.
- 4) The web service becomes more responsive and interactive as the user gets the response without clicking any buttons.
- 5) If one section of a page encounters any error, other sections do not get affected and the data entered by the user is also not lost.

Q.8 What is the role of a callback function in performing a partial page update in an AJAX application ?**AU : May-16**

Ans. : Partial page update are a part of callback functions which process returned data from a server and only update the parts of the page the callback says to update.



Unit V

10	Web Services
----	--------------

Syllabus

JAX-RPC-Concepts-Writing a Java Web Service-Writing a Java Web Service Client-Describing Web Services: WSDL- Representing Data Types: XML Schema-Communicating Object Data: SOAP Related Technologies-Software Installation-Storing Java Objects as Files

Contents

10.1 Concept of Web Service

10.2 JAX- RPC Concepts **Dec.-12,** Marks 8

10.3 Writing a Java Web Service..... **May-12,14,Dec.-13, 16,** Marks 16

10.4 Describing Web Services: WSDL **Dec.-11,15, May-14,16,** Marks 16

10.5 Representing Data Types: XML Schema..... **May-12, Dec.-12,** Marks 8

10.6 Communicating Object Data: SOAP **May-11,** Marks 8,

..... **Dec.-11,12, 13,15,16** Marks 16

10.7 Storing Java Objects as Files **May-11,** Marks 2

Two Marks Questions with Answers

10.1 Concept of Web Service

Definition : The Web Services are the software systems that are displayed by the web browser using the web protocol. These software systems are used by the some software applications rather than by end-users directly.

- Web service is a software system designed which is **independent** of specific hardware or software on which it is running.
- **Examples** of a web service are -
 1. **Credit card validation system** - For using this service the client simply enters the ID and password for the credit card and web service reports the validity of it.
 2. **Whether forecast system** - The clients submits the location and the web service returns the information about current whether as well as it may report some predictions.
 3. **Currency converter** - The client enter the source and destination currency along with the amount and appropriate converted information will be displayed.

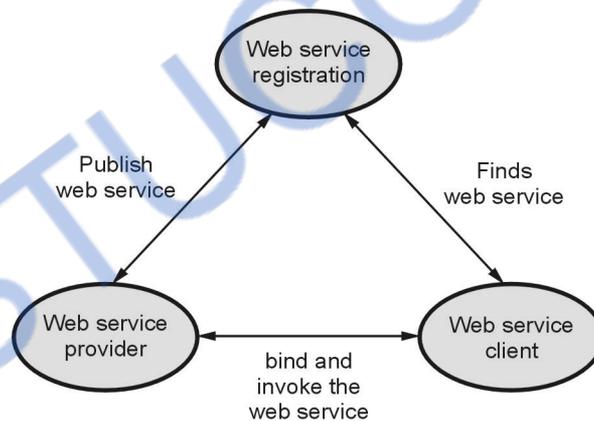


Fig. 10.1.1 Basic web service model

Following are the steps that a web service model follows -

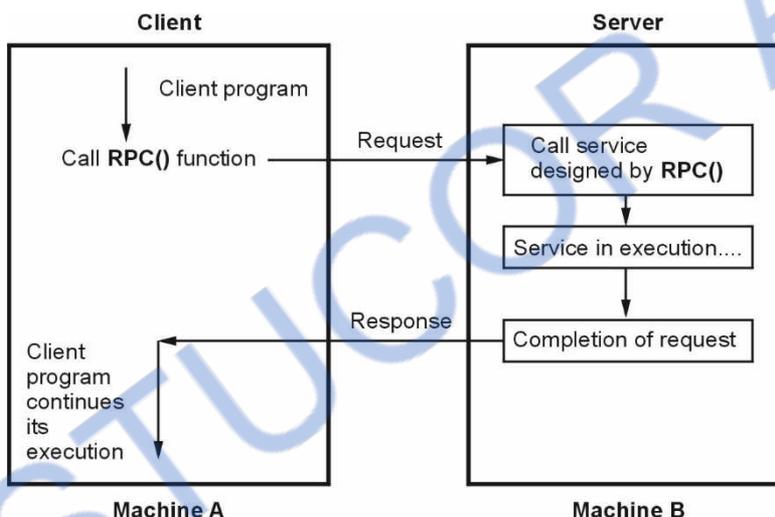
- Step 1 :** In the first step, a service provider **publishes** a web service in the **web service registry**.
- Step 2 :** A web client who demands for some web service **searches** in the registry. After finding a match for the desired web service in the registry. After finding a match for the desired web service in the registry, the client chooses it.
- Step 3 :** The client **binds** to the corresponding web service provider and **invokes** for the service.

10.2 JAX- RPC Concepts

AU : Dec.-12, Marks 8

10.2.1 Concept of RPC

- RPC stands for **Remote Procedure Call**. This technique is basically used in distributed client-server applications.
- By using RPC one can call the any procedure situated in some remote machine. Thus, this technique facilitates that the procedure need not exist in the same address space as the calling procedure.
- Following scenario illustrates the working of RPC.



- RPC is similar to the procedure call.
- When a RPC is called the calling arguments are passed to the remote procedure and the caller waits for the response.
- The client makes a remote procedure call by sending a request to the server.
- At the server end, when the request arrives, server calls dispatch routine and process for the requested service. Finally it sends the replay to the client and then client program continues its execution.

10.2.2 JAX-RPC

The role of various components such as XML, SOAP, WSDL, UDDI in web service building is as shown below -

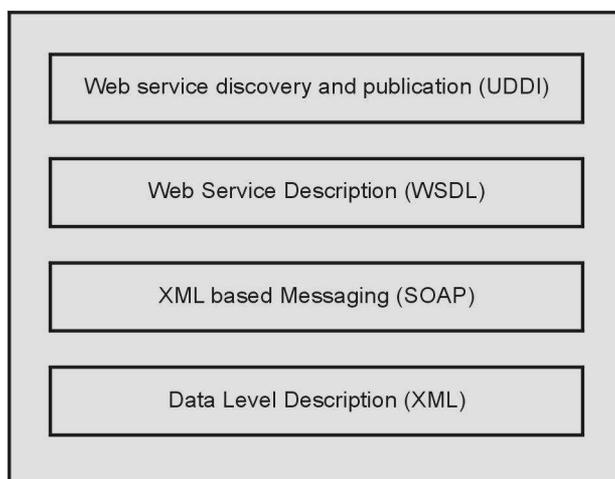


Fig. 10.2.1 Web service elements

- At the lowest level of web services there are simply XML files that are for data level description.
- SOAP - **S**imple **O**bject **A**ccess **P**rotocol is a simple XML based protocol which allows exchange of messaging over HTTP.

Typically web services make use of this protocol for exchange of information .

The SOAP makes use of JAX-RPC component. These are **J**AVA **A**PI for **X**ML that makes use of **R**emote **P**rocedure **C**alls.

Basically SOAP defines message structure, encoding rules and conventions for exchange of information in RPC programming model.

The following table describes core JAX-RPC interfaces and classes.

Interface / Class	Description
Service	Main client interface. Used for both static and dynamic invocations.
Service Factory	This is a factory class which is used to create services
Stub	It is used by client for invoking the web services.
Call	For invoking the web services the call is used.
JAXRPCException	This exception is thrown when an error occurs while invoking web service.

- At the higher level very important element of web service lies i.e. WSDL. The **W**eb **S**ervice **D**escription **L**anguage is used for abstract description of web services. This description is

provided in XML document. When the client wants to make use of web service, the users of the client make read this description file and understand what input data is needed to perform the web service. Such a special file is called WSDL i.e. web services Description Language.

- UDDI i.e. **U**niversal **D**escription and **D**iscovery **I**ntegration, provides a directory of web services, so that client can easily discover the services of their choices. The UDDI maintains a registry in which the web services are registered.

In this chapter we will learn how to develop a simple web service using JAX-RPC. Further we will learn how to build a **web client** for using this service.

University Question

1. Explain the basic concept of RPC.

AU : Dec.-12, Marks 8

10.3 Writing a Java Web Service

AU : May-12,14, Dec.-13, 16, Marks 16

In this section We will Create two Web Projects – one for Server and other for Client.

The Server defines the web service and client invokes this web service

We can develop a web service using Apache Axis2 and Eclipse.

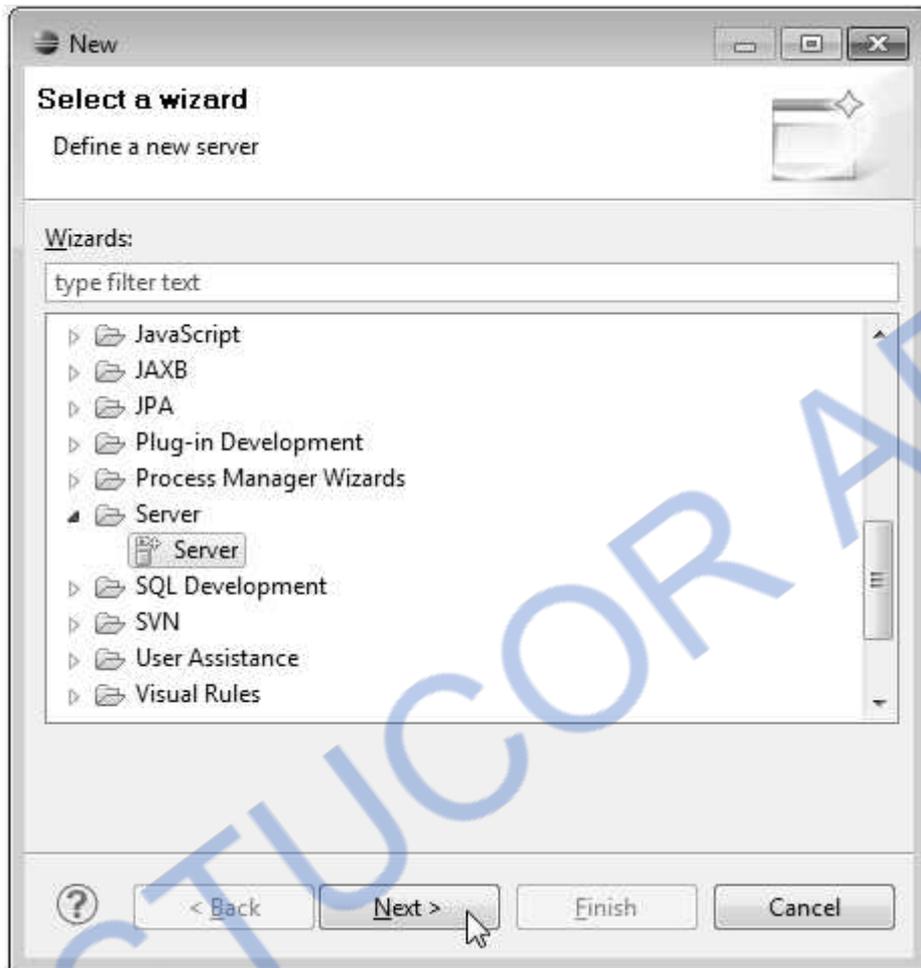
Prerequisite :

- (1) JDK must be installed.
- (2) Eclipse must be installed
- (3) Apache Server, Tomcat must be installed. Install the software package XAMPP for this purpose.

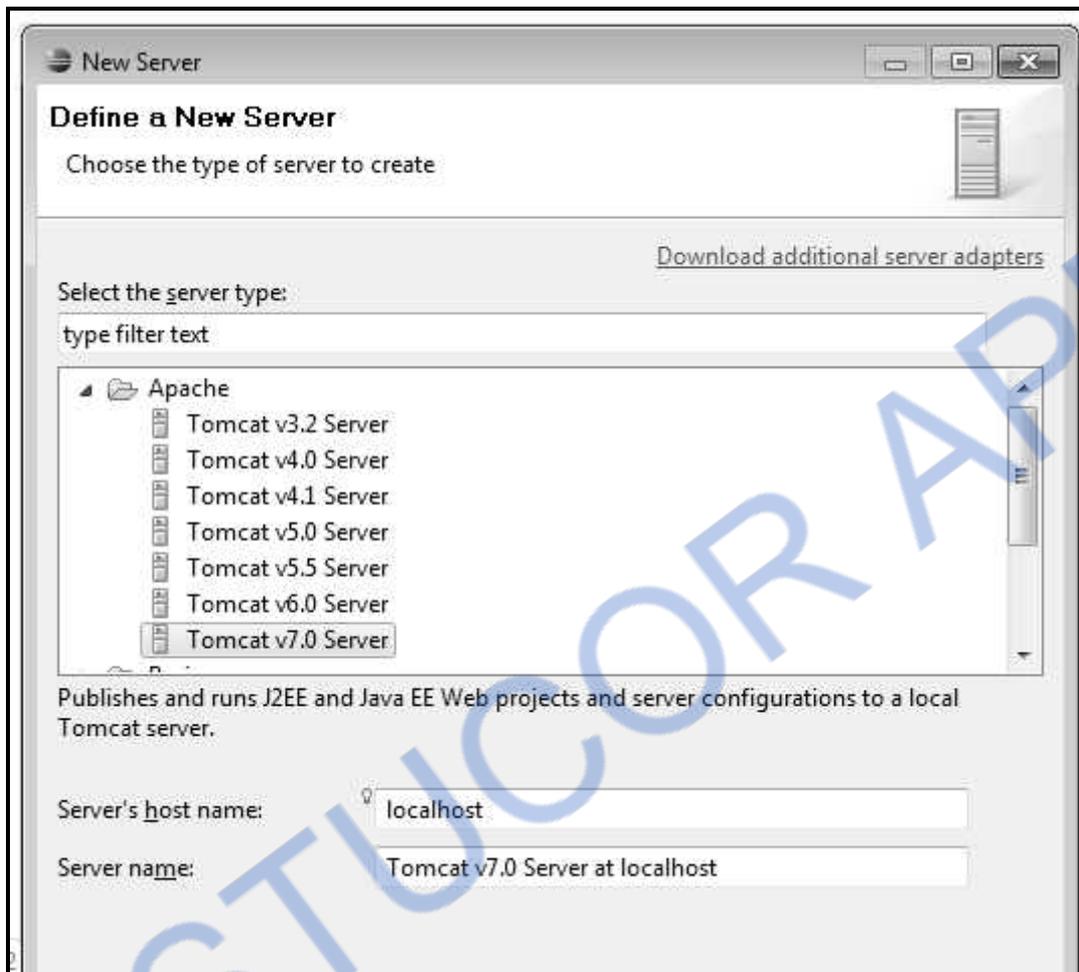
Configuring Tomcat to Eclipse

Step A : To integrate Tomcat Server with your Eclipse, Select **File->New->Other...** from Eclipse main menu.

Step B: Select Server->Server and click Next



Step (3) Select Tomcat Version XX and click **Next**



Step C: Browse to the folder of Tomcat. If XAMP is installed then it is c:\XAMPP\tomcat

Step D: Click **Finish**

(4) Install Axis2.

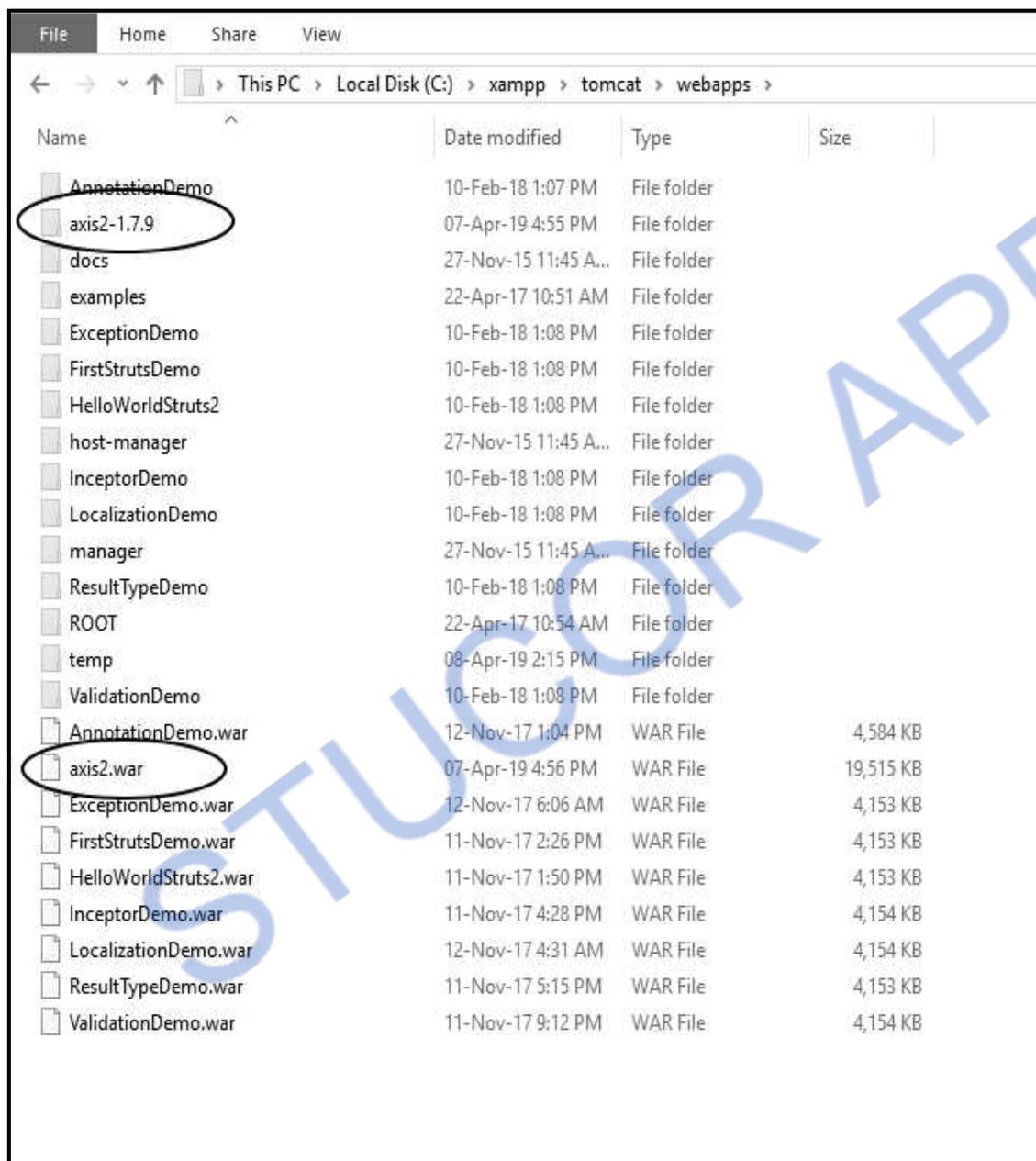
Installation of Axis 2:

Step 1: Open the web browser and get the axis2 package using following URL

<http://axis.apache.org/axis2/java/core/download.html>

Step 2: Click on Binary Distribution Link and download axis2-1.7.9-bin.zip. Also download axis2-1.7.9-war.zip

Step 3: Unzip both the folders and copy these folders at your c:\tomcat-directory\webapps folder. Following screenshot helps in understanding the location



Configuring Axis2 in Eclipse

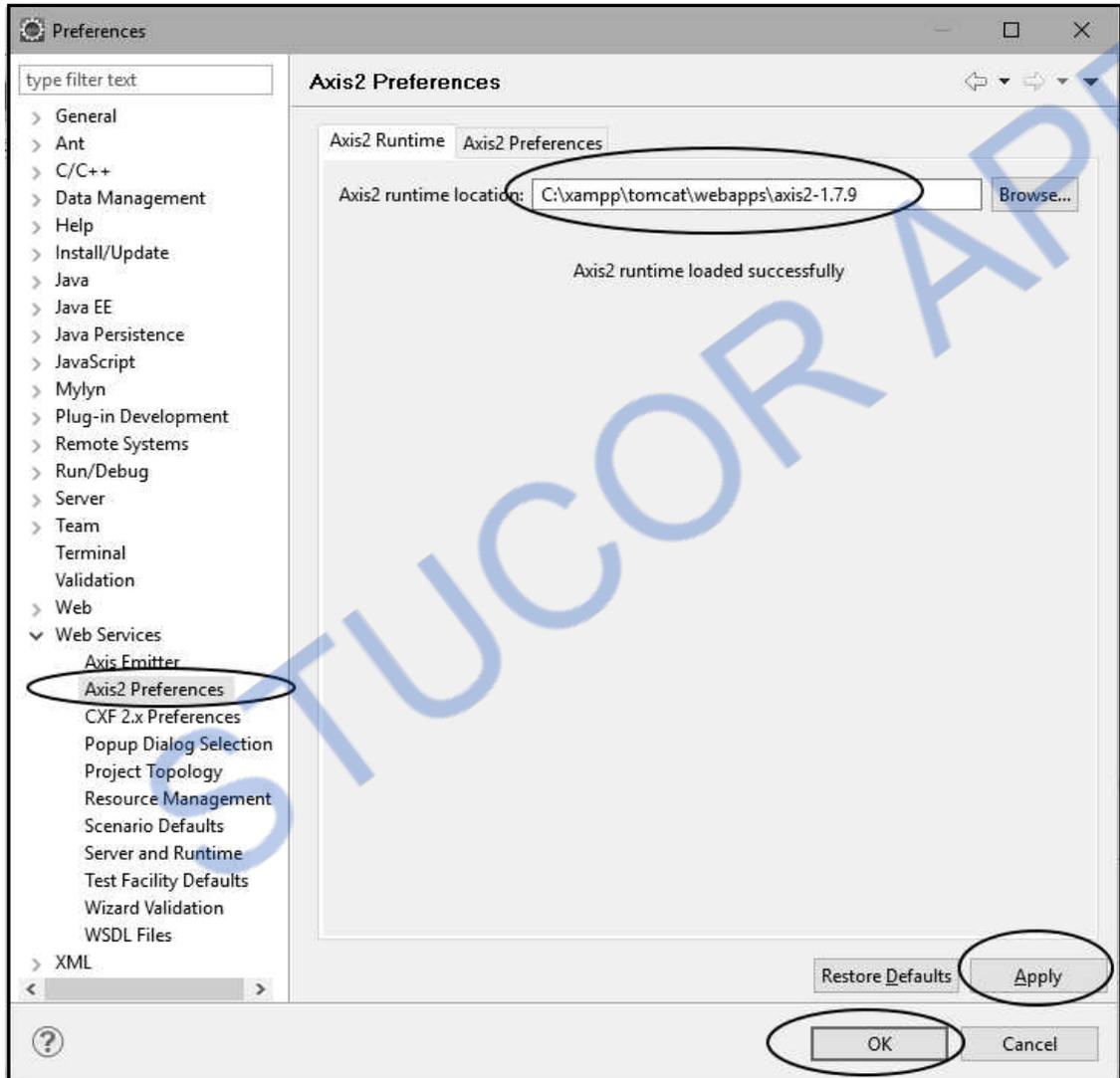
Step 1: Open the Eclipse.

Step 2: Click on **Window** -> **Preferences**.

Step 3 : Locate **Web Services**. Click **Axis2 Preferences**.

Step 4 : Set the location of Axis2 folder with the help of **Browse..** button. It is your Webapps folder in Tomcat directory. On successful deploying of Axis2, we get “Axis2 runtime loaded successfully” message. Click **Apply** and then **Ok** button.

Following Screenshot illustrates it.



Let us now discuss step by step - how to create a simple Web Service.

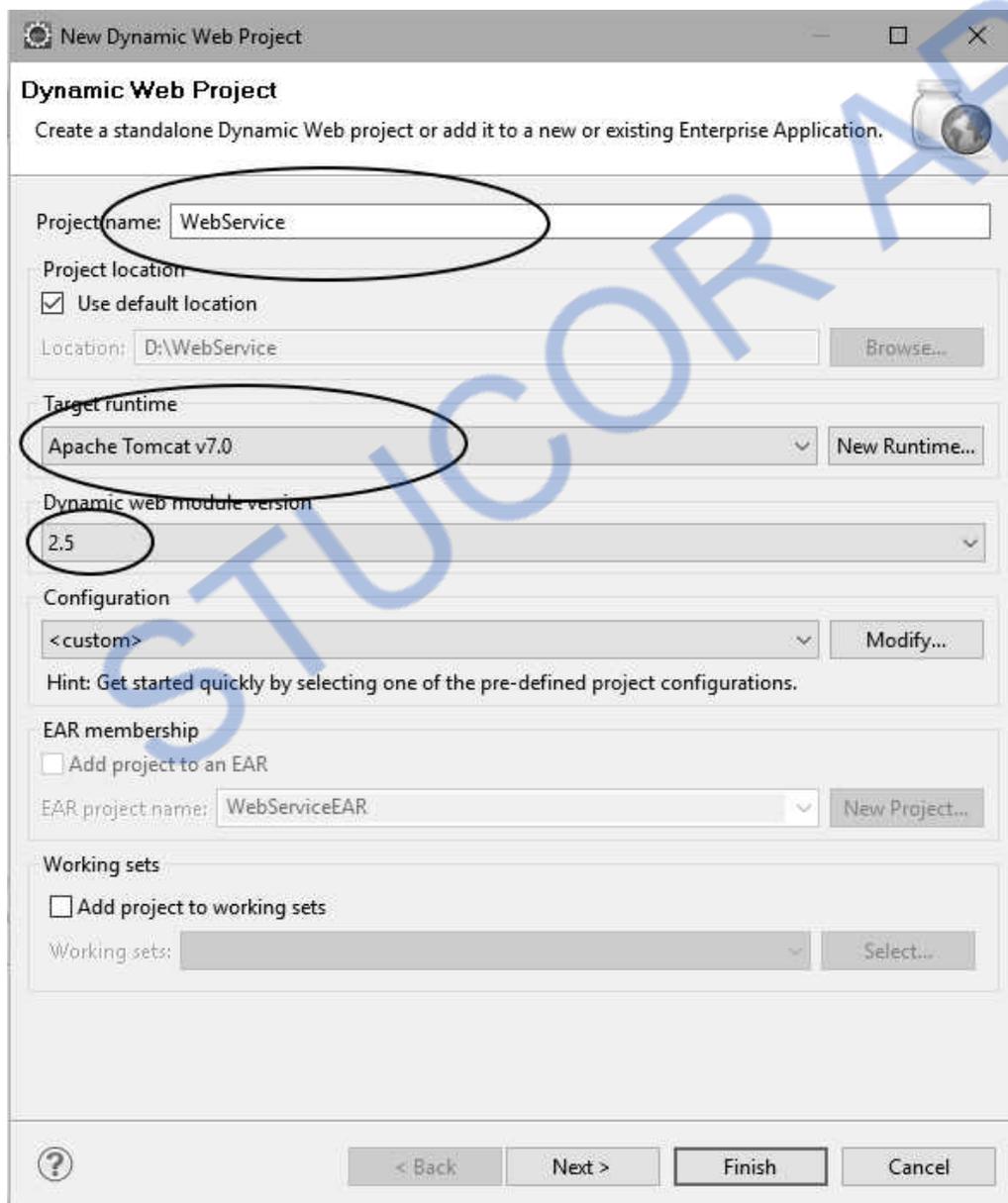
Ex. 10.3.1 Create a web service in java environment to return the sum of two integers with necessary deployment procedures.

AU : May-14, Marks 16

Sol.:

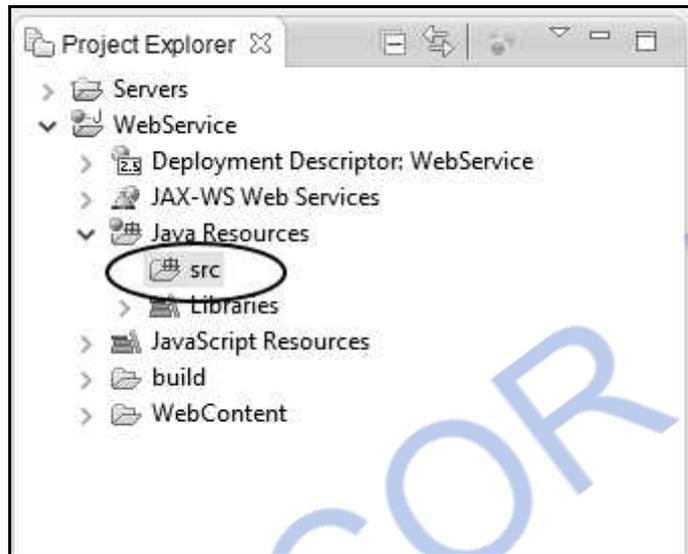
Stage I: Creating Server

Step 1 : Click on Eclipse icon. Click on File->New->Dynamic Web Project. Give the suitable name to the project. I have given the project name as **WebService**. Set the Tomcat Web Server and appropriate version number. Following screenshot illustrates this.

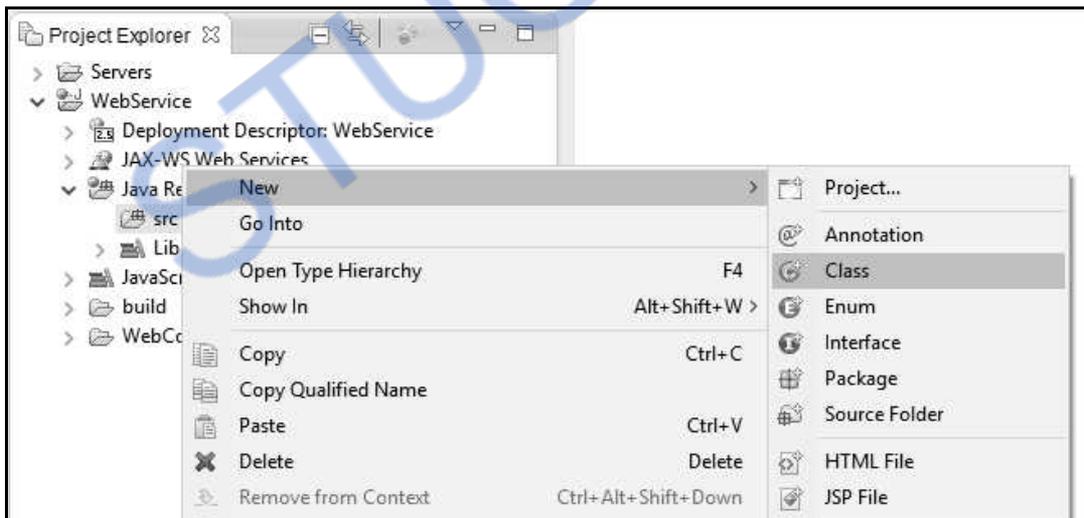


Click **Finish** button.

Step 2 : Locate **src** folder under the project in WebService project. The snapshot of Project Explorer is as shown below



Right click on **src** folder Select **New-> Class**



A New Class Window opens up. Give suitable Package name, Class name and uncheck the check box for void main function. Then click **Finish** button. It is illustrated by following screenshot

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

Enclosing type:

Name:

Modifiers: public default private protected
 abstract final static

Superclass:

Interfaces:

Which method stubs would you like to create?

`public static void main(String[] args);`
 Constructors from superclass
 Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

Step 3 : Now write the following code for **MyServer.java**

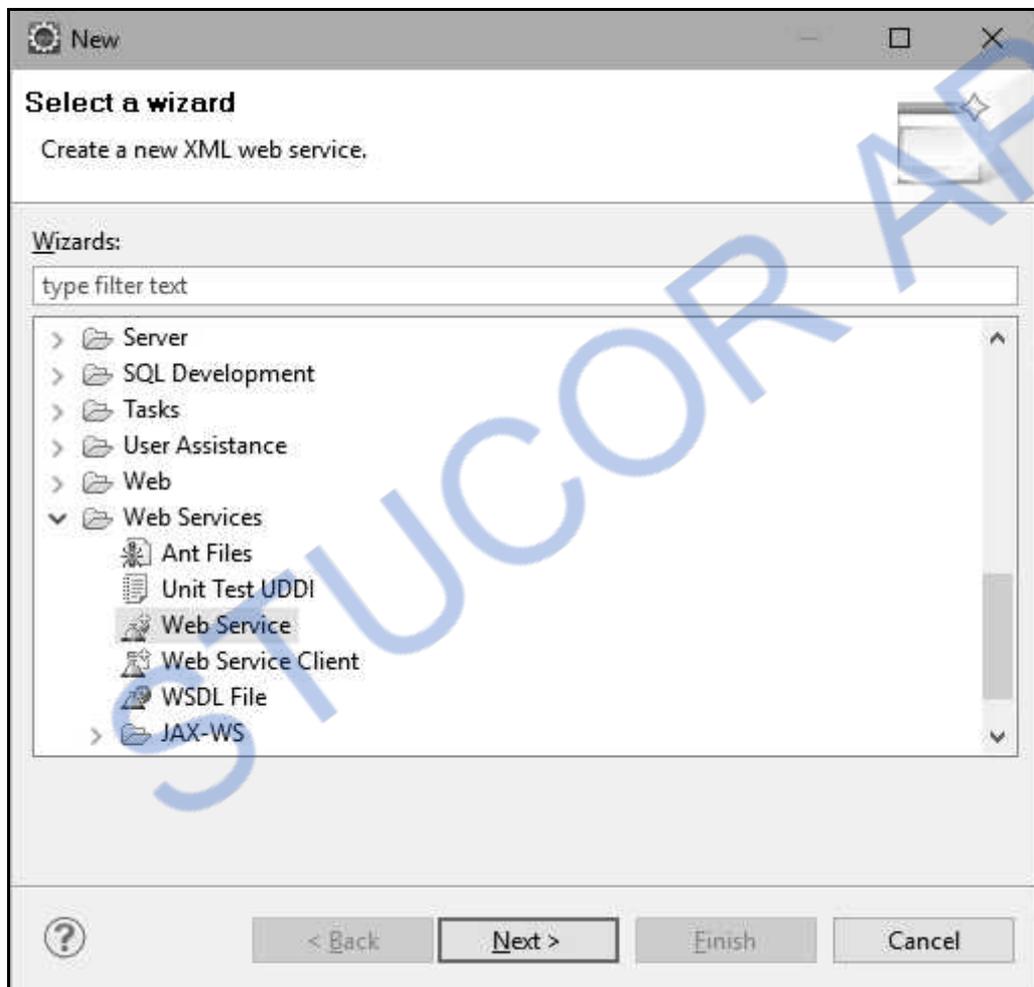
```
package com.mypackage;
```

```
public class MyServer {  
    public int add(int a,int b){
```

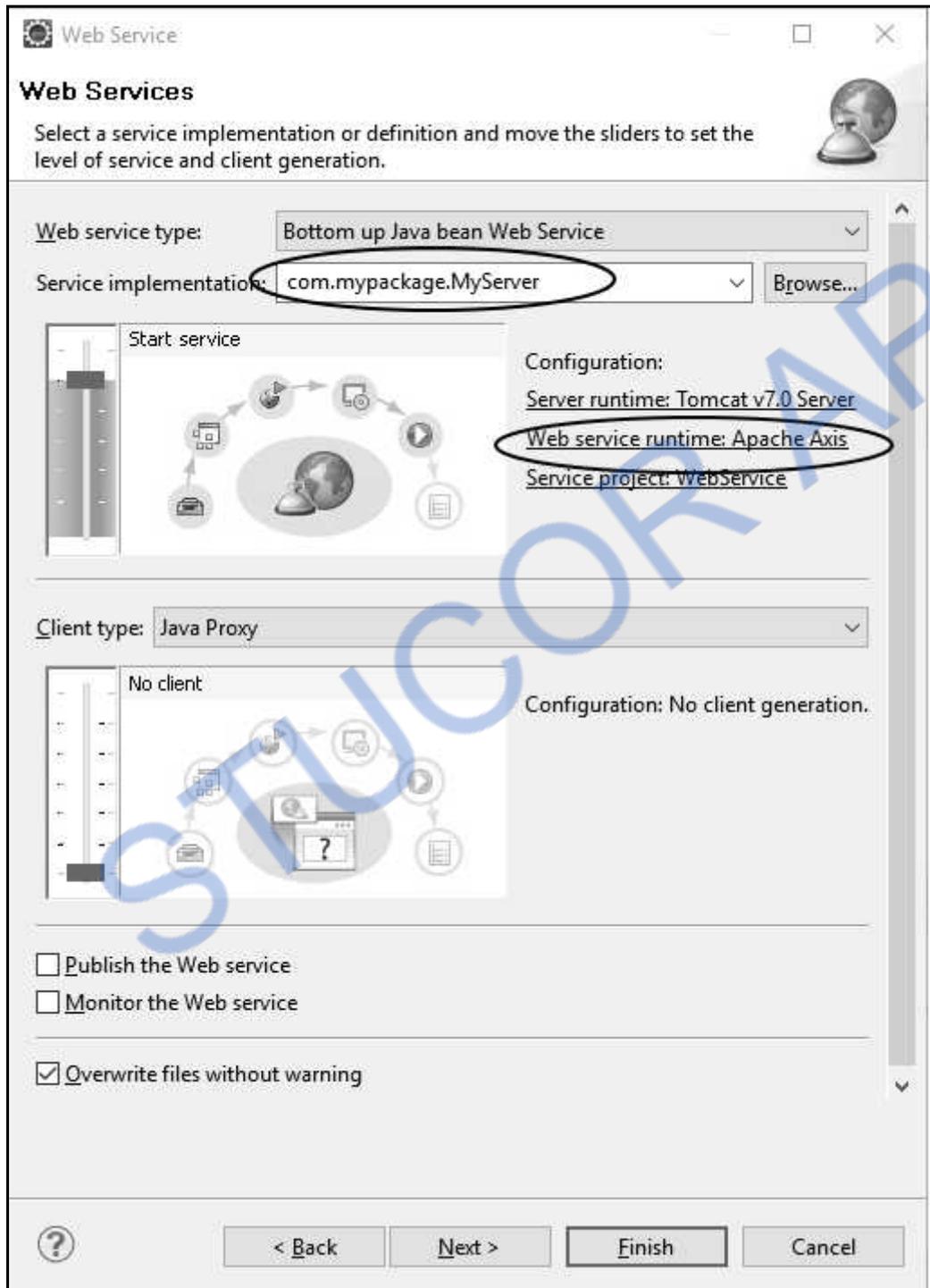
```
    return a+b;
}
}
```

Save the above code.

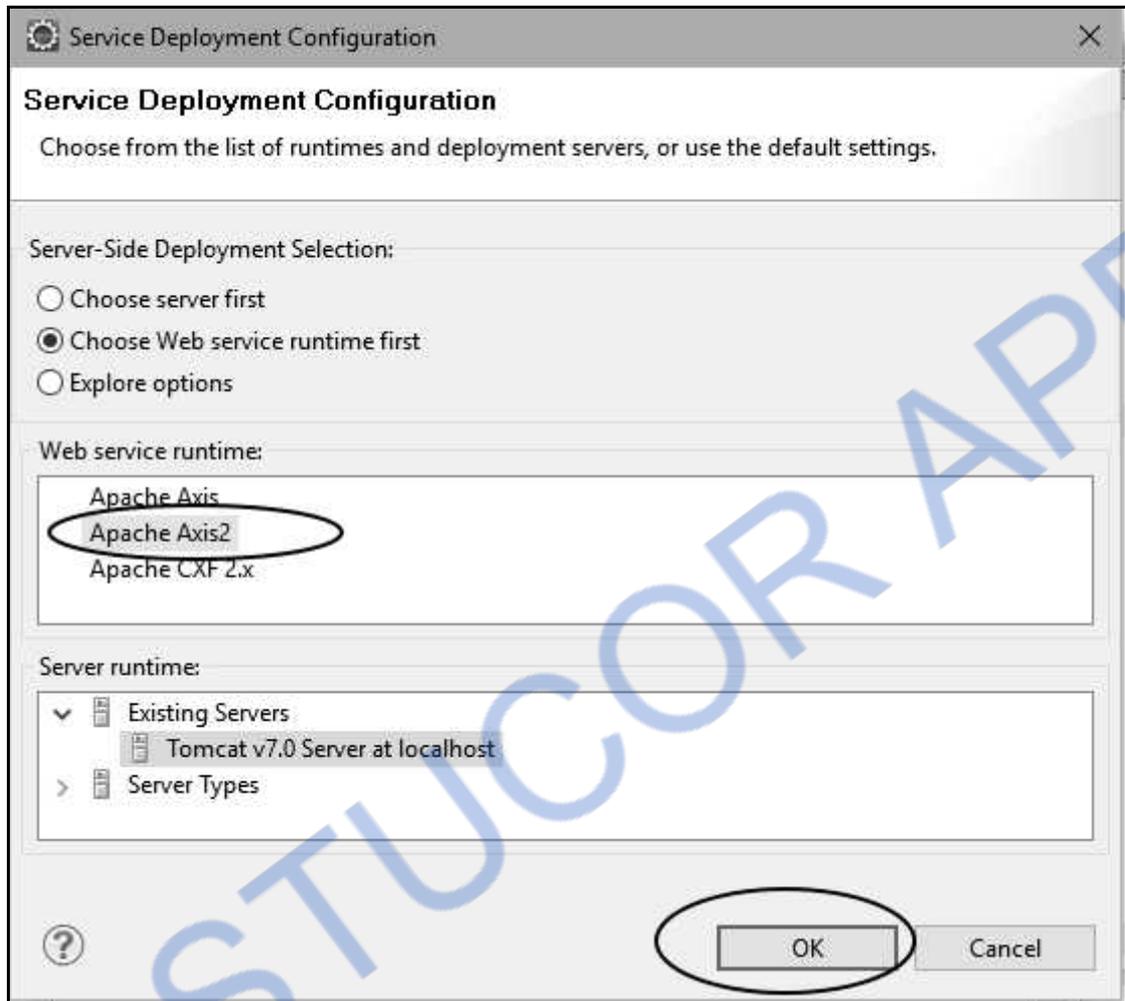
Step 4 : Now right click on **src** folder. Click **New->other->Web Services->Web Service**. Then click **Next** button



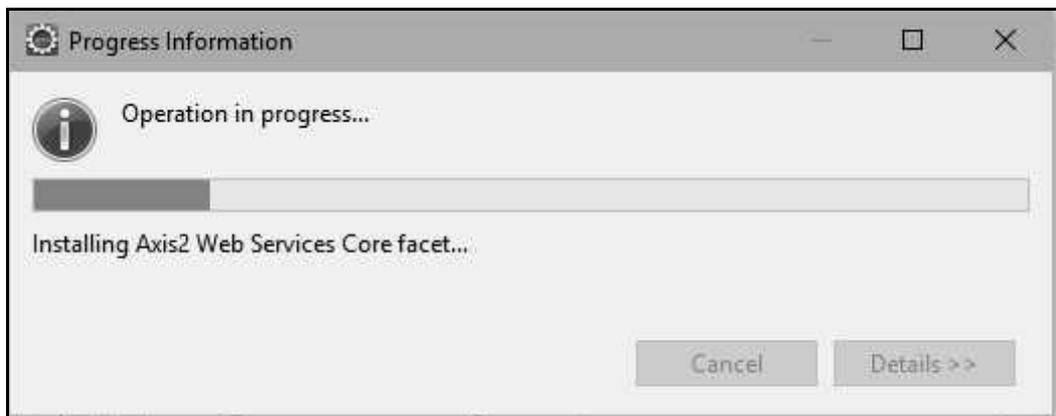
The **Web Services** window will appear. Write the name of service implementation and click on **Web service runtime:Apache Axis**. Following Screenshot shows this-



Click on **Apache Axis2** and then click **Ok** button. Refer following screenshot



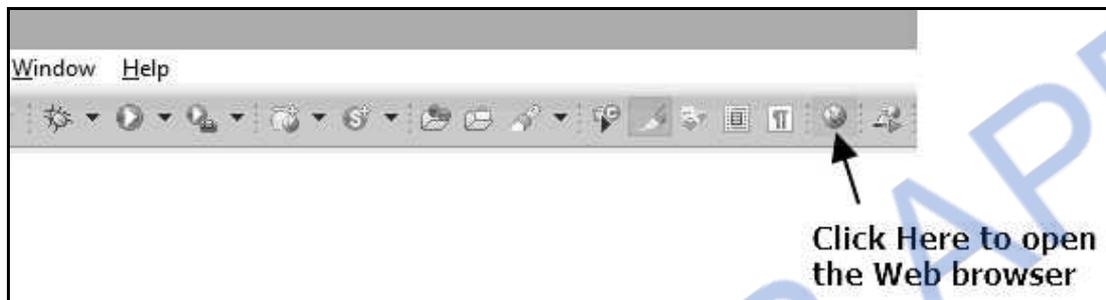
Then click on [Finish](#) button. You will get the progress window as



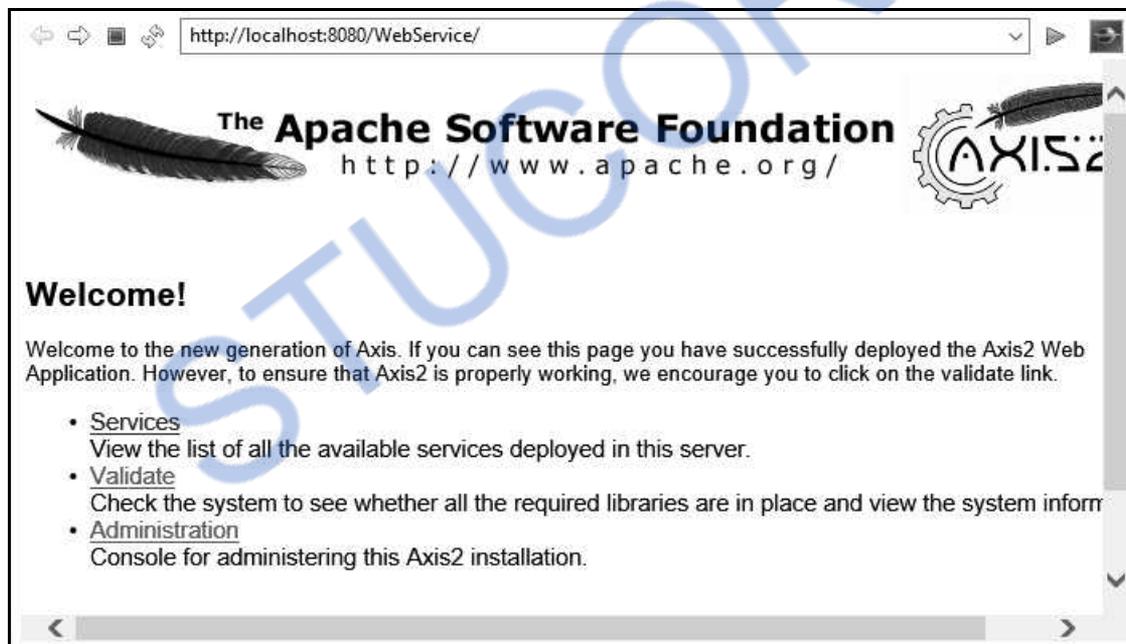
This will generate the WSDL file for your application.

Step 5 : To view the generated WSDL file we need to open up the Web Browser in Eclipse. For that

1. Open Web Browser present on the Toolbar



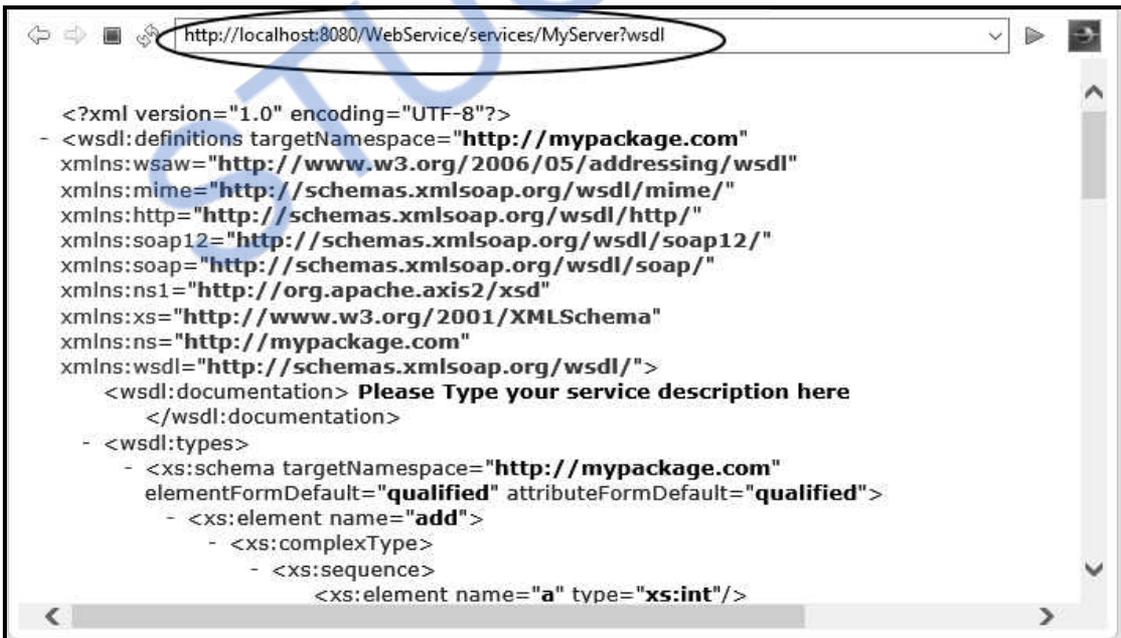
2. Type the URL as follows -



3. Now click on **Services**. Your Web Service name **MyServer** can be located on the page.



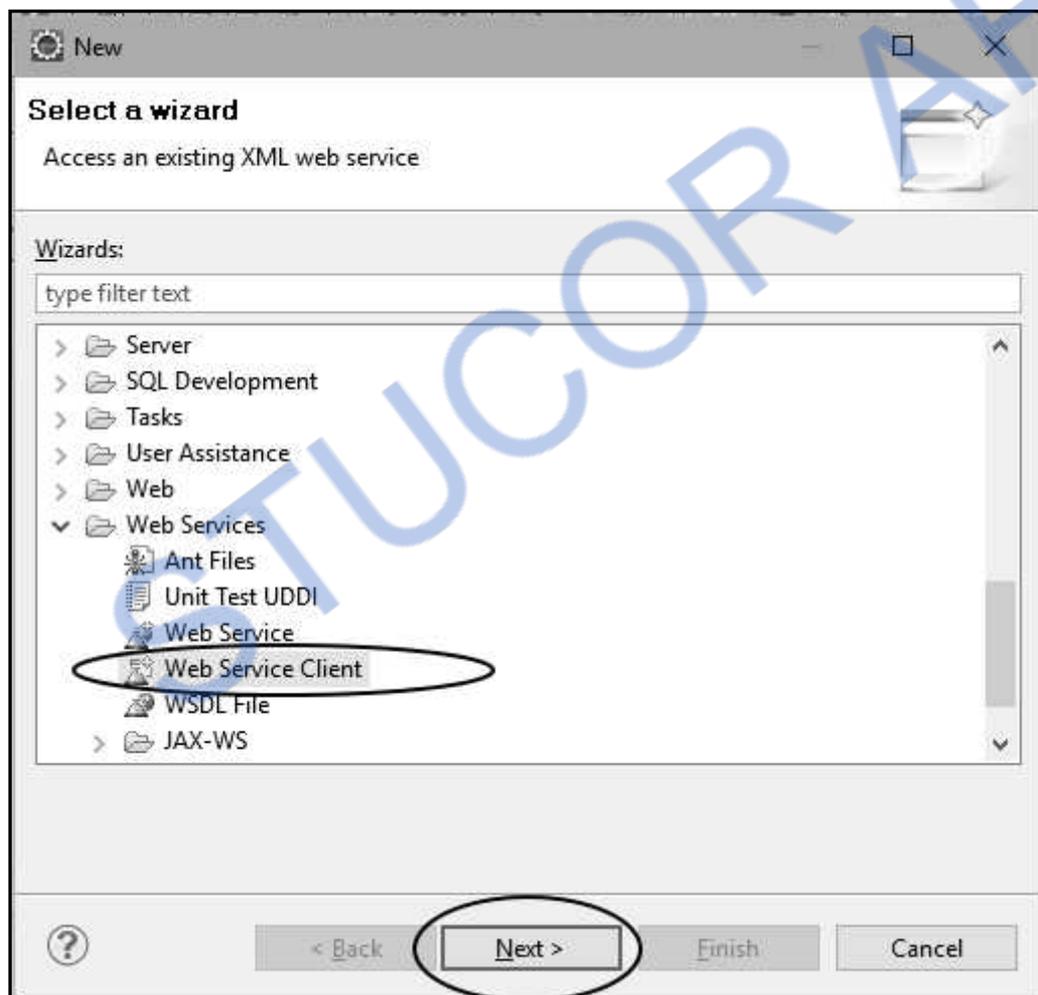
4. Just click on **MyServer** link another web page will be opened up in the browser. This is actually your WSDL file. Copy the full URL for WSDL file marked in screenshot given below. Save this URL in some Notepad file. Note that: We need this URL during the creation of Client.



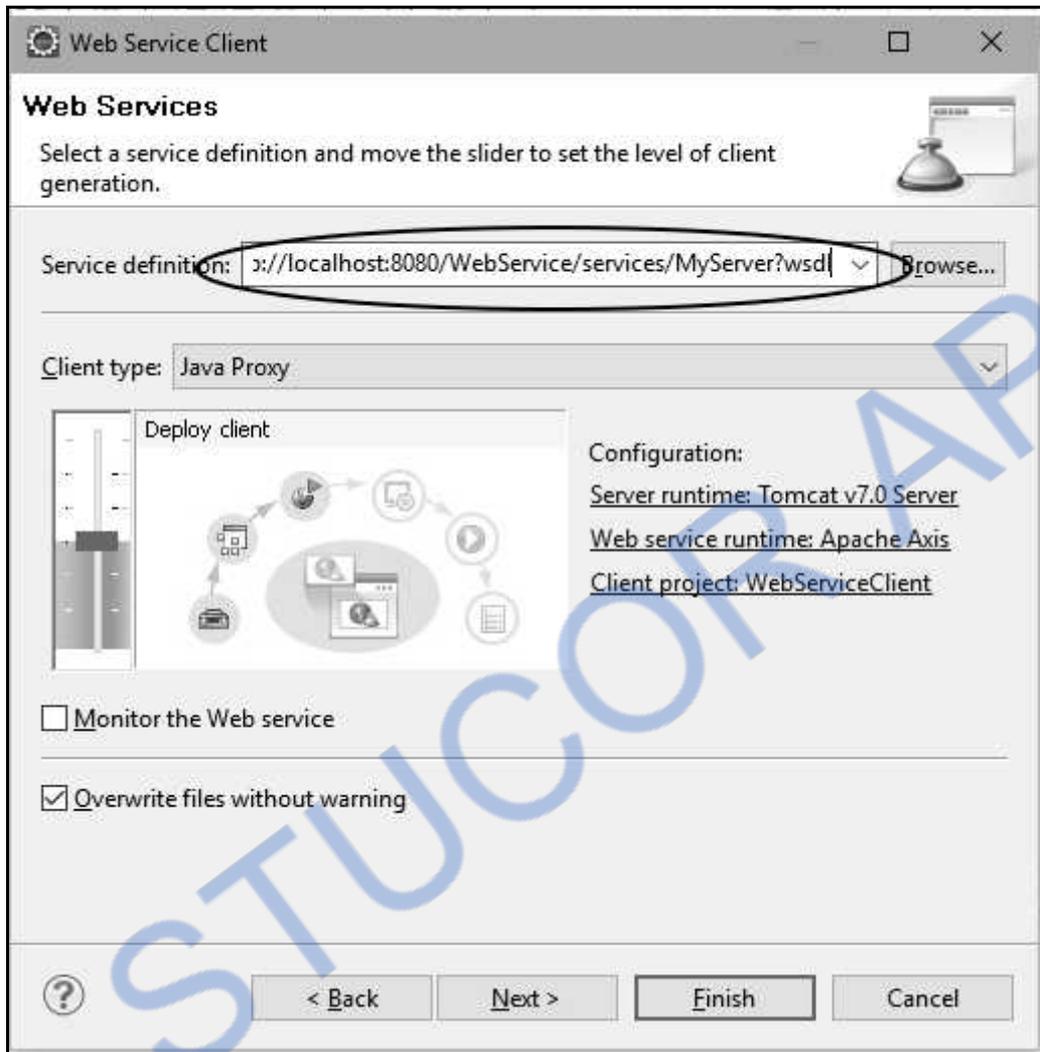
Stage II: Creating Client for Web Service

Step 6 : Now go to **File->New->Dynamic Web Project**. Give suitable name for your application. Here we are creating a client for the Web service created in above steps. Hence I have given the name **WebServiceClient**. Then click **Finish** button

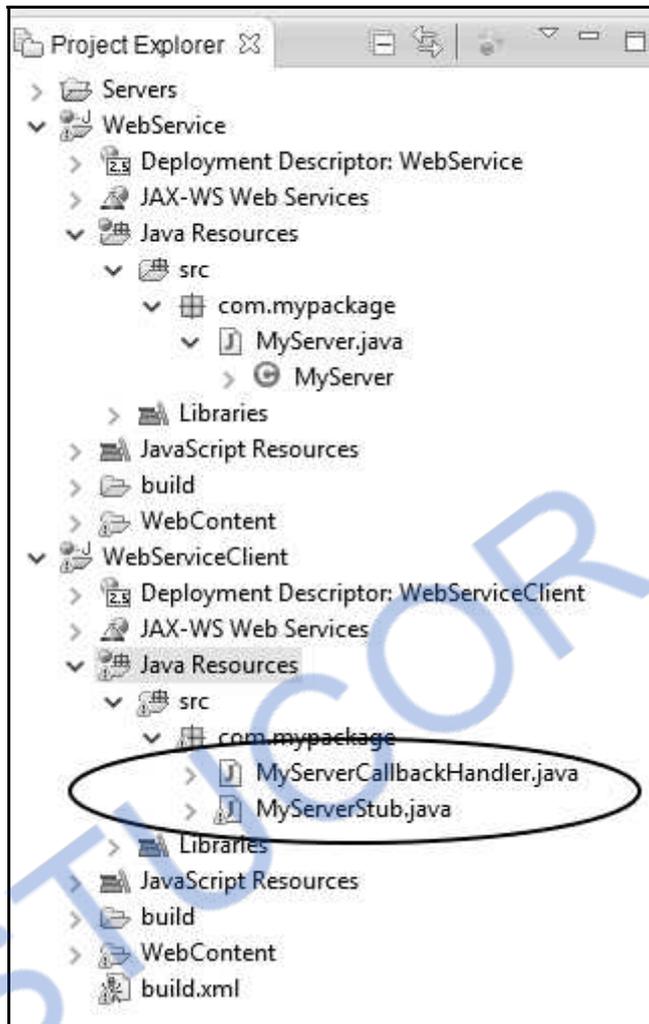
Step 7 : Now under **WebServiceClient**, right click on **src** folder. Click on **New->Other-> Web Services->Web Service Client**



Click the **Next** button. Now on **Web Services** window. Copy the URL of WSDL which you have copied in Step 5(4).



Then Select **Web Service runtime:Apache Axis**. Set it to **Apache Axis2** click **OK** button. Then Click on **Finish** button. You will get Progress Information and then two files will get generated. These are as shown in following screenshot



Step 8 : Now right click on **src** folder of **WebServiceClient**. Click on **New->Class**. The **New Java Class** window will appear. Set appropriate package name, class name and Check for void main method. Here is the screenshot

New Java Class

Create a new Java class.

Source folder:

Package:

Enclosing type:

Name:

Modifiers: public default private protected
 abstract final static

Superclass:

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)
 Constructors from superclass
 Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
 Generate comments

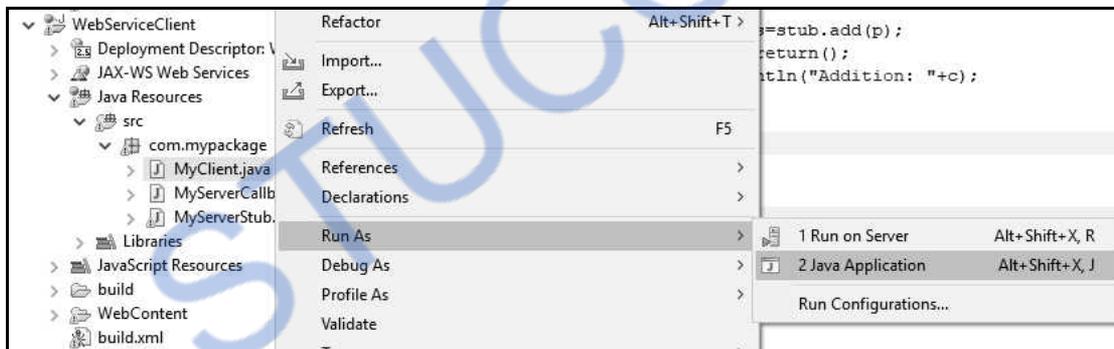
Click on **Finish** button. You can now write the Java Client code for web service. The code is as follows

MyClient.java

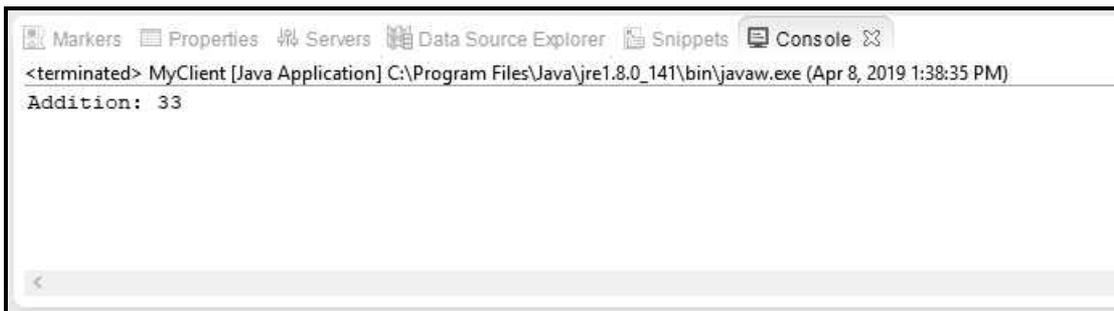
```
package com.mypackage;  
import com.mypackage.MyServerStub.Add;  
import com.mypackage.MyServerStub.AddResponse;
```

```
public class MyClient {
    /**
     * @param args
     */
    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub
        MyServerStub stub=new MyServerStub();
        Add p=new Add();
        p.setA(11);
        p.setB(22);
        AddResponse res=stub.add(p);
        int c=res.get_return();
        System.out.println("Addition: "+c);
    }
}
```

Step 8 : Right click on **MyClient.java** file in Project Explorer. Click on **Run As -> Java Application**



You will get the output at the Console window as follows



Troubleshooting :

We may get following error on running the above client

Exception in thread "main" org.apache.axis2.AxisFault:...

...

...

To overcome through this Exception, we need to make some changes in **services.xml** file. It can be located in the Web Services folder. To locate this **xml** file, follow the path as

Web Content-> WEB-INF-> Services->MyServer->META-INF. Locate services.xml

Change following lines

```
<messageReceivermep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
<messageReceivermep="http://www.w3.org/2004/08/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
```

To

```
<messageReceivermep="http://www.w3.org/ns/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
<messageReceivermep="http://www.w3.org/ns/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
```

Stop the Apache Tomcat Server and start it again. Now Run your **MyClient.java** file as Java Application and you will get the required output.

University Questions

1. Explain the creation of Java web service in detail with examples.

AU : May-12, Marks 16

2. Explain in detail the steps in the creation, publishing and testing of a web services in detail.

AU : Dec.-13, Marks 16

10.4 Describing Web Services: WSDL

AU : Dec.-11,15, May-14,16, Marks 16

- The WSDL is a Web Service Descriptor Language which is based on XML.
- The purpose of this file is to describe the web service and the method of accessing it.
- WSDL is a W3C recommendation.
- It contain various definitions. Following elements are described in WSDL file.

Elements	Description
types	It specifies the data types of the symbols used by the web service.
messages	It specifies the messages used by the web service.
portType	It specifies the name of the operation(function) defined by the web service
binding	It specifies the name of the protocol of the web service, typically it is SOAP.

1) Types

If we declare and use some variable in the web service then their data types can be specified.

For example -

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SimpleCalculator" targetNamespace="http://tempuri.org/wsdl" >
  <types>
    <schema targetNamespace="http://tempuri.org/types">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType name="CalciClass">
        <sequence>
          <element name="operator1" type="double"/>
          <element name="operator2" type="double"/>
          <element name="operation" type="String"/>
        </sequence>
      </complexType>
    </schema>
  </types>
```

2) Messages

The messages define the **part** of the element along with the data types. The message name specified the name of the function.

In following example the **HelloFun** is a name of the function which is declared in the interface named **MyInterface**. The return parameter is specified by **name="MyInterface_HelloFunResponse"**.

For example -

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SimpleHelloMessage" targetNamespace="http://tempuri.org/wsdl" > <types/>
  <message name="MyInterface_HelloFun">
    <part name="String_1" type="xsd:string"/>
  </message>
  <message name="MyInterface_HelloFunResponse">
    <part name="result" type="xsd:string"/>
  </message>
```

3) PortTypes

This is an important element of the WSDL file. It defined various **operations** that can be performed by the web service.

Various operation types are -

Elements	Description
one-way	This operation will send the message but it will not get any response.
request-response	This operation will send a message and will receive a response.
Solicit-response	This operation will send a message and will wait for a response.
Notification	This operation will send a message and will not wait for a response.

The messages in portTypes define the name of the operation along with the prefix **tns** where tns stands for TypeNameSpace.

4) Binding

The binding defines the message format and details of the protocol.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SimpleHelloMessage" targetNamespace="http://tempuri.org/wsdl" > <types/>
...
...
<binding name="MyInterfaceBinding" type="tns:MyInterface">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
  <operation name="HelloFun">
...
...
...

```

There are three main elements -

binding

The binding defines **name** which defines the name of the binding and **type** attributes specifies the port which is typically the name of the interface file.

soap:binding

The soap:binding defines **transport** and **style** attributes. The style is remote procedure call which allows the distributed clients to access the web service.

operation

The operations defines the port operations. Each operation contains the **soap:operation**, **input** and **output** elements. The **soap:body** element provides the communication details such as whether or not the associated message is encoded and if it is encoded then the **encodingStyle** is specified.

5) Service

The service element specifies the name for overall web service.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SimpleHelloMessage" targetNamespace="http://tempuri.org/wsdl" >
  <types/>
  ...
  ...
  ....
  <service name="SimpleHelloMessage">
    <port name="MyInterfacePort" binding="tns:MyInterfaceBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
    </port>
  </service>
</definitions>
```

University Questions

- | | |
|--|-------------------------------|
| 1. Describe the significance and working of WSDL with an example. | AU : Dec.-11, Marks 16 |
| 2. Explain WSDL structure and its elements. | AU : May-14, Marks 8 |
| 3. Write short note on WSDL . | AU : Dec.-15, Marks 8 |
| 4. Describe the structure of a WSDL document, its elements and their purposes with appropriate examples. | AU : May-16, Marks 16 |

10.5 Representing Data Types: XML Schema

AU : May-12, Dec.-12, Marks 8

- The XML schemas are used to represent the **structure of XML document**.
- The goal or purpose of XML schema is to define the building blocks of an XML document. These can be used as an alternative to XML DTD.
- The XML schema language is called as **XML Schema Definition (XSD) language**.
- XML schema defines elements, attributes, elements having child elements, order of child elements. It also defines fixed and default values of elements and attributes.
- XML schema also allows the developer to use **data types**.

How to write a simple schema ?

Step 1 : We will first write a simple xsd file in which the desired structure of the XML document is defined. I have named it as *StudentSchema.xsd*

This file contains the complex type with four child simple type elements.

XML Schema [StudentSchema.xsd]

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```

        <xs:element name="Student">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="std" type="xs:string"/>
            <xs:element name="marks" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>

```

Script Explanation :

- 1) The **xs** is qualifier used to identify the schema elements and types.
- 2) The document element of schema is **xs:schema**. The **xs:schema** is the root element. It takes the attribute **xmlns:xs** which has the value **http://www.w3.org/2001/XMLSchema**. This declaration indicates that document should follow the rules of XML schema. The XML schema rules are defined by the W3 recommendation in year 2001.
- 3) Then comes **xs:element** which is used to define the xml element. In above case the element **Student** is of complex type who have four child elements : *name*, *address*, *std* and *marks*. All these elements are of simple type **string**.

Step 2 : Now develop XML document in which the desired values to the XML elements can be given. I have named this file as **MySchema.xml**

XML Document [*MySchema.xml*]

```

<?xml version="1.0" encoding="UTF-8"?>
<Student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="StudentSchema.xsd">
    <name>Anand</name>
    <address>Pune</address>
    <std>Second</std>
    <marks>70 percent</marks>
</Student>

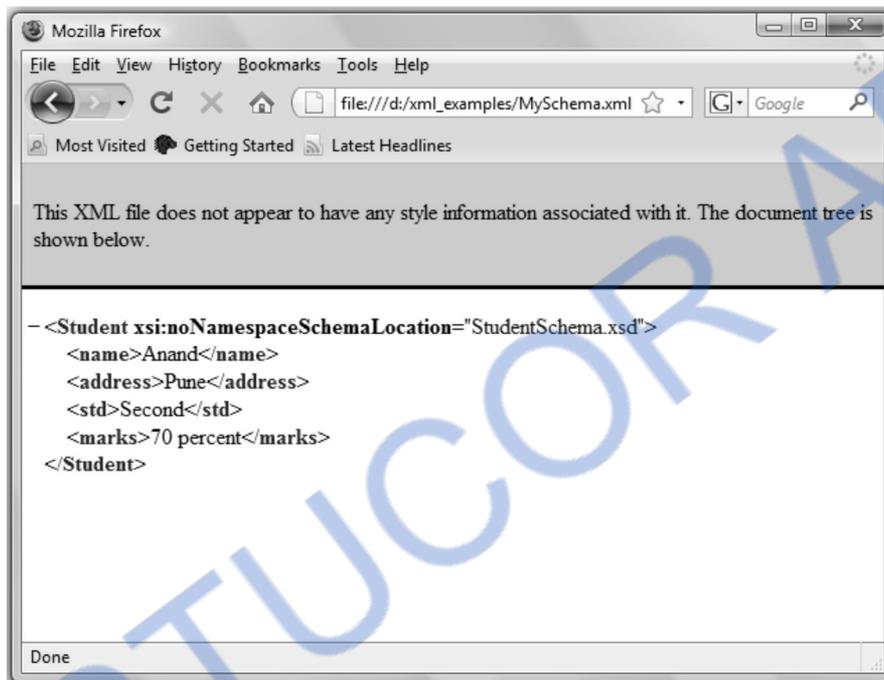
```

Program Explanation :

- 1) In above XML document, the first line is typical in which the version and encoding is specified.
- 2) There is **xmlns:xsi** attribute of document which indicates that XML document is an instance of XML schema. And it has come from the namespace "http://www.w3.org/2001/XMLSchema-instance."

- 3) To tie this XML document with the some Schema definition we use the attribute **xsi:noNamespace SchemaLocation**. The value that can be passed to this attribute is the name of xsd file “StudentSchema.xsd”.
- 4) After this we can define the child elements.

Step 3 : See the output in the browser window as follows -



10.5.1 Data Types

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

- String
- Date
- Numeric
- Boolean

Let us discuss each of these data types with the help of examples -

1. String data type

The string data type is used to define the **element** containing characters, lines, tabs or white spaces. Following schema definition illustrates this data type.

We can write a schema definition in which the data type of element **Student_Name** is declared as of string type.

XML Schema [stringType.xsd]

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

2. Date data type

For specifying the date we use date data type. The format of this date is yyyy-mm-dd where yyyy denotes the year, mm denotes the month and dd specifies the day. While specifying the date if any entry is single digit then it must be with leading zero. All the entries (i.e. yyyy,mm and dd) must be present.

Here is an illustration.

XML Schema [dateType.dtd]

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

XML Document [dateTypeDemo.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<Date_of_Birth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dateType.xsd">
2009-09-01
</Date_of_Birth>
```

We can also specify the time by specifying the time as follows -

XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<Date_of_Birth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dateType.xsd">
2009-09-01-01:57
</Date_of_Birth>
```

3. Numeric

If we want to use numeric value for some element then we can use the data types as either **decimal** or **integer**.

XML Schema [decimal Type.xsd]

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

XML Document [decimal Type Demo.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<My_Marks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="decimalDemo.xsd">
99.99
</My_Marks>
```

Using decimal data type we can specify the value that may contain some decimal point. We can have another data type called integer which is used to specify the numeric values that are without any decimal point.

XML Schema [integer Type.xsd]

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

4. Boolean

For specifying the true or false values we must use the boolean data type.

XML Schema [booleantype.xsd]

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

10.5.2 Advantages and Disadvantages of Schema**Advantages of schema**

1. The schemas are more specific.
2. The schema provide the support for data types .
3. The schema is aware of namespaces.
4. The XML schema is written in XML itself and has a large number of built in and derived types.
5. The XML schema is the W3C recommendation. Hence it is supported by various XML validator and XML processors.

Disadvantages of schema

1. The XML schema is complex to design and hard to learn.
2. The XML document cannot be if the corresponding schema file is absent.
3. Maintaining the schema for large and complex operations sometimes slows down the processing of XML document.

Advantages of schema over DTD

1. Both the schemas and DTDs are useful for defining structural components of XML. But the DTDs are basic and cannot be much specific for complex operations. On the other hand schemas are more specific.
2. The schemas provide support for defining the type of data. The DTDs do not have this ability. Hence content definition is possible using schema.
3. The schemas are namespace aware and DTDs are not.
4. The XML schema is written in XML itself and has a large number of built in and derived types.
5. The schema is the W3C recommendation. Hence it is supported by various XML validator and XML processors but there are some XML processors which do not support DTD.
6. Large number of web applications can be built using XML schema. On the other hand relatively simple and compact operations can be built using DTD.

University Questions

1. Explain the role of XML schema in building web services in detail.
2. Briefly discuss how data types are represented in XML schema.

AU : May-12, Marks 8

AU : Dec.-12, Marks 8

10.6 Communicating Object Data: SOAP

AU : May-11, Marks 8, Dec.-11,12, 13,15,16 ,Marks 16

Simple Object Access Protocol (SOAP) is a protocol based on XML. It is used by the web services for exchange of information. It is a W3C recommendation.

This exchange of information is done over HTTP. This is a platform and language independent protocol.

The client-server communication is based on RPC. The HTTP does not designed to handle the distributed objects that are required by the RPC. Hence another application protocol is build over HTTP which popularly known as SOAP. SOAP allows to talk different applications that are running in two different operating systems.

10.6.1 Structure of SOAP

The structure of the SOAP is defined by four building blocks

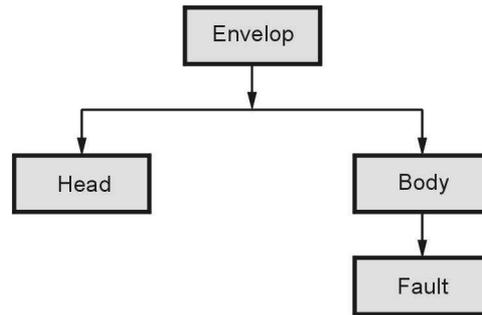


Fig. 10.6.1 SOAP building blocks

1) Envelop

The Envelop is the root of any SOAP document. It identifies that the XML document as a SOAP message. It defines two things - **Namespace** and **encodingStyle**. The Namespace value can be given as -

`xmlns:soap="http://www.w3.org/2001/12/soap-envelope".`

The encoding style is useful to define the data types used in the document. For example the value of encoding can be given as -

`soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding"`

2) Header

The SOAP header field is optional. It contains the header information. The header contains three attributes such as **mustUnderstand**, **actor** and **encodingStyle**

The **mustUnderstand** attribute specifies 0 or 1. If the value is 1 then it means that the receiver should process the header.

Normally the SOAP message travels over a message path having the endpoints at the two ends. The SOAP **actor** is used to specify the URI for this endpoint.

The **encodingStyle** is useful to define the data types used in the document

The SOAP header defines how the recipient should process the SOAP message.

3) Body

The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.

4) Fault

This is optional element of the SOAP message. It is used to represent the error code. It consists of elements such as **faultcode** which is a code for fault. The **faultstring** message gives the details about the fault.

10.6.2 SOAP and HTTP

HTTP communicates over TCP/IP. HTTP is based on Request-Response mechanism. SOAP is a HTTP Request-Response protocol. SOAP is a combination of HTTP and XML.

The SOAP request can be HTTP GET or HTTP POST request. Each request consists of two HTTP headers :**content-type** and **content-length**. The content-type specifies the MIME type and the content-length specifies the size of the message in bytes. After this header the body part begins which defines SOAP Envelop, Header, Body and Fault.

Example

The SOAP request can be

```
POST /mycreation HTTP/1.0
Host: www.mywebsite.com
Content-Type: text/xml; charset=utf-8
Content-Length: 500
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <SOAP-ENV:Body xmlns:m="http://www.mywebsite.com/mycreation">
    <m:GetDetails>
      <m:Name>ABCD</m:Name>
    </m:GetDetails>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP response can be

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 500
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <SOAP-ENV:Body xmlns:m="http://www.mywebsite.com/mycreation">
    <m:GetDetailsResponse>
      <m:ItemDetails>This is a my new creation</m:ItemDetails>
    </m:GetDetailResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.6.3 SOAP Encoding

SOAP makes use of XML Schema to encode the programming constructs. Following are examples of SOAP encoding.

1. Encoding of Struct Data

Following is a code fragment which can be encoded as follows -

```
<xs:element name="Student">
<xs:complexType>
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="RollNo" type="xs:integer"/>
    <xs:element name="address" type="tns:Address"/>
  </xs:sequence>
  <xs:element name="Address">
<xs:complexType>
  <xs:sequence>
    <xs:element name="street" type="xsd:string"/>
    <xs:element name="city" type="xsd:string"/>
    <xs:element name="country" type="xsd:string"/>
  </xs:sequence>
</xs:complexType>
</e:element>
```

The following is an example of how the preceding schema definition could be subsequently used in a SOAP encoding.

```
<e:Student>
  <Name>AAA</Name>
  <RollNo>11</RollNo>
  <address>
    <street>XYZ Road</street>
    <city>PQR City</city>
    <country>India</country>
  </address>
</e:Student>
```

2. Encoding of Arrays

An array in SOAP is of type **SOAP-ENC:Array**

```
<element name="myArray"
type="SOAP-ENC:Array"/>
```

The preceding schema can be represented in SOAP encoding as follows -

```
<myArray SOAP-ENC:arrayType="xsd:int[3]">
  <number>1</number>
  <number>2</number>
  <number>3</number>
</myArray>
```

The above encoding shows that there is an array of three integers. All these members of array are called **number**.

10.6.4 RPC Representation

- The RPC (Remote Procedure Call) is a mechanism used when the client makes a call to a method that resides on another machine. The SOAP has a support for RPC representation.
- The **style** attribute of **soap:binding** element in the WSDL document makes use of string “rpc” to denote that the communication with the client is in RPC style.

```
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"
  style="rpc"/>
```

- The **soap:body** element indicated using an encoding specified by SOAP 1.1 to represent RPC.
- In SOAP specified representation of RPC calls, the call is modeled as **struct**.
- The specified structure used in SOAP representation of RPC call has a name for the remote procedure call and has one accessor for each parameter that is to be passed to the procedure. This struct is then encoded as XML markup. This markup is then made a child of SOAP body element.

For example – Following is a soap document sent by the client to display message web service.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelop/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:ns0="http://tempuri.org/wsdl"
  xmlns:ns1="http://tempuri.org/types"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ns0:HelloMsg>
      ← Call to procedure which is RPC
      <msg_1 xsi:type="xsd:string">Welcome</msg_1>
    </ns0:HelloMsg>
  </env:Body>
</env:Envelope>
```

University Questions

1. Describe major elements of SOAP.

AU : May-11, Marks 8, Dec.-11, 13, Marks 16

2. Briefly discuss how SOAP encodes struct data and arrays.

AU : Dec.-12, Marks 8

3. Explain the SOAP elements and their RPC representation in detail.

AU : Dec.-13, Marks 16

4. Write short notes on SOAP.

AU : Dec.-15, Marks 8

5. Define SOAP and explain the building blocks of a SOAP message in detail.

AU : Dec.-16, Marks 8

10.7 Storing Java Objects as Files

AU : May-11, Marks 2

- The Object I/O supports **ObjectInputStream** and **ObjectOutputStream** classes. These classes are used to perform I/O operations for objects in addition to primitive data types.
- The **ObjectInputStream** is a subclass of **InputStream** and implements **ObjectInput** and **ObjectStreamConstants**. It is as shown in the Fig. 10.7.1 (a).

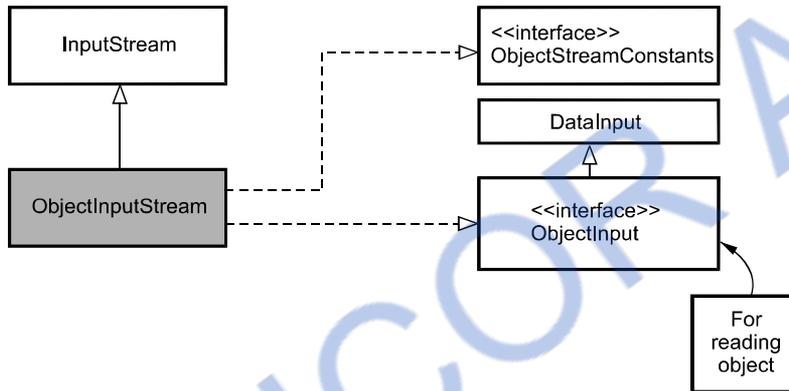


Fig. 10.7.1 (a) ObjectInputStream class

Similarly **ObjectOutputStream** is a subclass of **OutputStream** and implements **ObjectOutput** and **ObjectStreamConstants**. It is as shown in the Fig. 10.7.1 (b).

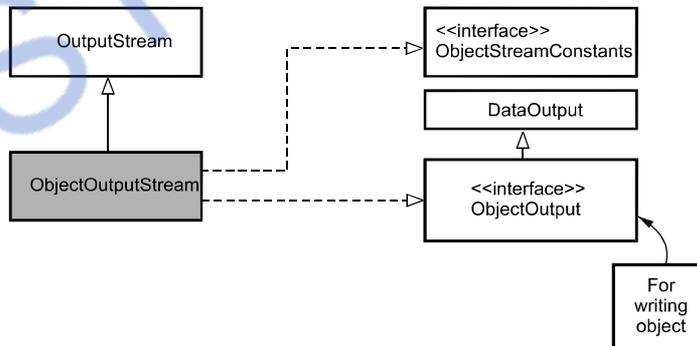


Fig. 10.7.1 (b) ObjectOutputStream class

- The syntax for ObjectInputStream and ObjectOutputStream classes is -

```

public ObjectInputStream(InputStream in)
public ObjectOutputStream(OutputStream out)

```

Following is a Java Program in which the object can be written to a file and is then retrieved back.

Java Program[ObjectDemo.java]

```
import java.io.*;
class ObjectDemo
{
    public static void main(String[] args)throws ClassNotFoundException,IOException
    {
        ObjectOutputStream o=new ObjectOutputStream(new FileOutputStream("object.txt"));
        o.writeObject(new java.util.Date()); ←Writing object to the file
        o.close();
        ObjectInputStream i=new ObjectInputStream(new FileInputStream("object.txt"));
        java.util.Date date=(java.util.Date)(i.readObject());←Reading object from the file
        System.out.println("\n The date is :"+date);
        i.close();
    }
}
```

Output

```

C:\I_0_programs>javac ObjectDemo.java
C:\I_0_programs>java ObjectDemo
The date is :Fri Jul 23 14:44:28 IST 2010
C:\I_0_programs>type object.txt
¼ø ¼sr ¼java.util.Dateh¼@KYt¼¼ xp ☺)¼ò¼¼x
C:\I_0_programs>
```

Program Explanation

In above program,

- (1) we have created a object for the **date**.
- (2) This date is written to a file named **object.txt** and then this object is retrieved using the **ObjectInputStream** class.
- (3) The date is then displayed on the console.
- (4) The **readObject** method throws the **ClassNotFoundException**, hence the main function throws both the **IOException** and **ClassNotFoundException**.

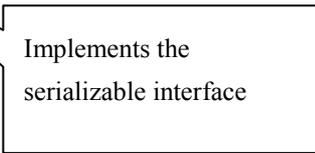
10.7.1 The Serializable Interface

- **Definition:** Serialization is the process of writing the state of the object to the byte stream.

- This technique is useful when we want to store the current state of the object to the File.
- For serialising the object we need to implement **serializable** interface. This interface does not define any member.
- **Example**

Java Program[SerializProg.java]

```
import java.io.*;
class SerializProg
{
public static void main(String[] args)throws ClassNotFoundException,IOException
{
    Myclass myobject=new Myclass("Parth",10);//creating the object
    System.out.println("The object is: "+myobject);//displaying the contents of the object
    ObjectOutputStream o=new ObjectOutputStream(new FileOutputStream("object.txt"));
    o.writeObject(myobject);//writing the object to a file
    o.flush();
    o.close();
    ObjectInputStream i=new ObjectInputStream(new FileInputStream("object.txt"));
    Myclass newobject=(Myclass)(i.readObject());//reading the object from the file
    i.close();
    System.out.println("\nYou have stored :"+newobject);//displaying the object's value
}
static class Myclass implements Serializable
{
String name;
int roll;
public Myclass(String n,int r)
{
this.name=n;
this.roll=r;
}
public String toString()
{
return "name= "+name+" and roll= "+roll;
}
}
}
```



Implements the serializable interface

Output

```

Command Prompt
F:\I_0_programs>javac SerializProg.java
F:\I_0_programs>java SerializProg
The object is: name= Parth and roll= 10
You have stored :name= Parth and roll= 10
F:\I_0_programs>

```

Program Explanation

In above program,

- (1) we have created an object of class named **Myclass**.
- (2) The members of this class are **name** and **roll number**.
- (3) From the main function assigns the values to these members.
- (4) This object is written to the file named **object.txt** using the **writeObject** method. Later on the value of this object can be read using the **readObject** method.
- (5) For the creation of the object we have to define a class who implements the **Serializable** interface.
- (6) It is necessary to throw the exceptions **ClassNotFoundException** and **IOException**

University Question

1. Define serialization .

AU : May-11, Marks 2

Two Marks Questions with Answers

Q.1 What do you means by the term web service ?

AU : May-17, Dec.-17

Ans. : The web services are the software systems that are displayed on the web browser using the web protocol. These software systems are used by some software applications rather than by end-users directly. For example: Weather forecast system, currency converters.

Q.2 What is SOAP ?

OR What is the need of SOAP ?

AU : May-17

Ans. : The SOAP stands for Simple Object Access Protocol. It is simple XML based protocol which allows applications to exchange information over HTTP. Typically web services make use of this protocol for exchange of information.

Q.3 What is meant by WSDL ?

AU : May-11, Dec.-12

OR State the use of WSDL

AU : May-12, Dec.-13, 16

Ans. : Web services provide a special file to its client in which the interface is described. This description is provided in XML document. When a client wants to make use of this web service, the users of this client read this description file and understand what input data is needed to perform the web service. Such a special file is called **WSDL** i.e Web Services Description Languages.

Q.4 List some examples of web services.

AU : May-12

Ans. : Following are some examples of web services -

1. Whether forecast system.
2. Currency converter.
3. Electronic payment processing system.
4. Credit card validation system
5. Resource management system.

Q.5 List some web service technologies.

Ans. : Following are some web service technologies -

1. Apache Axis
2. SOAP
3. JAX-RPC
4. .Net Framework.

Q.6 List the basic concepts behind JAX-RPC technology.

AU : Dec.-11

Ans. : The JAX-RPC are the higher level technologies of web services. These are JAVA API for XML (JAX) based on Remote Procedure Call (RPC).

Q.7 What is PortType in WSDL ?

Ans. : This is an important element of WSDL file. It defines various operations that can be performed by web service.

Q.8 What is the purpose of element 'binding' in WSDL ?

Ans. : The binding in WSDL specifies the name of the protocol of the web service. Typically it is SOAP.

Q.9 What is UDDI ?

AU : Dec.-11

Ans. : The UDDI stands for Universal Description, Discovery and Integration. It is XML based specification for registry of web services. It is platform independent framework.

Q.10 What are the elements of SOAP building block ?

Ans. : The following are the elements of SOAP architecture -

1. Envelop
2. Header
3. Body
4. Fault

Q.11 Explain how SOAP is related to HTTP ?

Ans. : The HTTP is a protocol which communicates over TCP/IP. It is based on request-response mechanism. SOAP is protocol which is a combination of both HTTP and XML. The SOAP request can be HTTP GET or HTTP POST. The SOAP request consists of two HTTP headers content-type and content-length the body of this request consists of SOAP Envelop, Header, Body and Fault.

Q.12 Give an example of a web service registry and its function.

AU : Dec.-12

Ans. : The Universal Description and Discovery Integration (UDDI) provides a directory of web services so that client can easily discover the services of their choices. The UDDI maintains a registry in which the web services are registered.

Q.13 What is service endpoint interface in RPC ?

AU : May-14

Ans. : A Service Endpoint Interface (SEI) is a Java interface that declares the methods that a client can invoke while using the service. The method parameters and return type specified by the SEI must be supported by the Remote Procedure Call (RPC).

Q.14 Specify how UDDI is utilized in Web Service ?

AU : May-14

Ans. : The UDDI provides a directory of web services so that client can easily discover the services of their choices. The UDDI maintains a registry in which the web services are registered.

Q.15 What is a web service ? Give any four examples.

AU : Dec.-15

Ans. : The Web Services are the software systems that are displayed by the web browser using the web protocol. These software systems are used by the some software applications rather than by end-users directly.

Examples are

1. Credit Card Validation System
2. Whether Forecast System
3. Currency Converter
4. Rate of Interest Calculator
5. Income Tax Calculator

Q.16 Draw the structure of SOAP messages.

AU : May-16

Ans. : Refer Fig. 10.6.1.

Q.17 List out data types that are available in Web-Service.

Ans : Various data types that are available in web service are - boolean, byte, short, int, long, float, double, decimal, string.

Q.18 Define complex types.

Ans. : Complex types are objects created by the developer. For example, an object called Customer with the properties CustID and name is a complex type.

Q.19 Differentiate between SOAP and HTTP.**Ans. :**

SOAP	HTTP
Simple Object Access protocol, is a protocol specification for exchanging structured information in the implementation of web services in computer networks.	The Hypertext Transfer Protocol (HTTP) is an application protocol used for communication for the World Wide Web.
It is used to build simple and useful APIs called web services.	It is used to transfer data using internet.
It uses XML information.	It relies on structured text.

Q.20 What are the different data types used in XML Schema ?**Ans. :**

Data Types	Description
String	The string is a collection of characters. This data type is used to define the elements containing characters, lines, tabs or white spaces. It is denoted by type=xs:string
Date	For specifying the date we use this data type. The format is yyyy-mm-dd where y denotes year, m denotes month and d denotes the day. It is specified using type=xs:date
Numeric	For specifying the numeric values this data type is used. This can be denoted as either type=xs:decimal or type=xs:integer
Boolean	For specifying the true or false values the Boolean data type is used. It is denoted as type=xs:boolean

Q.21 Compare DTD and Schema.**AU : May-10****Ans. :**

- Both the schemas and DTDs are useful for defining structural components of XML. But the DTDs are basic and cannot be much specific for complex operations. On the other hand schemas are more specific.
- The schemas provide support for defining the type of data. The DTDs do not have this ability. Hence content definition is possible using schema.
- The schemas are namespace aware and DTDs are not.
- The XML schema is written in XML itself and has a large number of built in and derived types.
- The schema is the W3C recommendation. Hence it is supported by various XML validator and XML processors but there are some XML processors which do not support DTD.

6. Large number of web applications can be built using XML schema. On the other hand relatively simple and compact operations can be built using DTD.

Q.22 What is the purpose of XML schema ?

AU : May-13

Ans. : The XML schemas are used to represent the structure of XML document. The purpose of XML schema is to define the building blocks of an XML document.

Q.23 Explain the term XML schema.

AU : Dec.-13

Ans. : The XML schema are used to represent the structure of XML document. The goal of XML schema is to define building blocks of an XML document.

Q.24 Why do web services need enhanced security ?

Ans. : Web services are widely used in diverse applications to fulfill the user requirements. Hence reliability of data, confidentiality and data nonrepudiation, user authentication are the most required features. Hence web services need enhanced security.



Notes

STUCOR APP

Web Technology Laboratory

Contents

<i>Experiment 1 :</i>	<i>Create a Web page with the following using HTML</i>	
	<i>i) To embed an image map in a web page.</i>	
	<i>ii) To fix the hot spots.</i>	
	<i>iii) Show all the related information when the hot spots are clicked.....</i>	<i>L - 2</i>
<i>Experiment 2 :</i>	<i>Create a web page with all types of Cascading style sheets</i>	<i>L - 3</i>
<i>Experiment 3 :</i>	<i>Client side scripts for validating web form controls using DHTML</i>	<i>L - 3</i>
<i>Experiment 4 :</i>	<i>Installation of Apache Tomcat Web Server.....</i>	<i>L - 3</i>
<i>Experiment 5 :</i>	<i>Write programs in Java using servlets</i>	
	<i>i) To invoke servlet from HTML forms</i>	
	<i>ii) Session tracking</i>	<i>L - 5</i>
<i>Experiment 6 :</i>	<i>Write programs in Java to create three tier application using JSP and Databases</i>	
	<i>i) For conducting on-line examination</i>	
	<i>ii) For displaying student marks list. Assume that student information is available in a database which has been stored in database server.....</i>	<i>L - 5</i>
<i>Experiment 7 :</i>	<i>Programs using XML- Schema-XSLT/XSL</i>	<i>L - 13</i>
<i>Experiment 8 :</i>	<i>Programs using DOM and SAX parsers.</i>	<i>L - 13</i>
<i>Experiment 9 :</i>	<i>Programs using AJAX.....</i>	<i>L - 13</i>
<i>Experiment 10 :</i>	<i>Consider a case where we have two web services – an airline service and a travel agent and the travel agent is searching for an airline. Implement this scenario using web services and Database.....</i>	<i>L - 13</i>

Experiment 1 : Create a Web page with the following using HTML**i) To embed an image map in a web page.****ii) To fix the hot spots.****iii) Show all the related information when the hot spots are clicked.****Sol. :**

```

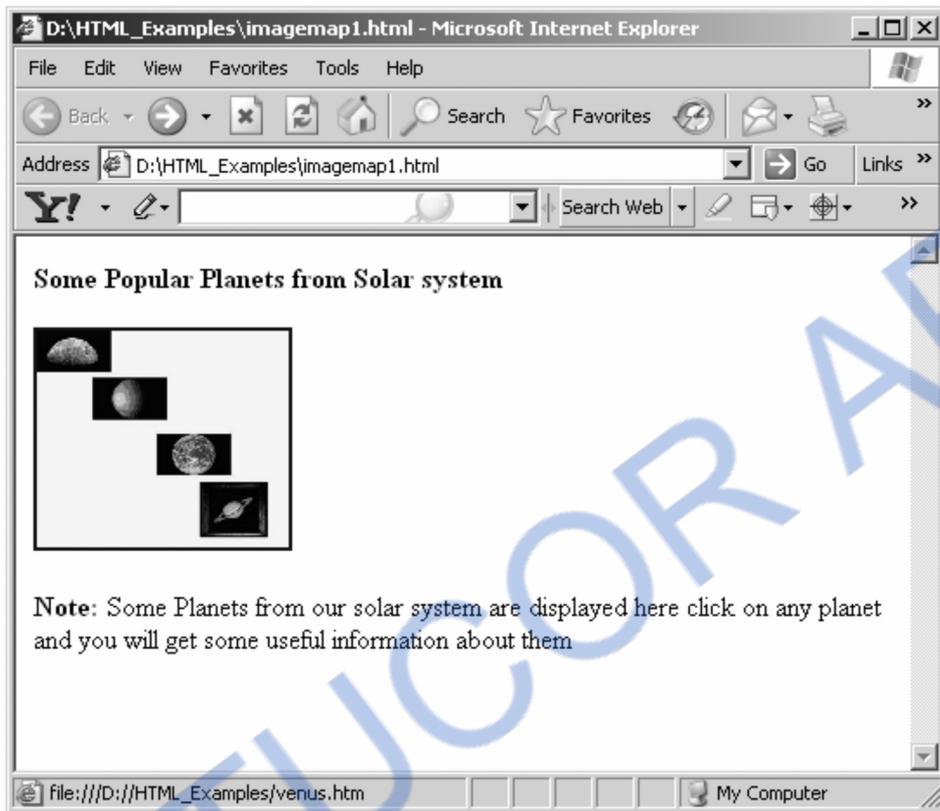
<html>
<body>
<p>
<b>Some Popular Planets from Solar system</b>
</p>

<map id="solarmap" name="solarmap">
<area shape="rect"
coords="10,10,40,25"
alt="Mercury"
href="D:\\HTML_Examples\Mercury.html">
<area shape="rect"
coords="35,20,75,55"
alt="venus"
href="D:\\HTML_Examples\venus.htm">
<area shape="rect"
coords="65,55,115,85"
alt="Earth"
href="D:\\HTML_Examples\earth.htm">
<area shape="rect"
coords="90,85,135,120"
alt="Saturn"
href="D:\\HTML_Examples\saturn.htm">
</map>
<p><b>Note:</b> Some Planets from our solar system are displayed here click on any planet and
you will get some useful information about them</p>
</body>
</html>

```

This sets the image width and height.

Hot spot for each planet is set.

Output

Experiment 2 : Create a web page with all types of Cascading style sheets

Sol. : Refer section 3.5.

Experiment 3 : Client side scripts for validating web form controls using DHTML

Sol. : Refer example 4.13.4.

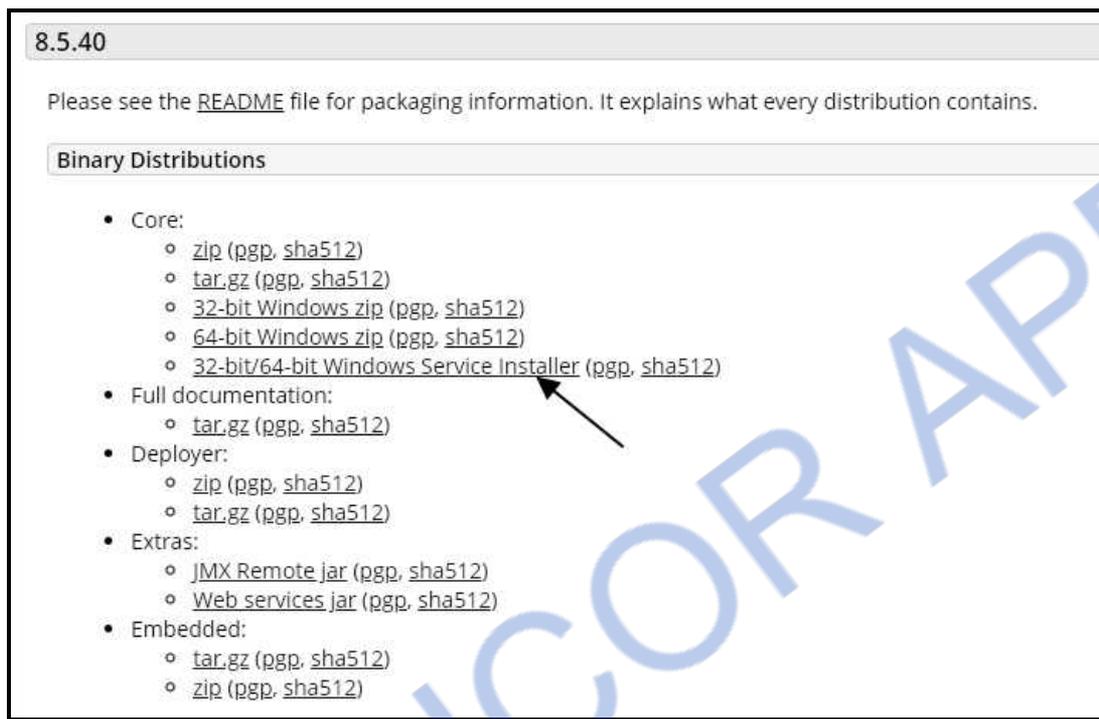
Experiment 4 : Installation of Apache Tomcat Web Server.

Sol. :

Step 1 : Download the Apache Tomcat web server from the site

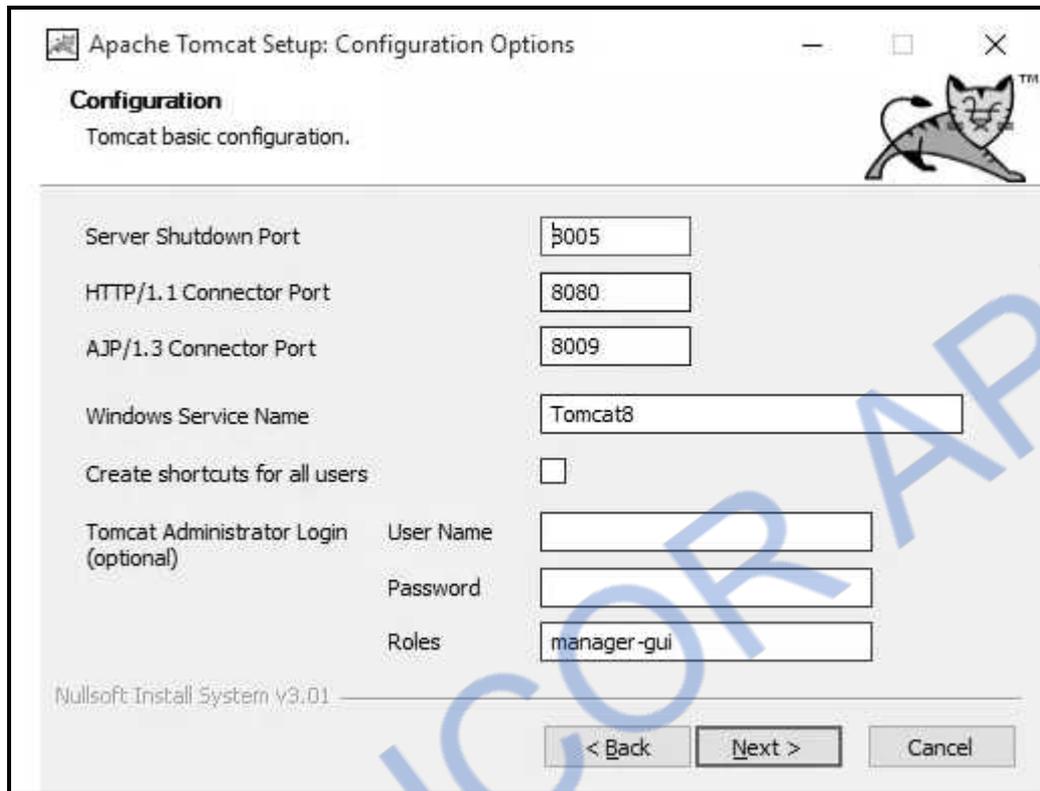
<https://tomcat.apache.org/download-80.cgi>

Step 2 : Click on the link mentioned in the following screenshot



Step 3 : Install the downloaded file and proceed by clicking the next button until you reach configuration option. Here you can set the name of the service, shutdown port and running port of Tomcat. By default Tomcat runs on port 8080.

Step 4 : After completing the installation, Tomcat folder can be present at C:\Program Files \ Apache Software Foundation\Tomcat 8.5.



We can start up Tomcat using **startup.bat** file and stop the tomcat using **shutdown.bat** file.

Experiment 5 : Write programs in Java using servlets –

- i) To invoke servlet from HTML forms
- ii) Session tracking

Sol. :

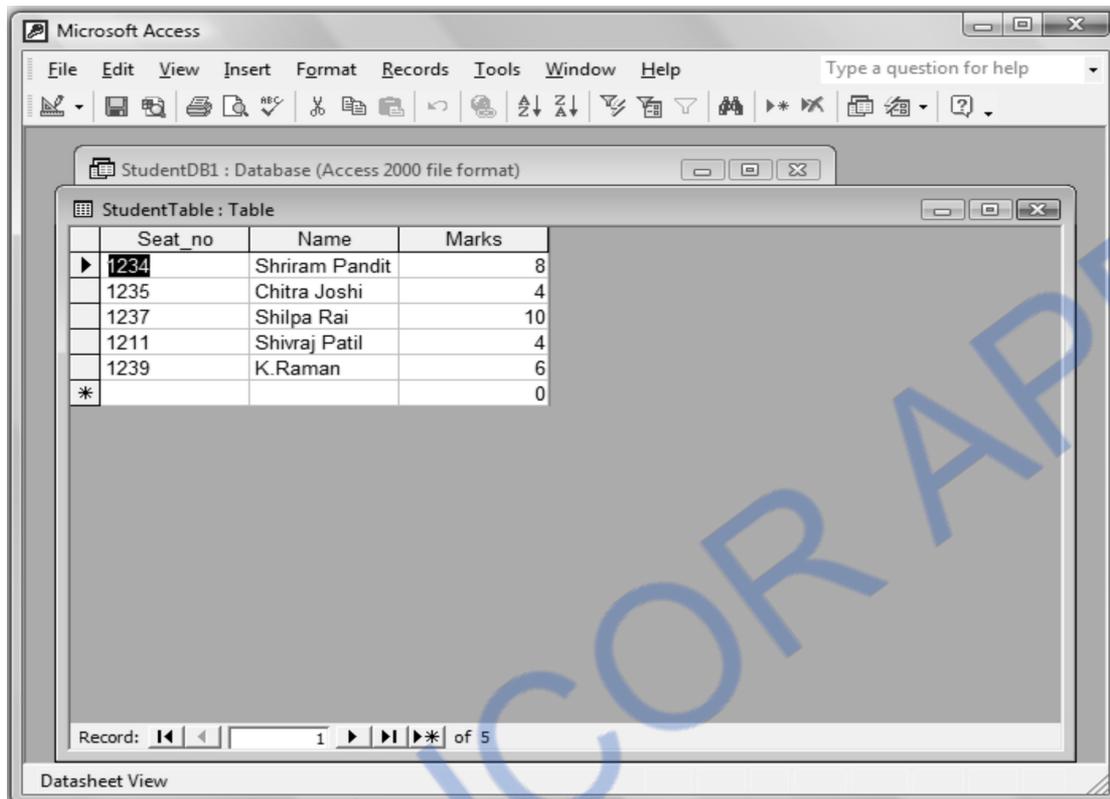
- i) Refer example 6.6.3.
- ii) Refer example 6.8.1.

Experiment 6 : Write programs in Java to create three tier application using JSP and Databases

- i) For conducting on-line examination
- ii) For displaying student marks list. Assume that student information is available in a database which has been stored in database server.

Sol. :

Step 1 : Create a database in MS-Access. For instance, I have created a student database in MS-Access. Inside this database create a table. This table is as follows.



Create a DSN for this database.

Step 2 : Now write a JSP document as follows -

Exam.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.util.*" %>
<%
String SeatNum,Name;
String ans1,ans2,ans3,ans4,ans5;
int a1,a2,a3,a4,a5;
a1=a2=a3=a4=a5=0;
Connection connect=null;
Statement stmt=null;
ResultSet rs=null;

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url = "jdbc:odbc:StudentDB1";
connect = DriverManager.getConnection(url, " ", " ");
if(request.getParameter("action")!=null)
```

```
{
    SeatNum = request.getParameter("Seat_no");
    Name = request.getParameter("Name");
    ans1 =request.getParameter("group1");
    if(ans1.equals("True"))
        a1=2;
    else
        a1=0;
        ans2 = request.getParameter("group2");
    if(ans2.equals("True"))
        a2=0;
    else
        a2=2;
    ans3 = request.getParameter("group3");
    if(ans3.equals("True"))
        a3=0;
    else
        a3=2;
    ans4 = request.getParameter("group4");
    if(ans4.equals("True"))
        a4=2;
    else
        a4=0;
    ans5 = request.getParameter("group5");
    if(ans5.equals("True"))
        a5=0;
    else
        a5=2;
    int Total=a1+a2+a3+a4+a5;

    stmt = connect.createStatement();
    String query = "INSERT INTO StudentTable (" + "Seat_no,Name,Marks" + ") VALUES (" +
+SeatNum + "," + Name + ","+Total+ ")";
    int result = stmt.executeUpdate(query);
    stmt.close();
    stmt = connect.createStatement();
    query = "SELECT * FROM StudentTable WHERE Name='"+Name+"'";
    rs = stmt.executeQuery(query);
%>
<html> <head> <title>Student Mark List</title> </head>
<body bgcolor=khaki>
```

```

<center>
<h2>Students Marksheet</h2>
<h3>Name of the College:ABC College of Engineering</h3>
<table border="1" cellspacing="0" cellpadding="0">
<tr>
<td><b>Seat_No</b></td>
<td><b>Name</b></td>
<td><b>Marks</b></td>
</tr>
<%
while(rs.next())
{
%>
<tr>
<td><%=rs.getInt(1)%></td>
<td><%=rs.getString(2)%></td>
<td><%=rs.getString(3)%></td>
</tr>
<%
}
rs.close();
stmt.close();
connect.close();
%>
</table>
</center>
<br/> <br/> <br/>
<table>
<tr><td><b>Date:<%=new java.util.Date().toString() %></td></tr>
<tr><td><b>Signature: X.Y.Z. <b></td></tr>
</table>
<div>
<a href="http://127.0.0.1:8080/jsp-examples/StudExam/Exam.jsp">Click here to go back</a>
</body>
</html>
<%}else{%>
<html>
<head><title>Online Examination</title>
<script language="javascript">
function validation(Form_obj)
{

```

```

        if(Form_obj.Seat_no.value.length==0)
        {
            alert("Please,fill up the Seat Number");
            Form_obj.Seat_no.focus();
            return false;
        }
    if(Form_obj.Name.value.length==0)
    {
        alert("Please,fill up the Name");
        Form_obj.Name.focus();
        return false;
    }
    return true;
    }
</script>
</head>
<body bgcolor=lightgreen>
<center>
    <h1>OnLine Examination</h1>
</center>
<form action="Exam.jsp" method="post"
name="entry" onSubmit="return validation(this)">
    <input type="hidden" value="list" name="action">
<table>
<tr>
    <td><h3>Seat Number:</h3></td>
    <td><input type="text" name="Seat_no"></td>
</tr>
<tr>
    <td><h3>Name:</h3></td>
    <td><input type="text" name="Name" size="50"></td>
</tr>
<hr/>
<tr>
    <td><b>Total Marks:10(Each question carries equal marks) </b></td>
<td></td><td></td><td><b>Time: 15 Min.</b></td>
</tr>
</table>
<hr/>
<b>1. Apache is an open source web server</b><br/>
<input type="radio" name="group1" value="True">True

```

```
<input type="radio" name="group1" value="False">False<br>
<br/>

<b>2. In Modern PC there is no cache memory.</b><br/>
<input type="radio" name="group2" value="True">True
<input type="radio" name="group2" value="False">False<br>
<br/>

<b>3. Tim-Berner Lee is the originator of Java.</b><br/>
<input type="radio" name="group3" value="True">True
<input type="radio" name="group3" value="False">False<br>
<br/>

<b>4. JPG is not a video file extension.</b><br/>
<input type="radio" name="group4" value="True">True
<input type="radio" name="group4" value="False">False<br>
<br/>

<b>5. HTTP is a statefull protocol</b><br/>
<input type="radio" name="group5" value="True">True
<input type="radio" name="group5" value="False">False<br>
<br/>

<center>
  <input type = "submit" value="Submit">
  <input type = "reset" value="Clear"><br><br>
</center>
</form>
<%}%>
```

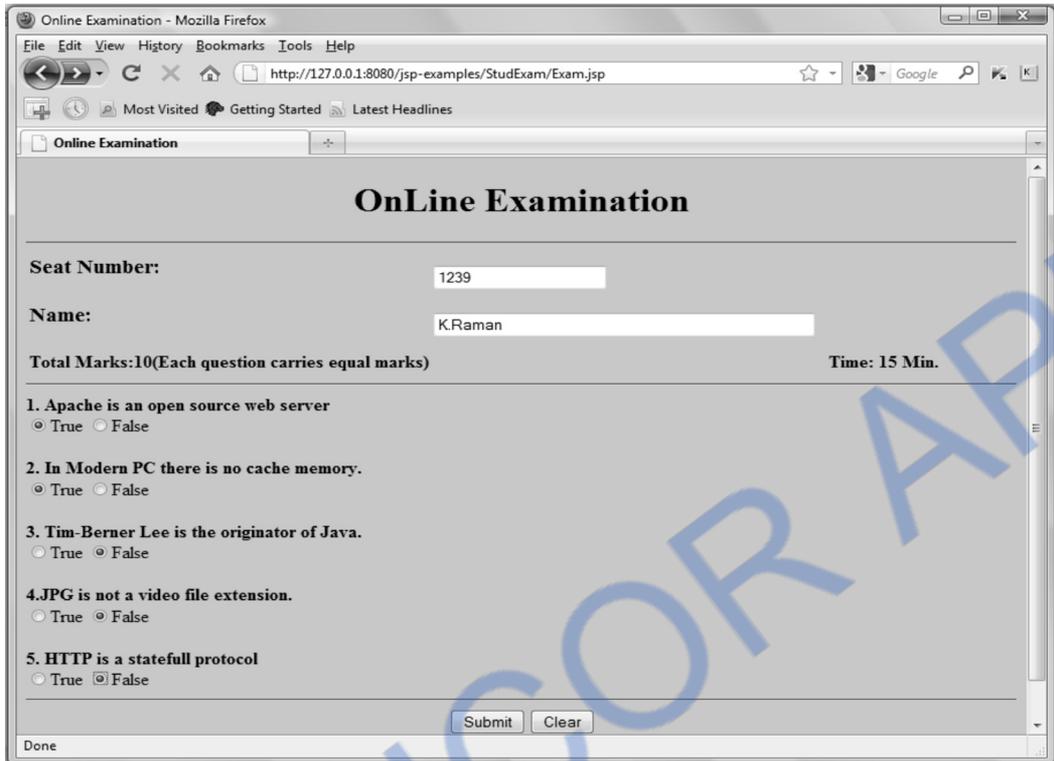
Step 3 : This file must be saved at

C : \your_tomcat_directory\webapps\jsp-examples. Create a folder at this location and in that folder store the above written JSP file. For example I have created a folder named **StudExam** at the location C:\Tomcat \jsp-examples. Create a folder at this location and saved my JSP file as **Exam.jsp**.

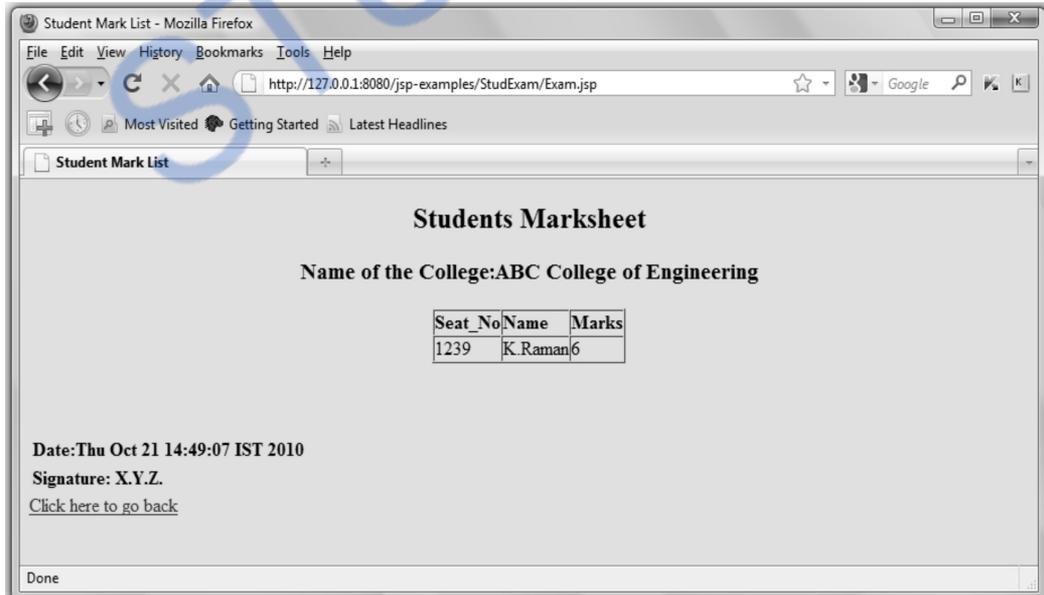
Then start running your tomcat.

Now open a suitable web browser and executed the JSP program by typing the URL

<http://127.0.0.1:8080/jsp-examples/StudExam/Exam.jsp>

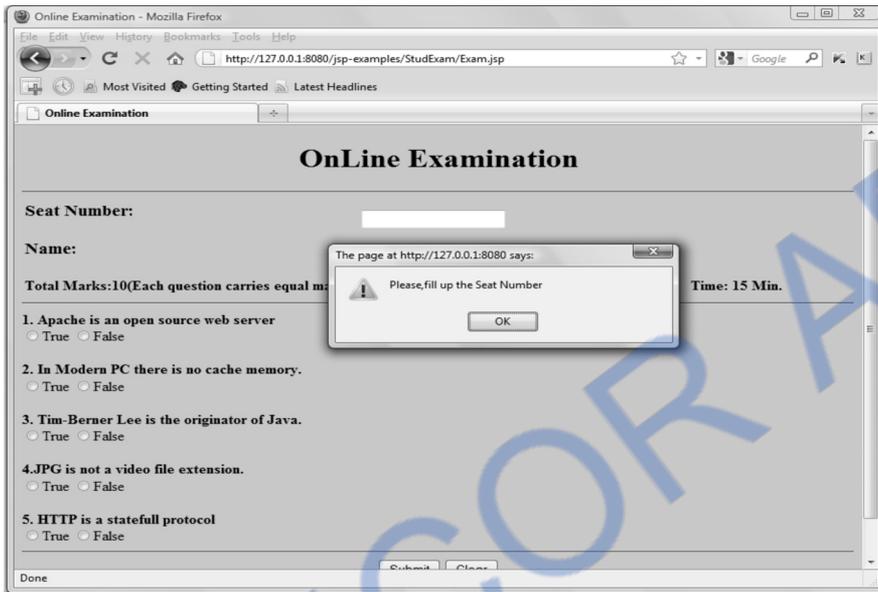


After inputting the data appropriately, Click the **Submit** button. And a Mark sheet will be displayed as follows -

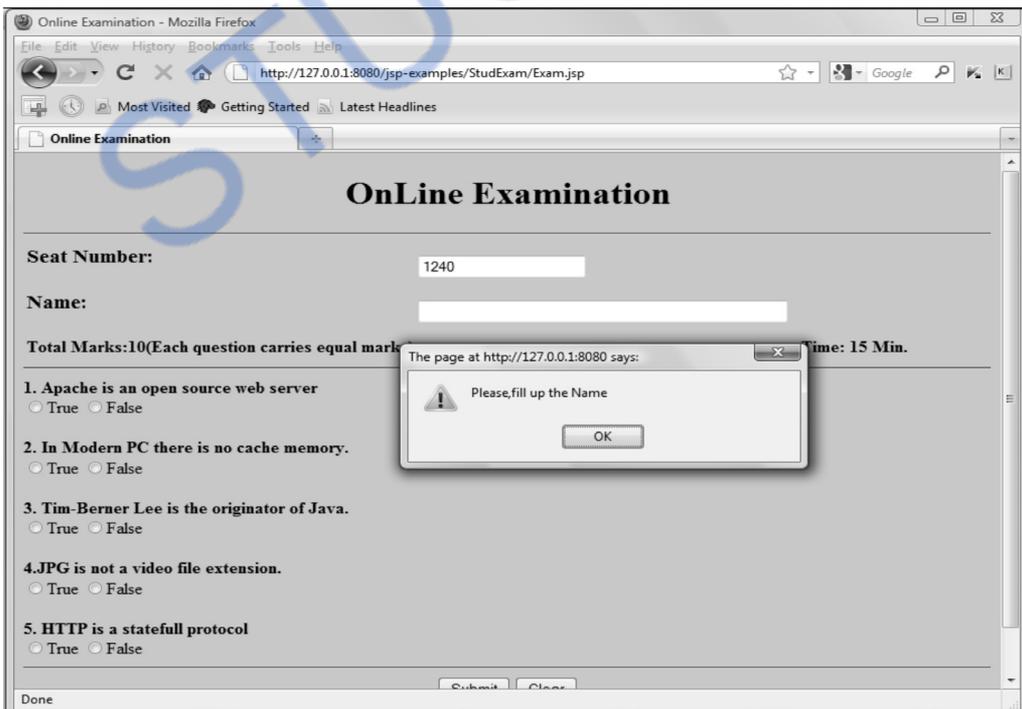


Validation

The form validation is also done in above JSP script. Following Output illustrate such validations



Similarly if you do not enter the name then



Experiment 7 : Programs using XML- Schema-XSLT/XSL

Sol. : Refer section 8.9 and 10.5.

Experiment 8 : Programs using DOM and SAX parsers.

Sol. : Refer example 8.7.1 and 8.8.1.

Experiment 9 : Programs using AJAX.

Sol. : Refer section 9.4.

Experiment 10 : Consider a case where we have two web services - an airline service and a travel agent and the travel agent is searching for an airline. Implement this scenario using web services and Database.

Sol. :

Prerequisite : For this assignment students are requested to refer section 10.3. It helps you to understand how to create a simple web service. Before attempting to this assignment, make sure that following software are already installed on your PC

- (1) JDK
- (2) XAMPP/ Apache Tomcat
- (3) MySQL
- (3) Axis2
- (4) Eclipse

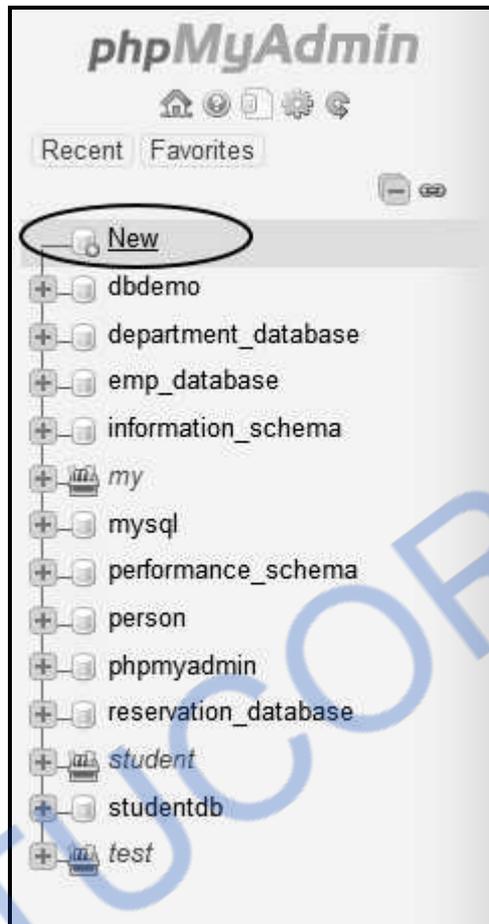
This assignment is completed in three stages -

- First stage is creation of database in MySQL,
- Second stage is creation of Web service
- Third stage is creation of Client for the created web service.

Stage I : Creating a Database

The database in MySQL can be created as follows – Note that the database name is **Reservation_database** and the table name is **Airways_table**.

- It is very simple to create a database in MySQL, if you have installed XAMPP. For creating database you simply need to start XAMP.
- Open the web browser and type the URL <http://localhost/phpmyadmin/>
- At the left hand Pane Click On **New** and create a database named **reservation_database**



Under this database the table named **Airways_table** can be created by using **Insert** Operation
The table is as follows -

```
SELECT * FROM `airways_table`
```

Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

Num	name	Start_from	Dest	Time
1001	British Airways	Pune	Chennai	12:10 p.m.
2001	AirIndia	Mumbai	Pune	4:15 a.m.
3001	Indian Airlines	Chennai	Pune	2:30 p.m.
2002	AirIndia	Chennai	Mumbai	11:45 a.m.
1002	British Airways	Chennai	Pune	11:00 a.m.
1003	British Airways	Mumbai	Pune	1:00 a.m.
3002	Indian Airlines	Pune	Mumbai	5:00 p.m.
2003	AirIndia	Pune	Mumbai	7:00 p.m.
3003	Indian Airlines	Pune	Mumbai	10:00 p.m.

Stage II : Creating Web Service

Step 1 : Start Eclipse. Click on **File->New->Dynamic Web Project**. Give the suitable name to the project.

Set the Tomcat Web Server and appropriate version number. By clicking

Window → Preferences → Server → Runtime Environment.

Step 2 : Locate **src** folder under the project in **WebService** project. Right click on **src** folder and click **New->Interface**. Give the name as **MyInterface**. The code for this interface is as given below –

MyInterface.java

```
package com.mypackage;
import java.rmi.RemoteException;
public interface MyInterface extends java.rmi.Remote
{
    public String getInfo(String src,String dest)throws java.rmi.RemoteException;
}
```

Save the above code.

Step 3 : Right click on **src** folder and click **New->Class**. Give the name as **MyServer**. The Java code for this file is as given below -

MyServer.java

package com.mypackage;

import java.sql.*;

public class MyServer **implements** MyInterface{

public String getInfo(String src,String dest)

 {

 String str="";

 Connection con=**null**;

 String url = "jdbc:mysql://localhost:3306/Reservation_database";

 String username = "root";

 String password = "";

try

 {

 Class.forName("com.mysql.jdbc.Driver");

 con = DriverManager.getConnection(url, username, password);

 Statement s=con.createStatement();

if(src.equals("Mumbai"))

 {

if(dest.equals("Pune"))

 s.executeQuery("SELECT * FROM Airways_table WHERE Start_from='Mumbai' AND Dest='Pune'");

else if(dest.equals("Chennai"))

 s.executeQuery("SELECT * FROM Airways_table WHERE Start_from='Mumbai' AND Dest='Chennai'");

 }

else if(src.equals("Pune"))

 {

if(dest.equals("Mumbai"))

 s.executeQuery("SELECT * FROM Airways_table WHERE Start_from='Pune' AND Dest='Mumbai'");

else if(dest.equals("Chennai"))

 s.executeQuery("SELECT * FROM Airways_table WHERE Start_from='Pune' AND Dest='Chennai'");

 }

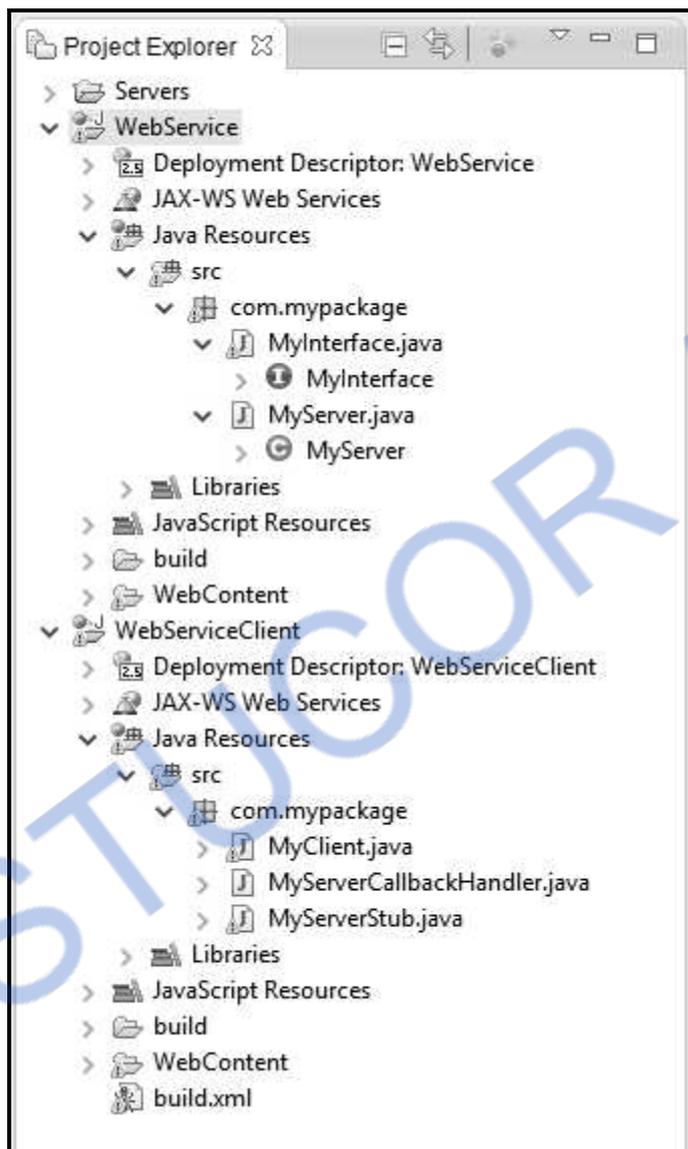
```
    else if(src.equals("Chennai"))
    {
        if(dest.equals("Pune"))
            s.executeQuery("SELECT * FROM Airways_table WHERE Start_from='Chennai' AND
Dest='Pune'");
        else if(dest.equals("Mumbai"))
            s.executeQuery("SELECT * FROM Airways_table WHERE Start_from='Chennai' AND
Dest='Mumbai'");
    }
    ResultSet rs=s.getResultSet();
    str+ "<table border=1>";
    while(rs.next())
    {
        str+ "<tr><b>";
        str+ "<td>";
        str+=rs.getString("name");
        str+ "</td>";
        str+ "<td>";
        str+=rs.getString("Time");
        str+ "</td>";
        str+ "</tr></b>";
    }
    str+ "</table>";

} catch(ClassNotFoundException ex)
{
    System.out.println(ex);
}
catch(SQLException ex)
{
    System.out.println(ex);
}
return str;
}
}
```

Save the above code.

The snapshot of Project Explorer is as shown as follows :

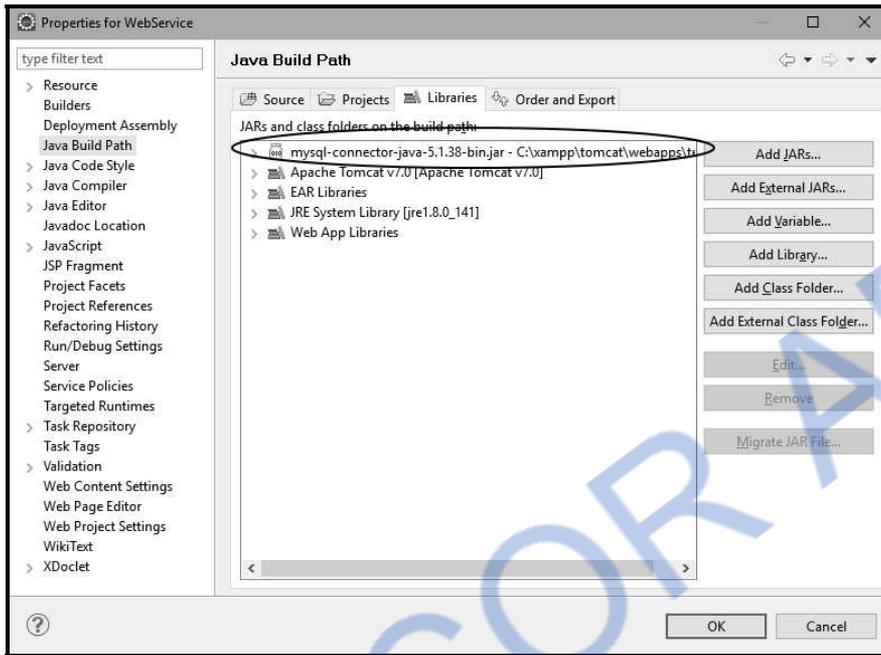
Project Explorer



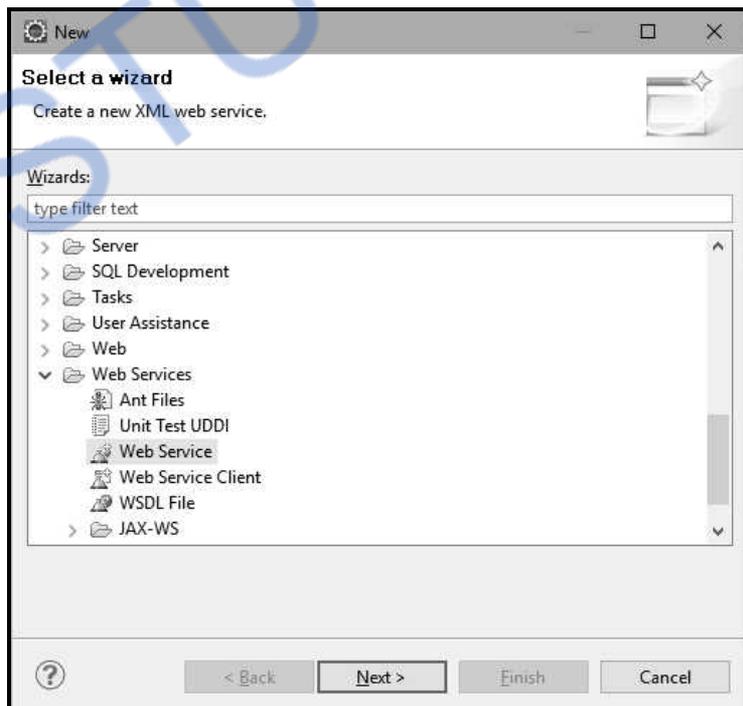
To establish connectivity between Database(Created in Stage I) and above Servlet created in Eclipse, we need some configuration. This can be done in following steps -

- Download mysql-connector-javaXXX.jar file from internet at suitable location.
- Right click on your Project.
- Click on **Properties->Java Build Path->Libraries**.
- Click Add External JARs... button and select **mysql-connector-javaXXX.jar** file.

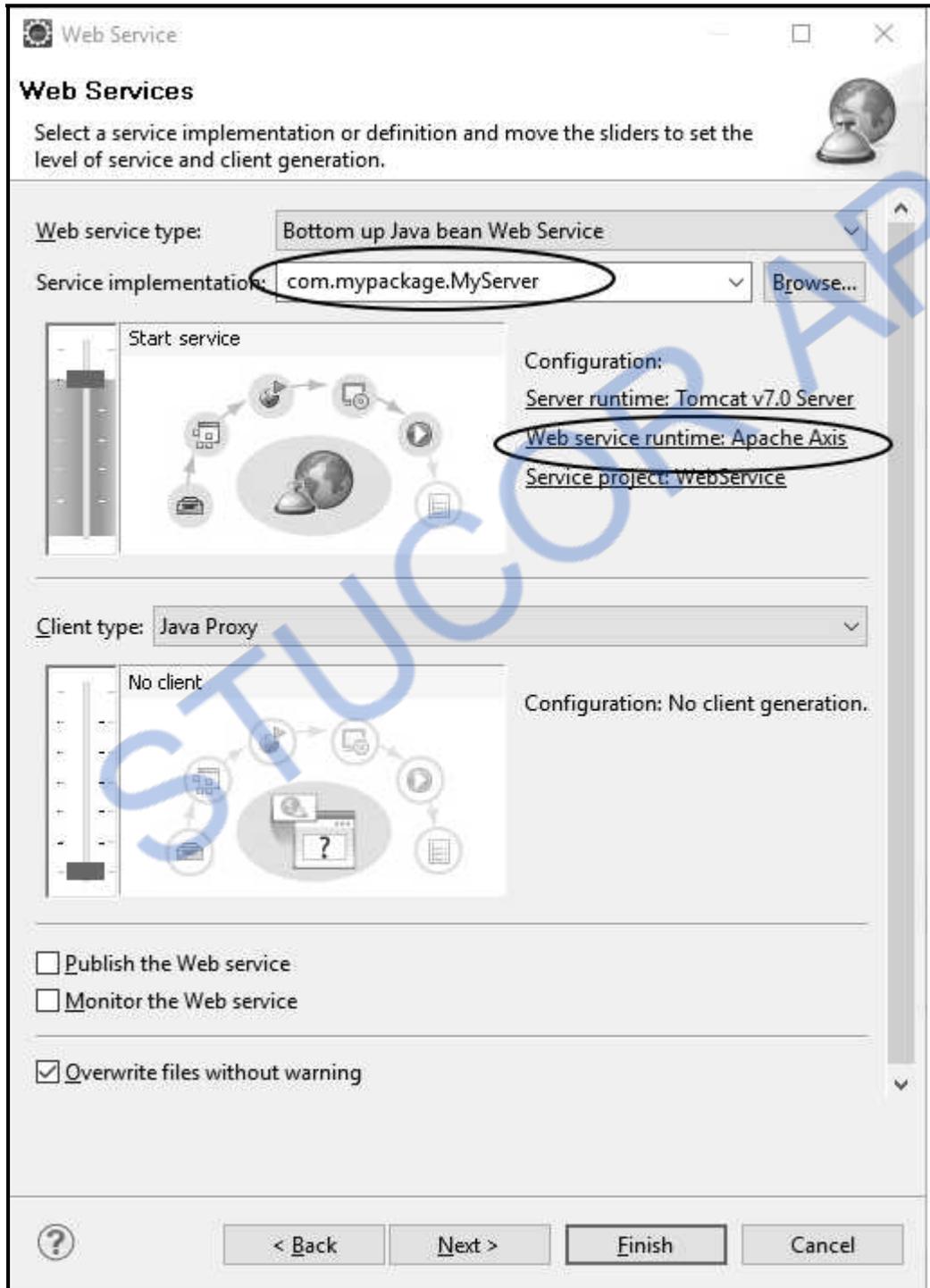
- Following screenshot illustrates these steps.



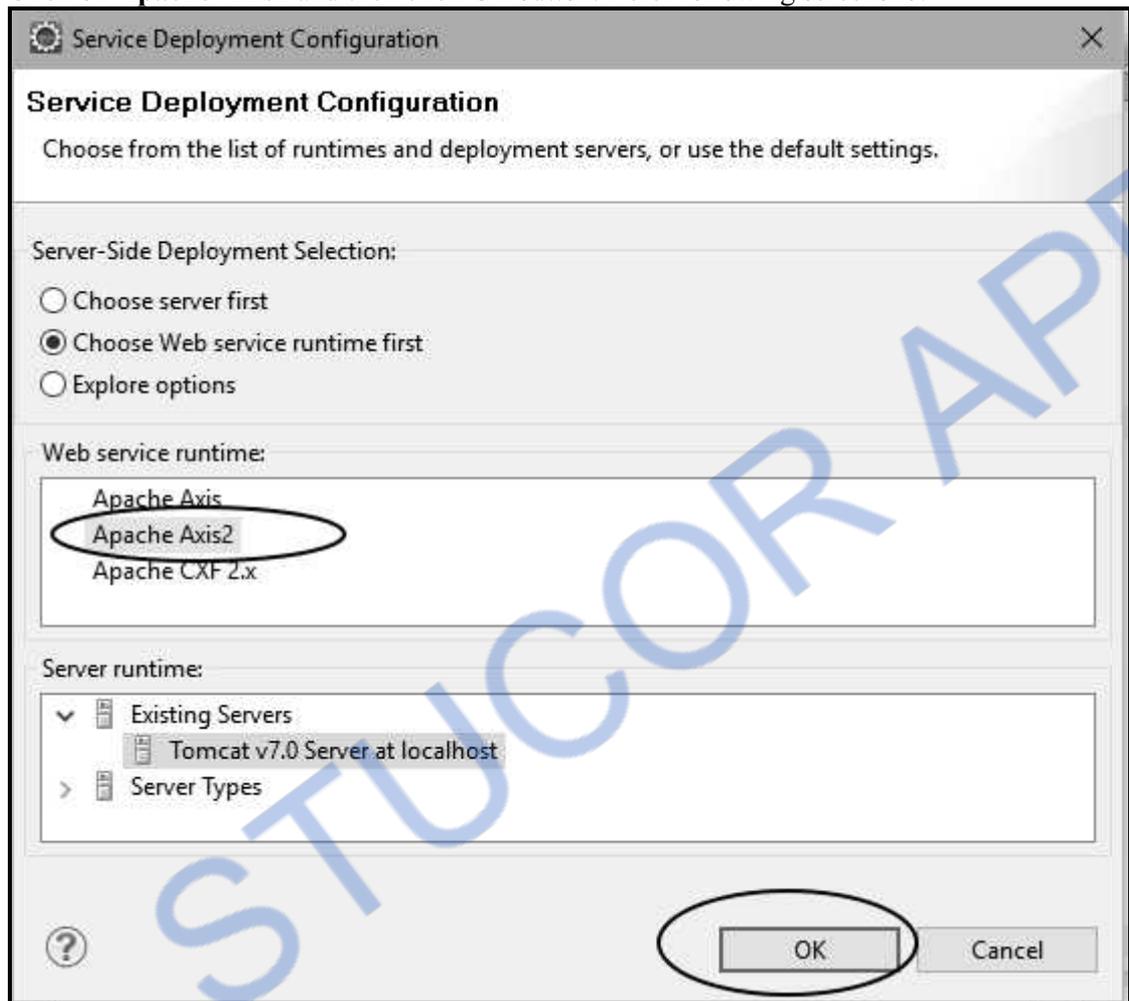
Step 4 : Now right click on **src** folder in Project Explorer under your **WebService**. Click **New->other->Web Services->Web Service**. Then click **Next** button



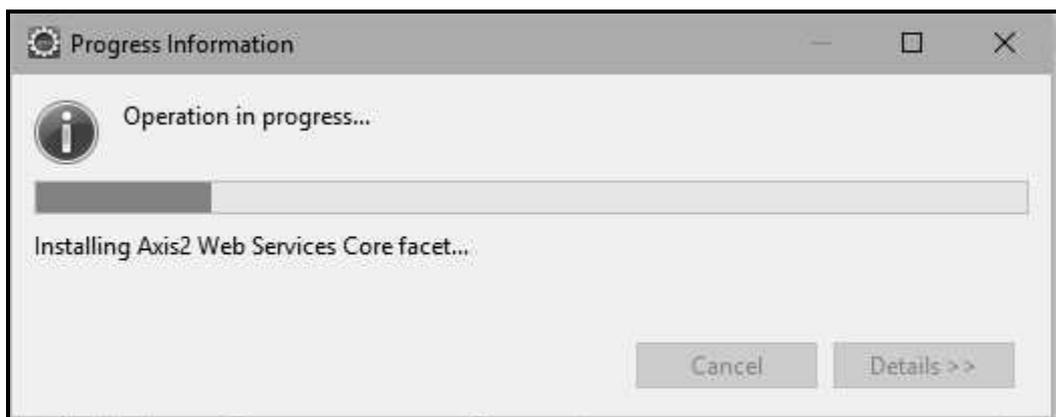
The **Web Services** window will appear. Write the name of service implementation and click on **Web service runtime: Apache Axis**. Following Screenshot shows this-



Click on **Apache Axis2** and then click **Ok** button. Refer following screenshot



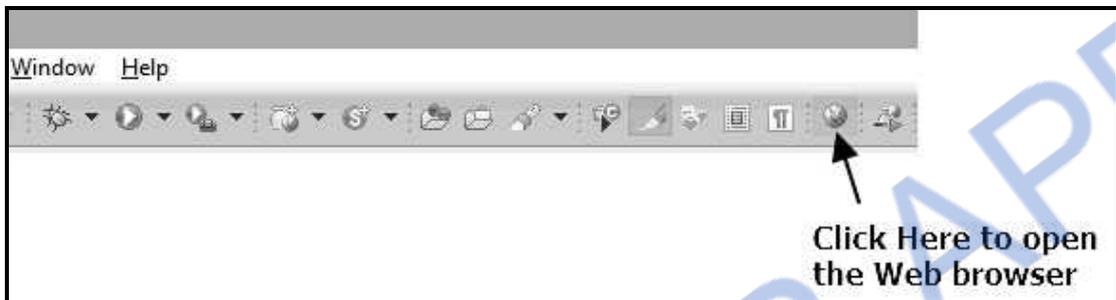
Then click on Finish button. You will get the progress window as



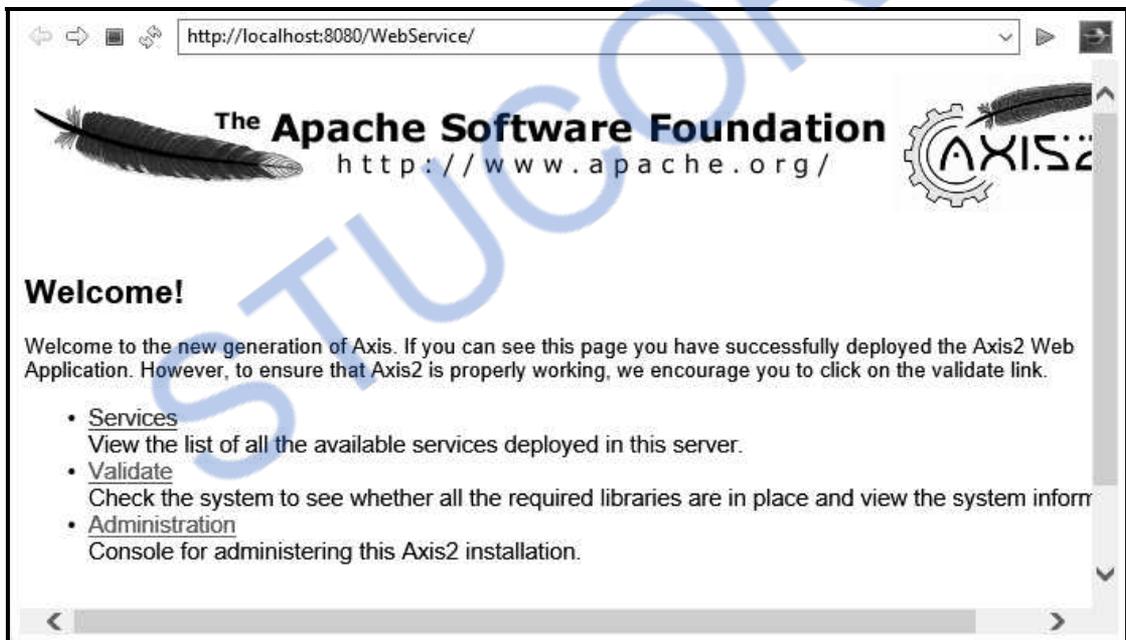
This will generate the WSDL file for your application.

Step 5 : To view the generated WSDL file we need to open up the Web Browser in Eclipse.
For that

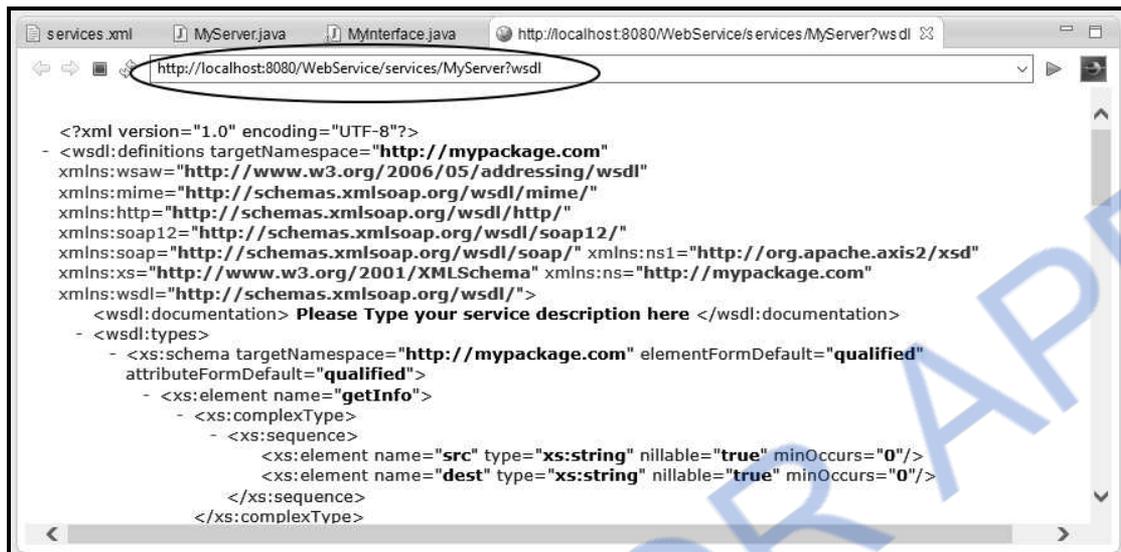
1. Open Web Browser present on the Toolbar



2. Type the URL as follows -



3. Now click on **Services**. Your Web Service name **MyServer** can be located on the page.
4. Just click on **MyServer** link another web page will be opened up in the browser. This is actually your WSDL file. Copy the full URL for WSDL file marked in screenshot given on next page. Save this URL in some Notepad file. Note that : We need this URL during the creation of Client.



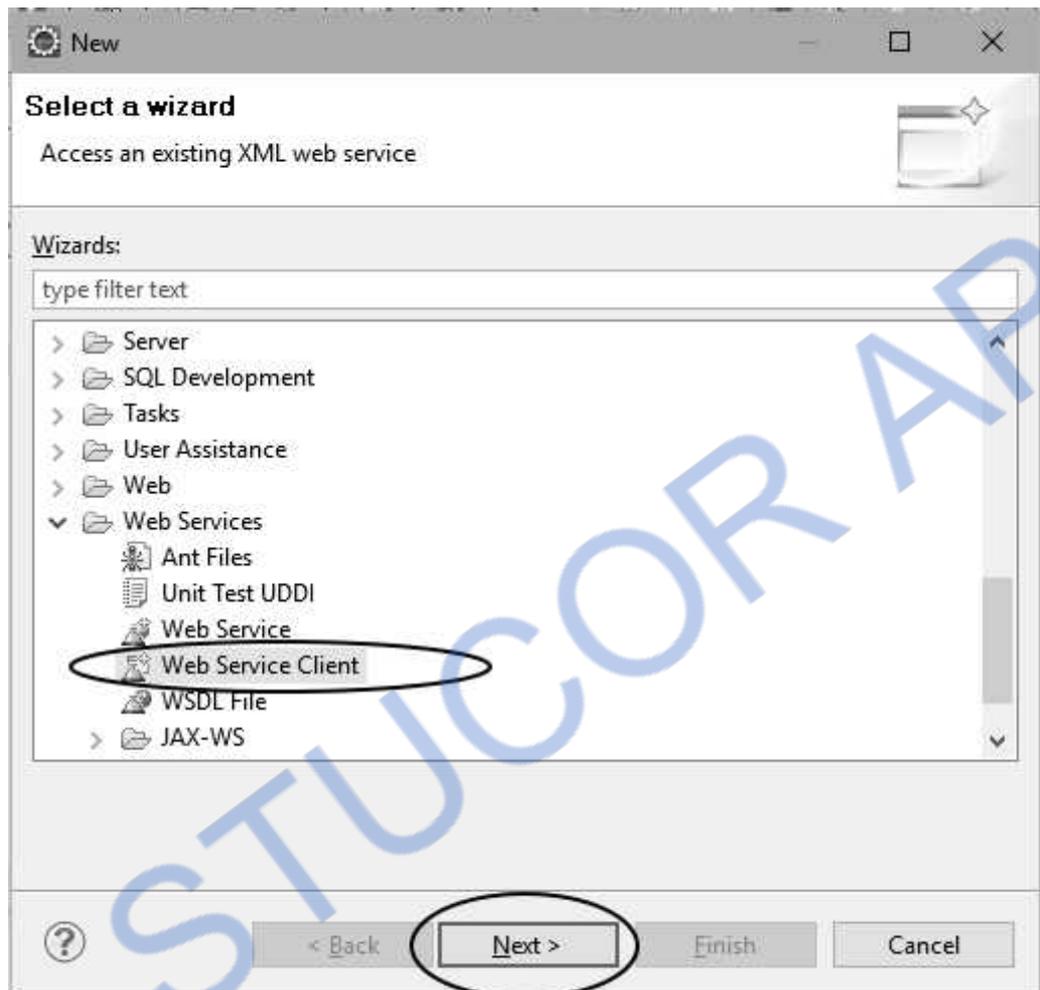
```
<?xml version="1.0" encoding="UTF-8"?>
- <wsdl:definitions targetNamespace="http://mypackage.com"
  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:ns1="http://org.apache.axis2/xsd"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ns="http://mypackage.com"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:documentation> Please Type your service description here </wsdl:documentation>
  - <wsdl:types>
    - <xs:schema targetNamespace="http://mypackage.com" elementFormDefault="qualified"
      attributeFormDefault="qualified">
      - <xs:element name="getInfo">
        - <xs:complexType>
          - <xs:sequence>
            <xs:element name="src" type="xs:string" nillable="true" minOccurs="0"/>
            <xs:element name="dest" type="xs:string" nillable="true" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
</wsdl:definitions>
```

Thus we have created the Web Service.

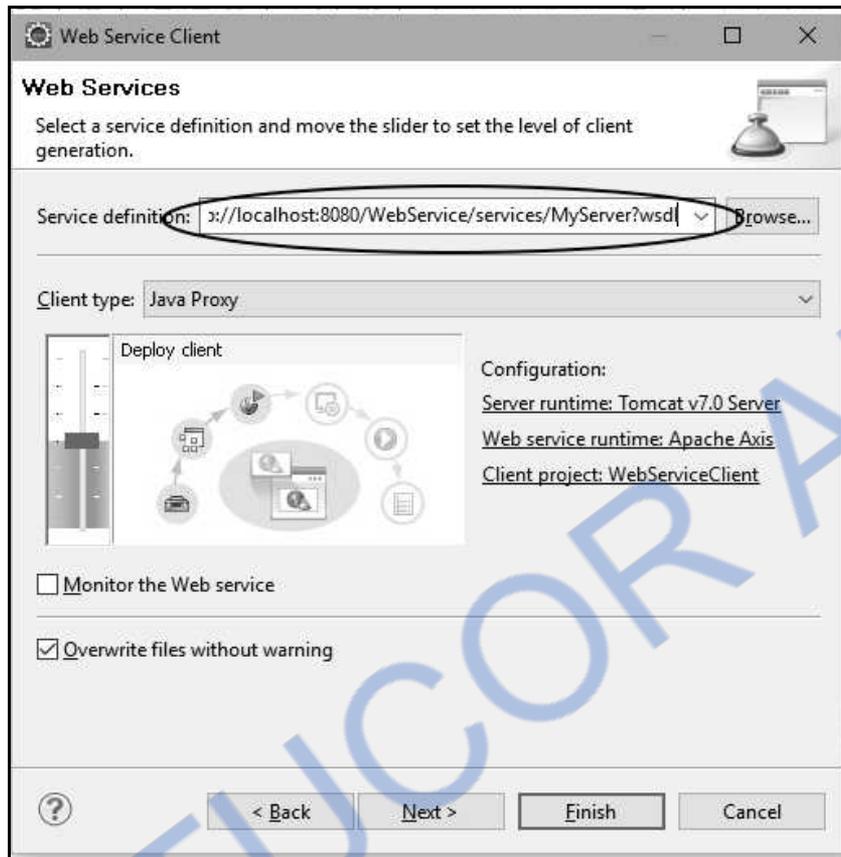
Stage III: Creating Client for Web Service

Step 6 : Create a simple Java servlet for which invokes the web service. For that purpose go to **File->New->Dynamic Web Project**. Give suitable name for your application. Here we are creating a client for the Web service created in above steps. Hence I have given the name **WebServiceClient**. Then click **Finish** button

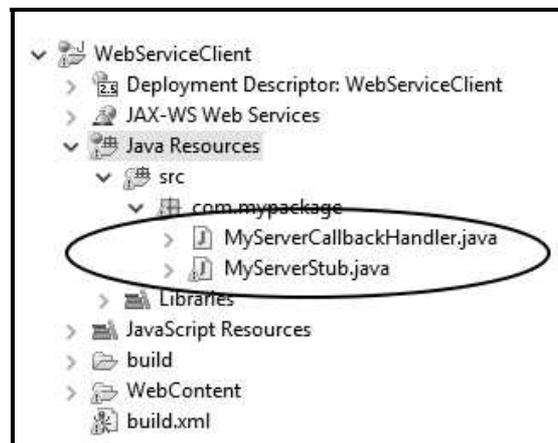
Step 7 : Now under **WebServiceClient**, right click on **src** folder. Click on **New->Other->Web Services->Web Service Client**



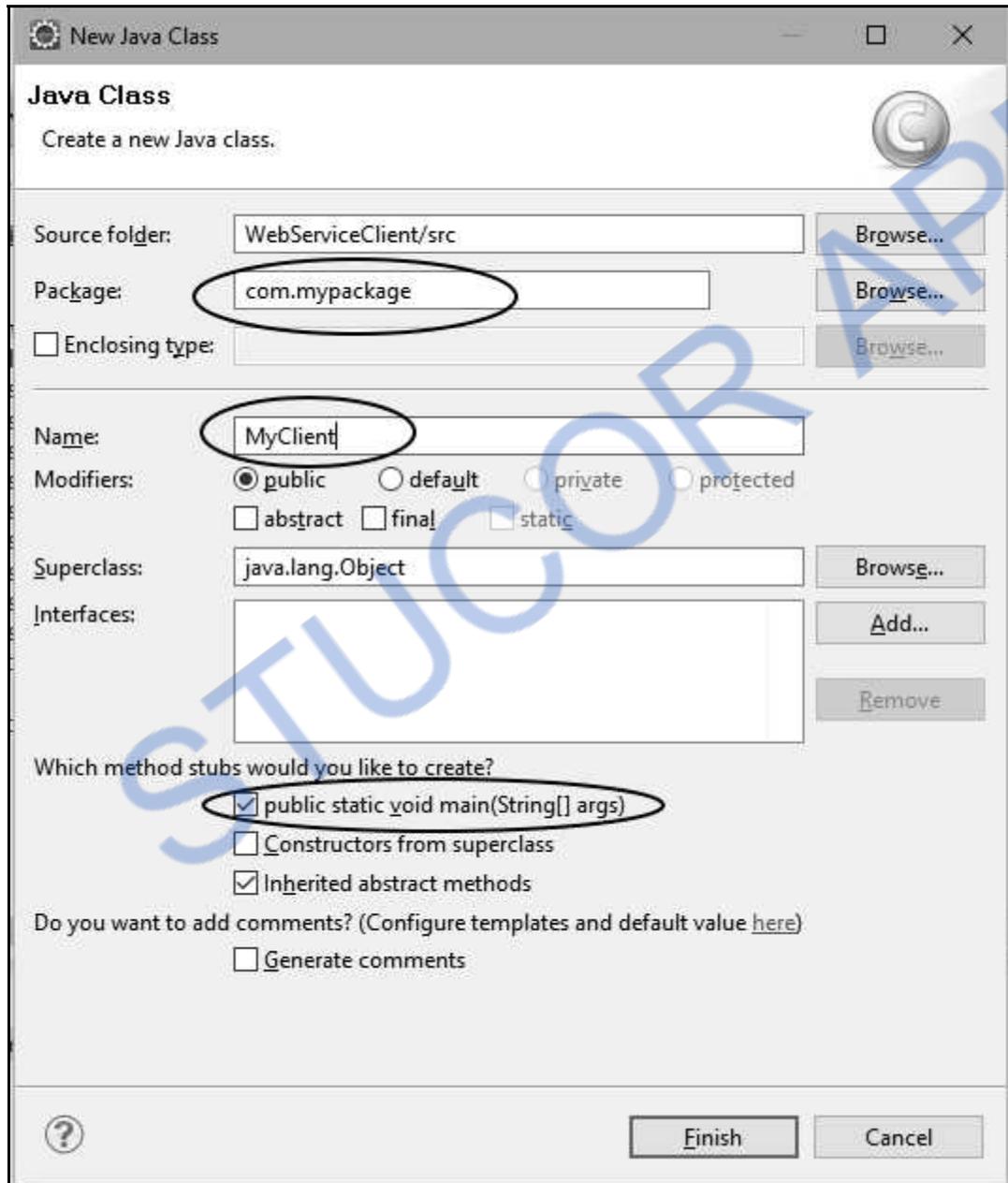
Click the **Next** button. Now on **Web Services** window. Copy the URL of WSDL which you have copied in Step 5(4).



Then Select **Web Service runtime:Apache Axis**. Set it to **Apache Axis2** click **OK** button. Then Click on **Finish** button. You will get Progress Information and then two files will get generated namely **MyServerCallbackHandler.java** and another **MyServerStub.java**. Following is a snapshot of **Project Explorer** Window -



Step 8 : Now right click on **src** folder of **WebServiceClient**. Click on **New->Class**. The **New Java Class** window will appear. Set appropriate package name, class name and Check for void main method. Here is the screenshot



Click on **Finish** button. You can now write the Java Client code for web service. The code in as follows

MyClient.java

```

package com.mypackage;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.mypackage.MyServerStub.GetInfo;
import com.mypackage.MyServerStub.GetInfoResponse;

public class MyClient extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
        throws IOException,ServletException
    {
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        String s=req.getParameter("Src_name");
        String d=req.getParameter("Dest_name");

        MyServerStub stub=new MyServerStub();
        GetInfo p=new GetInfo();
        p.setSrc(s);
        p.setDest(d);
        GetInfoResponse response = stub.getInfo(p);
        String result=response.get_return();
        out.println("<html>");
        out.println("<body bgcolor='khaki' >");
        out.println ( result );
        out.println("</body>");
        out.println("</html>");

    }
}

```

Save the above code.

Step 9 : Update the web.xml file. Here is it -

web.xml

(Location: WebServiceClient->Web Content->WEB-INF->web.xml]

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

```

```

<display-name>WebServiceClient</display-name>
<welcome-file-list>
  ...
  ...
  ...

<servlet>
  <servlet-name>MyClient</servlet-name>
  <servlet-class>com.mypackage.MyClient</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>MyClient</servlet-name>
  <url-pattern>/servlet/com.mypackage.MyClient</url-pattern>
</servlet-mapping>

  ...
  ...
  ...

<servlet-name>AxisAdminServlet</servlet-name>
  <servlet-class>org.apache.axis2.transport.http.AxisAdminServlet</servlet-class>
  <load-on-startup>100</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>AxisAdminServlet</servlet-name>
  <url-pattern>/axis2-admin/*</url-pattern>
</servlet-mapping>
</web-app>

```

Step 10 : Now create a simple HTML file which will invoke the client servlet program(Created in step 8) . This file can be at any desired location. For instance I have stored this file in my D:\ drive.

Input.html

```

<!DOCTYPE html>
<html>
<head>
<title>AIRLINE RESERVATION SERVICE</title>
</head>
<body bgcolor ="khaki">
<h4>Travel Agent should fill up following information to get information about the Airline</h4>
  <form name="form1" method="GET" action
"http://localhost:8080/WebServiceClient/servlet/com.mypackage.MyClient">
  <table>
  <tr>

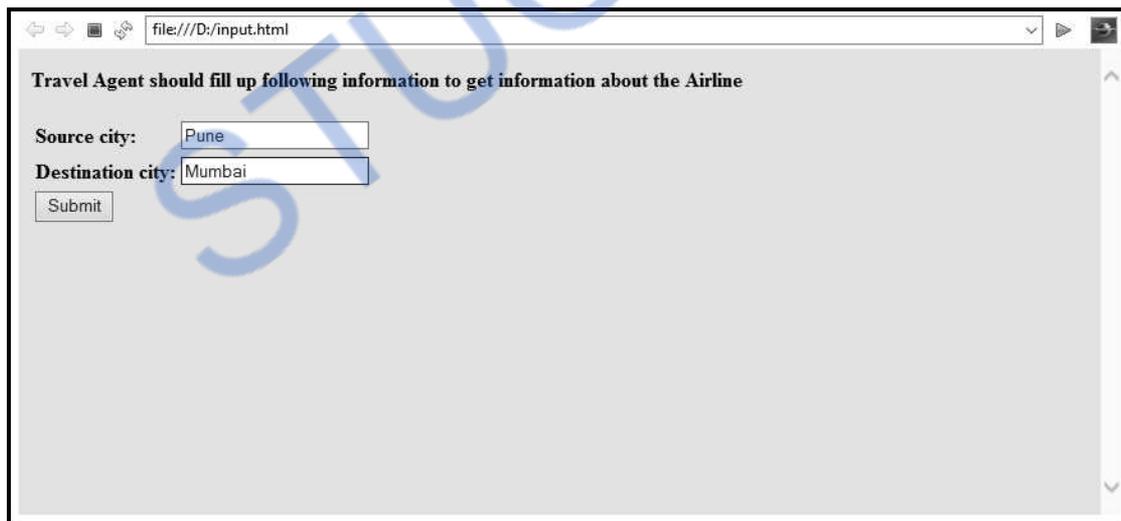
```

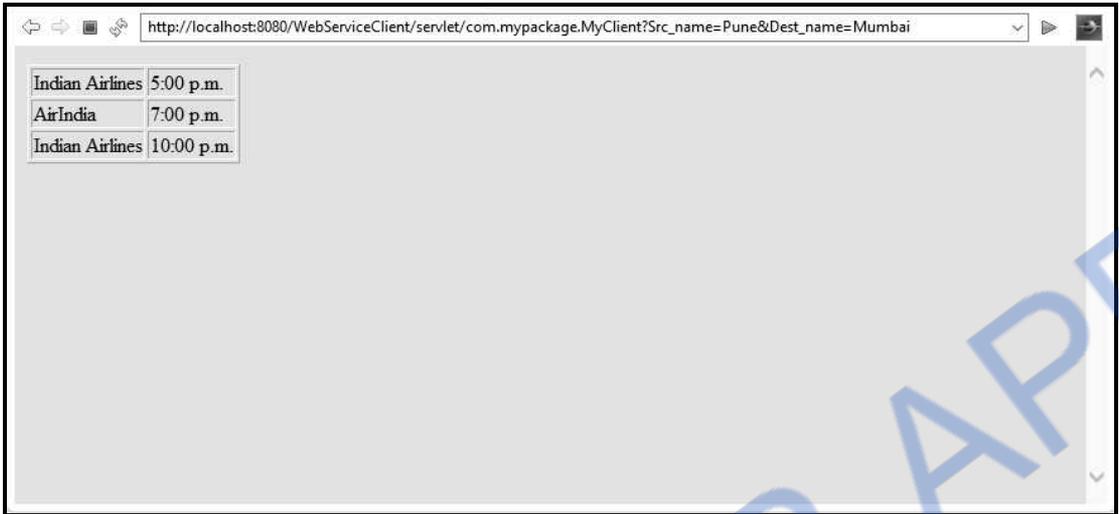
```
<td><b>Source city: </b></td>
<td><input type="text" name="Src_name" size="20" value=""/></td>
</tr>
<tr>
<td><b>Destination city: </b></td>
<td><input type="text" name="Dest_name" size="20" value=""/></td>
</tr>
<tr>
<td><input type="submit" value="Submit"/></td>
</tr>
</table>
</form>
</body>
</html>
```

Save the above code.

Step 11 : Open the web browser and invoke the above created HTML file. This HTML file is for Travel agent. He/she will use it for searching the desired airline. In this application our criteria for searching the airline is – **Source city** and **destination city**. This criterion may change from developer's point of view.

Output





STUCOR APP