NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE MATERIALS

CS 368 WEB TECHNOLOGIES

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- Established in: 2002
- Course offered : B.Tech in Computer Science and Engineering

  M.Tech in Computer Science and Engineering

  M.Tech in Cyber Security
- Approved by AICTE New Delhi and Accredited by NAAC
- Affiliated to the University of A P J Abdul Kalam Technological University.

DEPARTMENT VISION

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

DEPARTMENT MISSION

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.
PEO2: Graduates will be able to Analyse, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.
PEO3: Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.
PEO4: Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Teamworkand leadership qualities.

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

COURSE OUTCOMES

CO1	To understand different component in web technology and learn about CGI and CMS.
CO2	To develop interactive web pages using HTML/XHTML
CO3	To design a web page using cascading style sheets.
CO4	To develop user interactive websites using Javascript and JQuery
CO5	To acquire knowledge about different information interchange formats like XML and JSON
CO6	To develop web applications using PHP.

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

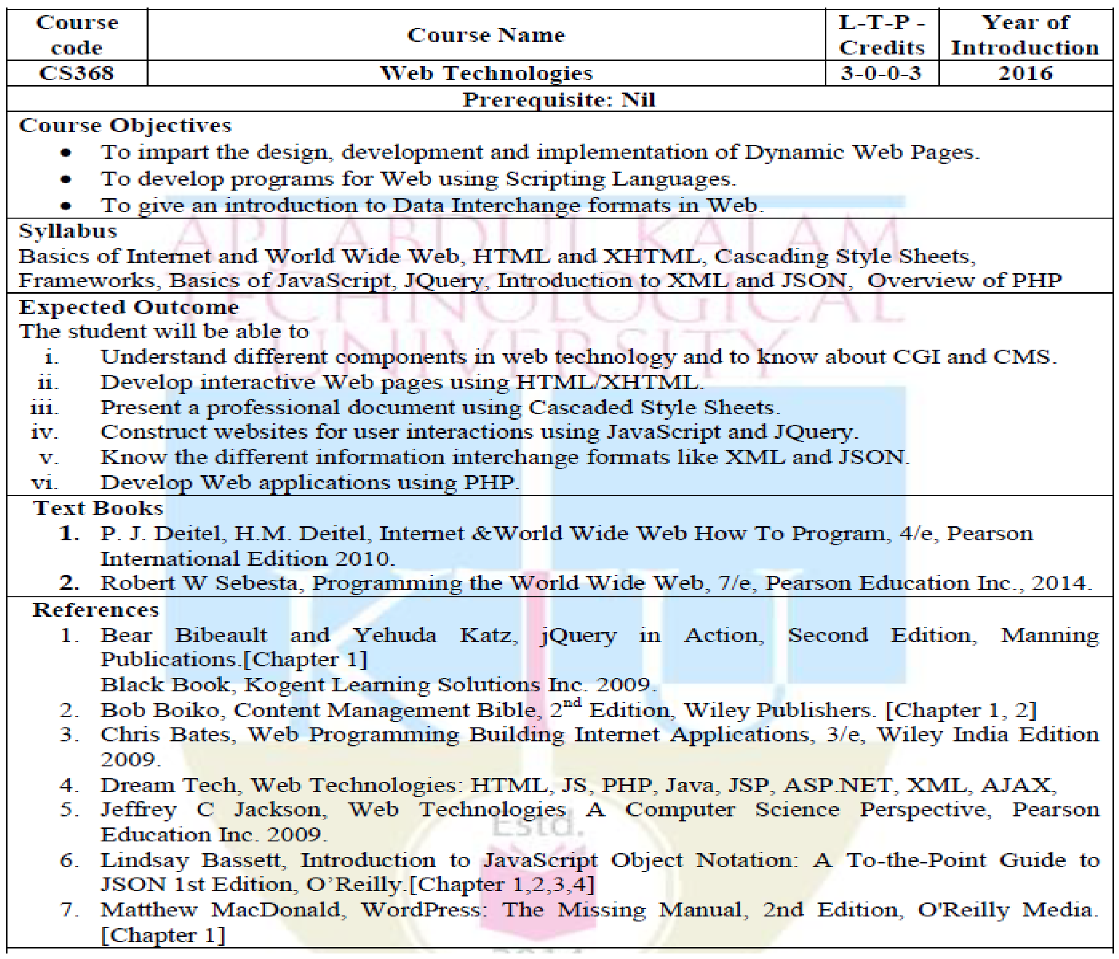
	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**
CO1	**3**	**3**	**-**	**-**								**2**
CO2	**3**		**3**		**3**							
CO3	**3**		**3**		**3**							
CO4	**3**		**3**		**3**							
CO5	**3**	**3**										
CO6	**3**		**3**		**3**							

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

MAPPING OF COURSE OUTCOMES WITH PROGRAM SPECIFIC OUTCOMES

	PSO1	PSO2	PSO3
CO1	-		3
CO2	3	3	3
CO3	-	2	3
CO4	3	3	3
CO5		2	3
CO6	3	3	3

SYLLABUS



Course code	Course Name	L-T-P - Credits	Year of Introduction
CS368	Web Technologies	3-0-0-3	2016

Prerequisite: Nil

Course Objectives

- To impart the design, development and implementation of Dynamic Web Pages.
- To develop programs for Web using Scripting Languages.
- To give an introduction to Data Interchange formats in Web.

Syllabus

Basics of Internet and World Wide Web, HTML and XHTML, Cascading Style Sheets, Frameworks, Basics of JavaScript, JQuery, Introduction to XML and JSON, Overview of PHP

Expected Outcome

The student will be able to

i. Understand different components in web technology and to know about CGI and CMS.
ii. Develop interactive Web pages using HTML/XHTML.
iii. Present a professional document using Cascaded Style Sheets.
iv. Construct websites for user interactions using JavaScript and JQuery.
v. Know the different information interchange formats like XML and JSON.
vi. Develop Web applications using PHP.

Text Books

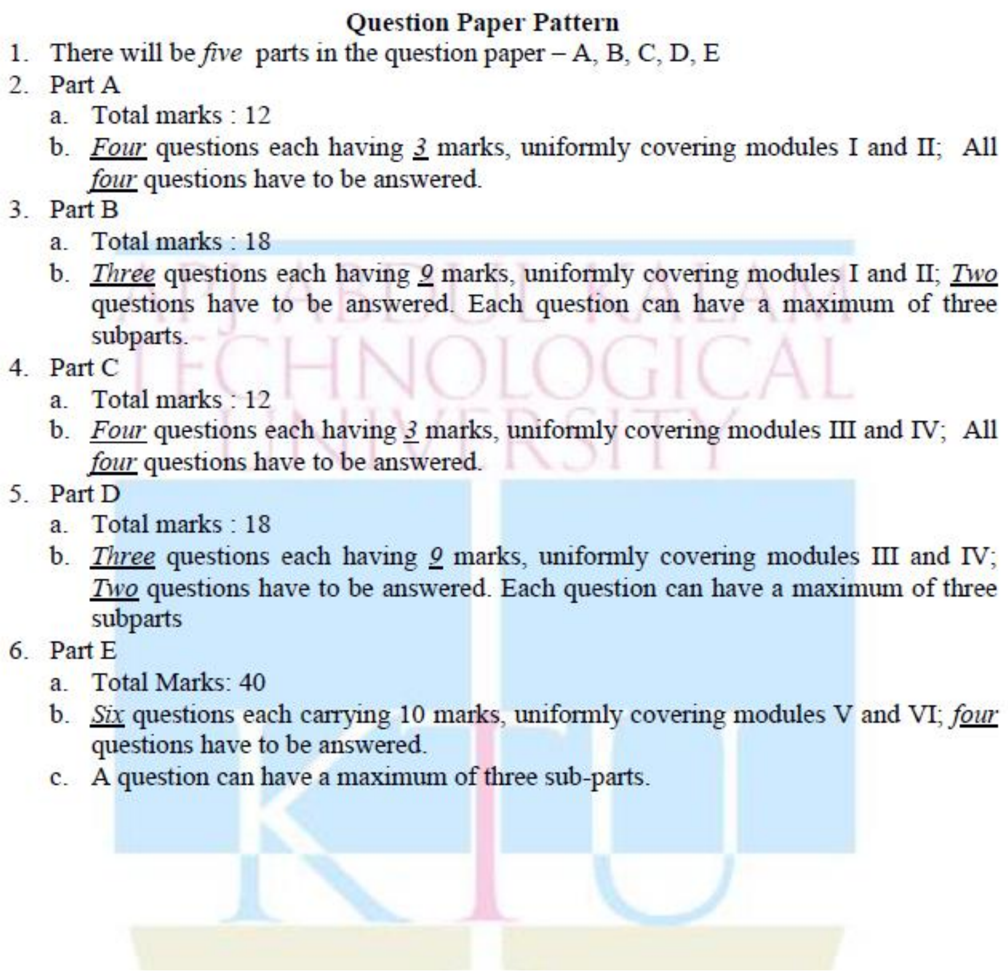
1. P. J. Deitel, H.M. Deitel, Internet &World Wide Web How To Program, 4/e, Pearson International Edition 2010.
2. Robert W Sebesta, Programming the World Wide Web, 7/e, Pearson Education Inc., 2014.

References

1. Bear Bibeault and Yehuda Katz, jQuery in Action, Second Edition, Manning Publications.[Chapter 1]
   Black Book, Kogent Learning Solutions Inc. 2009.
2. Bob Boiko, Content Management Bible, 2nd Edition, Wiley Publishers. [Chapter 1, 2]
3. Chris Bates, Web Programming Building Internet Applications, 3/e, Wiley India Edition 2009.
4. Dream Tech, Web Technologies: HTML, JS, PHP, Java, JSP, ASP.NET, XML, AJAX,
5. Jeffrey C Jackson, Web Technologies A Computer Science Perspective, Pearson Education Inc. 2009.
6. Lindsay Bassett, Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON 1st Edition, O'Reilly.[Chapter 1,2,3,4]
7. Matthew MacDonald, WordPress: The Missing Manual, 2nd Edition, O'Reilly Media. [Chapter 1]

Course Plan

Module	Contents	Hours	End Sem. Exam Marks
I	**Introduction to the Internet:** The World Wide Web, Web Browsers, Web Servers, Uniform Resource Locators, Multipurpose Internet Mail Extensions, The Hypertext Transfer Protocol. Common Gateway Interface(CGI), Content Management System – Basics *Case Study:* Apache Server, WordPress.	**06**	**15%**
II	**Introduction to HTML/XHTML :** Origins and Evolution of HTML and XHTML, Basic Syntax of HTML, Standard HTML Document Structure, Basic Text Markup, Images, Hypertext Links, Lists, Tables, Forms, HTML5, Syntactic Differences between HTML and XHTML.	**07**	**15%**
	FIRST INTERNAL EXAM		
III	**Introduction to Styles sheets and Frameworks** **Cascading Style Sheets:** Levels of Style Sheets - Style Specification Formats, Selector Forms, Property-Value Forms, Font Properties, List Properties, Alignment of Text, Color, The Box Model, Background Images, The span and div Tags. **Frameworks:** Overview and Basics of Responsive CSS Frameworks - Bootstrap.	**06**	**15%**
IV	**Introduction to JavaScript and jQuery** **The Basics of JavaScript:** Overview of JavaScript, Object Orientation and JavaScript, General Syntactic Characteristics- Primitives, Operations, and Expressions, Screen Output and Keyboard Input, Control Statements, Object Creation and Modification,Arrays, Functions. Callback Functions, Java Script HTML DOM. **Introduction to jQuery:** Overview and Basics.	**07**	**15%**
	SECOND INTERNAL EXAMINATION		
V	**Introduction to Data Interchange Formats** **XML:** The Syntax of XML, XML Document Structure, Namespaces, XML Schemas, Displaying Raw XML Documents, Displaying XML Documents with CSS, XSLT Style Sheets, XML Applications. **JSON**(Basics Only): Overview, Syntax, Datatypes, Objects, Schema, Comparison with XML.	**08**	**20%**
VI	**Introduction to PHP**: Origins and Uses of PHP, Overview of PHP - General Syntactic Characteristics - Primitives, Operations, and Expressions - Control Statements, Arrays, Functions, Pattern Matching, Form Handling, Cookies, Session Tracking.	**08**	**20%**
	END SEMESTER EXAM		



Question Paper Pattern

1. There will be *five* parts in the question paper – A, B, C, D, E
2. Part A
   a. Total marks : 12
   b. *Four* questions each having *3* marks, uniformly covering modules I and II; All *four* questions have to be answered.
3. Part B
   a. Total marks : 18
   b. *Three* questions each having *9* marks, uniformly covering modules I and II; *Two* questions have to be answered. Each question can have a maximum of three subparts.
4. Part C
   a. Total marks : 12
   b. *Four* questions each having *3* marks, uniformly covering modules III and IV; All *four* questions have to be answered.
5. Part D
   a. Total marks : 18
   b. *Three* questions each having *9* marks, uniformly covering modules III and IV; *Two* questions have to be answered. Each question can have a maximum of three subparts
6. Part E
   a. Total Marks: 40
   b. *Six* questions each carrying 10 marks, uniformly covering modules V and VI; *four* questions have to be answered.
   c. A question can have a maximum of three sub-parts.

QUESTION BANK

NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to KTU University, Kerala)

CS368 Web Technologies

Module – I Question Bank

1. Write a note on Mime & specification
2. Describe about Uniform resource locator
3. Explain about CMS concept?
4. Point out advantages of CMS
5. Write a note on features of CMS
6. Elaborate about CGI method
7. Illustrate with diagram, the concept of sending request and reply in CGI
8. Define Web server and write its types
9. Elaborate about HTTP Protocol
10. Describe about web browser with an example
11. Briefly analyze apache web server

CS368 – Web Technology

Question Bank

Module – II

1. **Describe the Evolution of HTML?**
2. **Explain the basic structure of HTML?**
3. **Explain font, pre, blockquote, br tags?**
4. **Explain about how to add an image in HTML?**
5. **What is a list? Explain about different types of list?**
6. **Explain the different tags used in table creation using HTML?**
7. **Explain about how to create a form in HTML and tags used?**
8. **Describe HTML5 in detail?**
9. **What are the differences between HTML and XHTML?**
10. **What is an ordered List? Name the tags used in it.**

CS 368 - Web Technologies

Semester: 6

Question Bank
Module – III

1. Define the levels of style sheet used in CSS.

2. How comments are added in CSS?

3. Describe the style specification format used in CSS.

4. Write a note on selector forms used in style sheets.

5. Explain Generic class selector with example.

6. Write a note on the property – value forms.

7. Describe the font properties used in CSS.

8. What is the different list properties used in style sheets?

9. How alignment of text can be done in CSS using properties?

10. Define the Colour properties used in style sheets.

11. Define tiling. How tiling is controlled in CSS?

12. Write a note on the box model in style sheets.

13. Define padding in CSS.

14. Describe margin and padding properties.

15. What is the use of span and div tags in CSS?

16. Write a note on responsive CSS frameworks.

17. Describe the bootstrap framework.

NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to KTU University, Kerala)
CS368 Web Technologies

Module – IV Question Bank

1. Write a note on selection statements used in JavaScript
2. Explain about primitives used in JavaScript.
3. Illustrate with example & explain screen output and keyboard input
4. Elaborate about Array concept in JavaScript
5. Discuss about general syntactic characteristics in JavaScript
6. Explain about object creation and modification in JavaScript
7. Write a JavaScript program to find the factorial of a number
8. What are the different methods by which we can convert a string to a number in JavaScript?
9. Briefly describe operators used in JavaScript
10. Explain callback functions in JavaScript
11. Differentiate between Java and JavaScript
12. Discuss about jQuery in detail

NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to KTU University, Kerala)
CS368 Web Technologies
Module – V Question Bank

1. List & Explain various data types in JSON
2. Design an XML document containing details of three students (Attributes can be selected by your own choice).
3. Differentiate between JSON and XML
4. Create a DTD for a catalogue of cars, where each car element has the child elements make, model, year, color, engine, doors. The engine element has the child elements no_of_cylinders and fuel_type.
5. Elaborate about XSLT style sheets
6. Write notes on entities in XML. State the format for declaring external entities (text and binary) in a DTD.
7. List & Explain various XML applications
8. Design an XML document containing details of three different cars(Attributes can be selected by your own choice)
9. Explain XML document structure
10. Analyze the different requirements for XML Schema
11. Differentiate between Simple and complex data type
12. Discuss about XML Namespace

NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to KTU University, Kerala)
CS368 Web Technologies

Module – VI Question Bank

1. Discuss about cookies and its write its significance
2. Discuss about Associative Arrays
3. Illustrate with example and explain functions in PHP
4. Write a PHP Script to display factorial of a number
5. Illustrate with example, the concept of arrays in PHP
6. Write a PHP Script to display a number is odd or even
7. Distinguish between bound and unbound variables. How can a variable be tested to check whether it is bound?
8. Write a note on pattern matching
9. Discuss about primitives supported by PHP
10. What are the contexts in which session tracking can be helpful?
11. Write a PHP Script to display Fibonacci of a number
12. Elaborate about Selection statement and loop statements in PHP

APPENDIX 1	
CONTENT BEYOND THE SYLLABUS	
S:NO;	**TOPIC**
1	Java Server Pages

MODULE NOTES

.

.

.

.

.

.

Web Technology

Module –I

Introduction to the Internet

The Internet is a huge collection of computers connected in a communications network.

World Wide Web

In 1989, a small group of people led by Tim Berners-Lee at CERN, proposed a new protocol for the Internet, as well as a system of document access to use it. The intent of this new system is to use the Internet to exchange documents.

The proposed new system was designed to allow a user anywhere on the Internet to search for and retrieve documents from databases on any number of different document-serving computers connected to the Internet. The system used hypertext, which is text with embedded links to text in other document.

Web Browsers

In the two communication entities one will be a server and other will be a client. The client initiates the communication, which is often a request for information stored on the server, which then sends that information back to the client. Documents provided by servers on the Web are requested by browsers, which are programs running on client machines. They are called browsers because they allow the user to browse the resources available on servers.

A browser is a client on the Web because it initiates the communication with a server, which waits for a request from the client before doing anything. In the simplest case, a browser requests a static document from a server. The server locates the document among its servable documents and sends it to the browser, which displays it for the user. The Web supports a variety of protocols; the most common one is the Hypertext Transfer Protocol (HTTP). HTTP provides a standard form of communication between browsers and Web servers.

Web Servers

Web servers are programs that provide documents to requesting browsers. Servers are slave programs: They act only when requests are made to them by browsers running on other computers on the Internet. The most commonly used Web servers are Apache, which has been implemented for a variety of computer platforms, and Microsoft's Internet Information Server (IIS), which runs under Windows operating systems. Web browsers initiate network communications with servers by sending them URL. A URL can specify one of two different things:

a) The address of a data file stored on the server that is to be sent to the client.

b) A program stored on the server that the client wants executed, with the output of the program returned to the client.

The primary task of a Web server is to monitor a communications port on its host machine, accept HTTP commands through that port, and perform the operations specified by the commands. All HTTP commands include a URL, which includes the specification of a host server machine.

The file structure of a Web server has two separate directories: The root of one of these is called the **document root**. The file hierarchy that grows from the document root stores the Web documents to which the server has direct access and normally serves to clients. The root of the other directory is called the **server root**. This directory, along with its descendant directories, stores the server and its support software. Many servers allow part of the servable document collection to be stored outside the directory at the document root. The secondary areas from which documents can be served are called **virtual document trees**.

Types of servers:

Contemporary servers: Large and complex systems that provide a wide variety of client services

Virtual hosts: Many servers can support more than one site on a computer, potentially reducing the cost of each site and making their maintenance more convenient.

Proxy Servers: Some servers can serve documents that are in the document root of other machines on the Web.

Uniform Resource Locators (URL)

Uniform (or universal) resource locators (URL) are used to identify documents (resources) on the Internet. Different kinds of resources in web are identified by different URL.

URL Formats

All URL have the same general format: **scheme: object-address**.

The scheme is often a communications protocol. Common schemes include HTTP, FTP, gopher, telnet, file, mail to, and news etc. HTTP protocol is used for supporting the Web. This protocol is used to request and send extensible Hypertext Markup Language (XHTML) documents. In the case of HTTP, the form of the object address of a URL is as follows:

// fully-qualified-domain-name / path-to-document

The file protocol means that the document resides on the machine running the browser. This approach is useful for testing documents to be made available on the Web without making them visible to any other browser.

When file is the protocol, the fully qualified domain name is omitted, making the form of such URL as follows:

file:// path-to-document.

The default port number of Web server processes is 80.

URL Paths:

The path to the document for the HTTP protocol is similar to a path to a file or directory in the file system of an operating system.. For UNIX servers, the path is specified with forward slashes; for Windows servers, it is specified with backward slashes. A path that includes all directories along the way is called a **complete path.** The path to the document is relative to some base path that is specified in the configuration files of the server. Such paths are called **partial paths.**

Multipurpose Internet Mail Extensions (MIME)

A browser needs some way of determining the format of a document it receives from a Web server. The forms of these documents are specified with Multipurpose Internet Mail Extensions (MIME).

Type Specifications:

MIME was developed to specify the format of different kinds of documents to be sent via Internet mail. These documents could contain various kinds of text, video data, or sound data. A Web server attaches a MIME format specification to the beginning of the document that it is about to provide to a browser. When the browser receives the document from a Web server, it uses the included MIME format specification to determine what to do with the document.MIME specifications have the following form:

type/subtype

The most common MIME types are text, image, and video. The most common text subtypes are plain and HTML. Some common image subtypes are gif and jpeg. Some common video subtypes are mpeg and quick time. Servers determine the type of a document by using the filename's extension.

Hypertext Transfer Protocol (HTTP)

All Web communications transactions use the same protocol: the Hypertext Transfer Protocol (HTTP). HTTP consists of two phases: the request and the response. Each HTTP communication is (request or response) between a browser and a Web server. Mainly consists of two parts: a header and a body. The header contains information about the communication; the body contains the data of the communication

The Request Phase

The general form of an HTTP request is as follows:

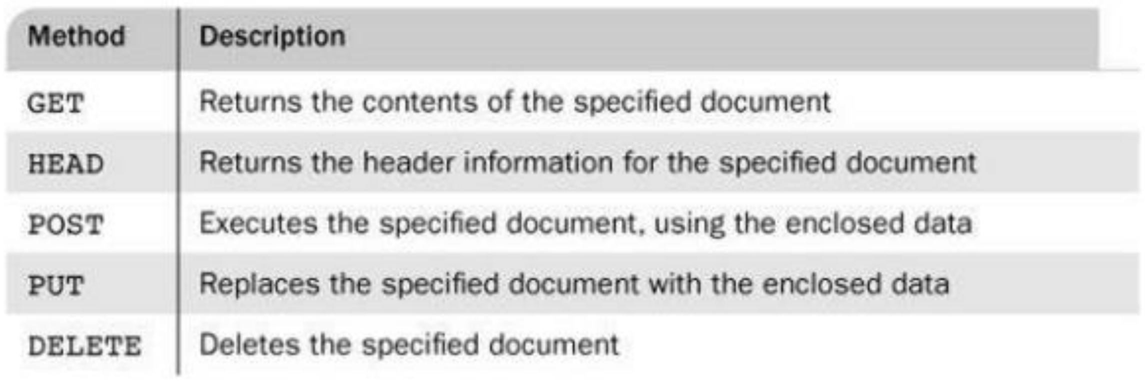
1. HTTP method Domain part of the URL HTTP version

2. Header fields

3. Blank line

4. Message body

Table 1.1 HTTP request methods



Method	Description
GET	Returns the contents of the specified document
HEAD	Returns the header information for the specified document
POST	Executes the specified document, using the enclosed data
PUT	Replaces the specified document with the enclosed data
DELETE	Deletes the specified document

HTTP Request Method

GET and POST are the most frequently used. POST was originally designed for tasks such as posting a news article to a newsgroup. Its most common use now is to send form data from a browser to a server, along with a request to execute a program on the server that will process the data.

Header Fields

The format of a header field is the field name followed by a colon and the value of the field. There are four categories of header fields:

1. General: For general information, such as the date

2. Request: Included in request headers

3. Response: For response headers

4. Entity: Used in both request and response headers

One common request field is the Accept field, which specifies a preference of the browser for the MIME type of the requested document. A wildcard character, the asterisk (*), can be used to specify that part of a MIME type can be anything

The Response Phase

The general form of an HTTP response is as follows:

1. Status line

2. Response header fields

3. Blank line

4. Response body

The status line includes the HTTP version used, a three-digit status code for the response, and a short textual explanation of the status code. The status codes begin with 1, 2, 3, 4, or 5.

Table 1.2 First digits of HTTP status codes

First Digit	Category
1	Informational
2	Success
3	Redirection
4	Client error
5	Server error

After the status line, the server sends a response header, which can contain several lines of information about the response, each in the form of a field. The only essential field of the header is Content-type. The response header must be followed by a blank line, as is the case for request headers For Security HTTPS(HTTP Secure) is used.

Common Gateway Interface (CGI)

CGI is a standard interface by which the web server passes the client's request to a program and receives the response from that program.

The usual steps in HTTP Transaction is:

1. Client opens connection to server

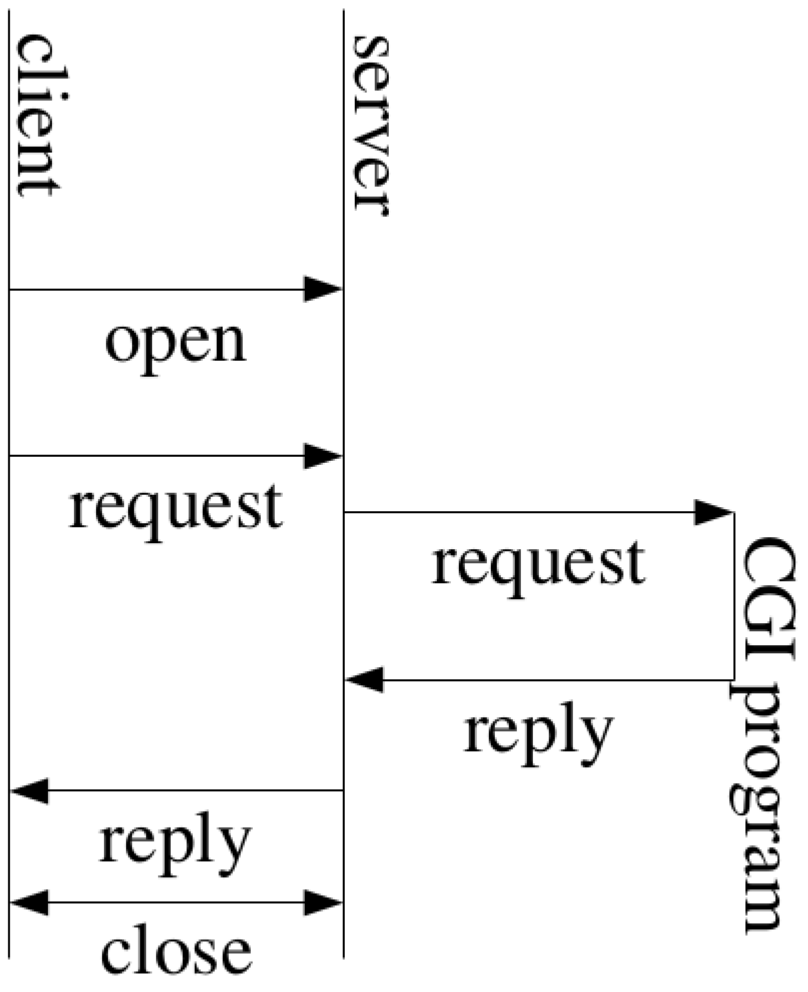
2. Client sends request to server

3. Server responds to request

4. Client and server close connection

CGI is all about what happens between steps 2 and 3.

The CGI Process



The steps are first the Client open a connection to server and then Client sends request to server. Server processes the request given and launches the CGI program. The CGI program is executed and a response is given back to the client. Then client and server close connection

Sending the Request to the Program

The web server sends information to the program using environment variables. This information includes HTTP headers, Server information, Client information and Information about the request

Receiving Data

A CGI program can receive form data in two different ways. If the form is submitted by the GET method then the query is encoded in the QUERY_STRING environment variable. If the form is submitted by the POST method then the data arrives on stdin (standard input). The CONTENT_LENGTH environment indicates how much data will arrive.

Sending the Reply

The CGI program should write it's output to stdout. The output consists of a header, containing, as a minimum the Content-type but possibly also other header fields, a blank line and the content (e.g., text, html, etc.)

Content Management Systems (CMS)

Content management refers to the system and processes whereby information is created, managed, published, and archived. Content is in essence any type or unit of digital information that is used to populate a page - web page or otherwise. It can be text, images, graphics, video, sound etc. Content management is the strategy and technology of storing and indexing information from and about analog or digital media.

Steps in CMS

First step is data collection the information is either created or acquired. It is then converted into a master format such as XML. Then content is divided into small chunks called content components. Content is managed within a repository that consists of database records. Finally the content management system publishes to targeted publications such as websites, printable documents and email, newsletters. Content management systems are sometime referred to as Web Content Management Systems (WCMS).

A CMS consists of two elements:

1) Content Management Application(CMA)

2) Content Delivery Application (CDA)

The CMA element allows the content manager or author to manage the creation, modification and removal of content from websites. CDA element uses the information, compiling it to update the website.

CMS Features

The web-based publishing feature allows individuals to use a template to create or modify web content. The format management feature allows documents to be formatted into HTML or Portable Document Format (PDF) for the website. The revision control feature allows content to be updated to a newer version or restored to a previous version. The CMS also provides features such as indexing, searching and retrieval of data.

Advantages of CMS

1) Fresh, consistent, high quality information
2) Reuse of content
3) Enhanced productivity
4) Decentralized content creation

Apache Web Server

Apache began as the NCSA server, httpd, with some added features. The name Apache because it was a patchy version of the httpd sever. The reasons for common use of Apache:

1) Fast and reliable

2) It is open-source software, which means that it is free and is managed by a large team of volunteers.

3) Best available servers for Unix-based systems.

4) Efficient and Effective.

Apache is capable of providing a long list of services beyond the basic process of serving documents to clients. When Apache begins execution, it reads its configuration information from a file and sets its parameters to operate accordingly. There are three configuration files in an Apache server: httpd.conf, srm.conf, and access.conf. In windows platform IIS is used.

Web Technology

Module –II

Introduction to HTML and XHTML

The HTML is defined with the use of the Standard Generalized Markup Language (SGML), which is an International Standards Organization (ISO). Used as a notation for describing text-formatting languages.

Origin and Evolution of HTML and XHTML

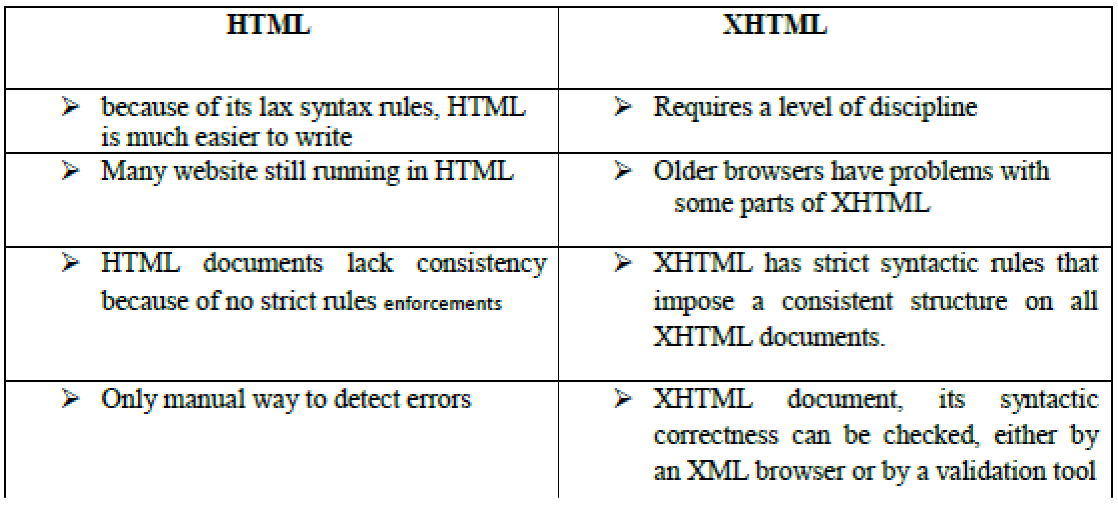
The original version of HTML was designed in conjunction with the structure of the Web and the first browser, at CERN. In late 1994, Tim Berners-Lee, who developed the initial version of HTML, started the World Wide Web Consortium (W3C). One of the primary goals is to develop and distribute standards for Web technologies starting with HTML.

The first HTML standard, HTML 2.0, was released in 1995. It was followed by HTML 3.2 in early 1997. The latest version of HTML, 4.01, was approved by W3C in late 1999. The XHTML 1.0 standard was approved in early 2000. XHTML 1.0 is a redefinition of HTML 4.01 using XML.

XHTML 1.0 is actually three standards: **Strict, Transitional, and Frameset.** The **Strict standard** requires all of rules of XHTML 1.0 be followed. The **Transitional standard** allows deprecated features of XHTML 1.0 to be included. The Frameset standard allows the collection of frame elements and attributes to be included.

HTML versus XHTML

HTML is much easier to write, whereas XHTML requires a level of discipline. Due to the huge number of HTML documents available on the Web, browsers will continue to support HTML, whereas XHTML have some problems with browsers. HTML is less consistent compared to XHTML, whereas XHTML is highly consistent and of high quality. HTML programmers have high degree of freedom. But XHTML programmers have less degree of freedom.



HTML	XHTML
➢ because of its lax syntax rules, HTML is much easier to write	➢ Requires a level of discipline
➢ Many website still running in HTML	➢ Older browsers have problems with some parts of XHTML
➢ HTML documents lack consistency because of no strict rules enforcements	➢ XHTML has strict syntactic rules that impose a consistent structure on all XHTML documents.
➢ Only manual way to detect errors	➢ XHTML document, its syntactic correctness can be checked, either by an XML browser or by a validation tool

Basic Syntax

The fundamental syntactic units of HTML are called tags. In general, tags are used to specify categories of content. The syntax of a tag is the tag's name surrounded by angle brackets (< tagname >). Tag names must be written in all lowercase letters.

Most tags appear in pairs: an opening tag and a closing tag. The name of a closing tag is the name of its corresponding opening tag with a slash (/) attached to the beginning.

Opening tag + Closing tag = Container

Container + Content = Element

Attributes, which are used to specify alternative meanings of a tag, can appear between an opening tag's name and its right angle bracket. Attribute names are written in lowercase letters. Attribute values must be delimited by double quotes.

Commenting in HTML

Comments in programs increase the readability of those programs. They can appear in HTML in the following form:

<! -- Comments to be written -->

Standard XHTML Document Structure

Every XHTML document must begin with an xml declaration element that simply identifies the document as being one based on XML.

<?xml version = “1.0” encoding = “utf-8”?>

This element includes an attribute that specifies the version number which is 1.0. The xml declaration usually includes a second attribute, encoding, which specifies the encoding used for the document.

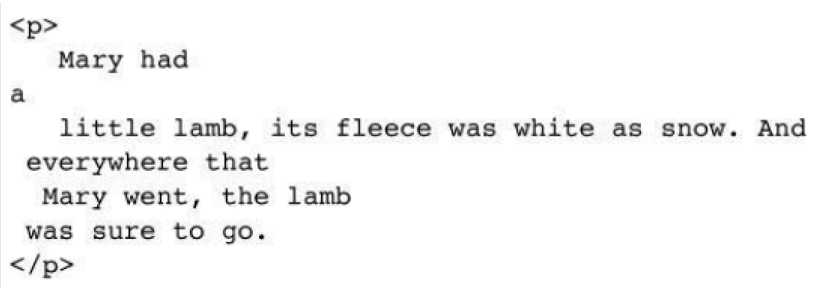
An XHTML document must include the four tags <html>, <head>, <title>, and <body>. The <html> tag identifies the root element of the document. The html element includes an attribute, xmlns that specifies the XHTML namespace, as shown in the following element:

<html xmlns = “http://www.w3.org/1999/xhtml”>

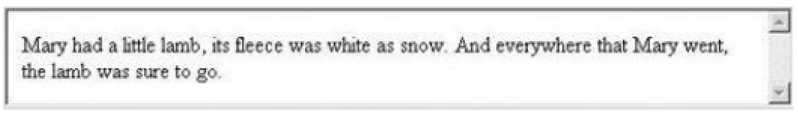
An XHTML document consists of two parts, named the head and the body. The <head> element contains the head part of the document which provides information about the document The body of a document provides the content of the document.

Paragraph tag

<p> tag is used for setting the given text into paragraphs.



```
<p>
   Mary had
a
   little lamb, its fleece was white as snow. And
 everywhere that
  Mary went, the lamb
 was sure to go.
</p>
```



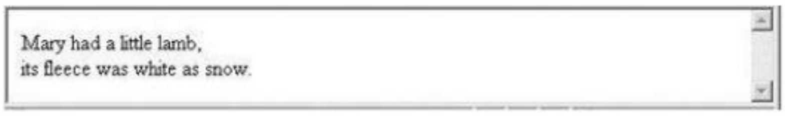
```
Mary had a little lamb, its fleece was white as snow. And everywhere that Mary went,
the lamb was sure to go.
```

Line Breaks

The break tag is specified as <br />. The slash indicates that the tag is both an opening and closing tag. The space before the slash represents the absent content.

```
<p>
Mary had a little lamb, <br />
  its fleece was white as snow.
</p>
```

This markup would be displayed as shown in Figure 2.4.



Mary had a little lamb,
its fleece was white as snow.

<pre> tag

To prevent the browser from eliminating multiple spaces and ignoring embedded line breaks. This can be specified with the pre tag.

Headings

Text is often separated into sections in documents by beginning each section with a heading. In HTML, there are six levels of headings, specified by the tags <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>, where <h1> specifies the highest-level heading. Headings are usually displayed in a boldface font whose default size depends on the number in the heading tag.

Block Quotations

Used when a block of text to be displayed in different style from the normal text. The <blockquote>tag is designed for this situation.

Font Styles

Early Web designers used a collection of tags to set font styles and sizes. For example, <i> specified italics and <b> specified bold.

Content-based style tags

These tags are called content based because they indicate that contents are displayed in different styles. Three of the most commonly used content-based tags are the emphasis tag, the strong tag, and the code tag.

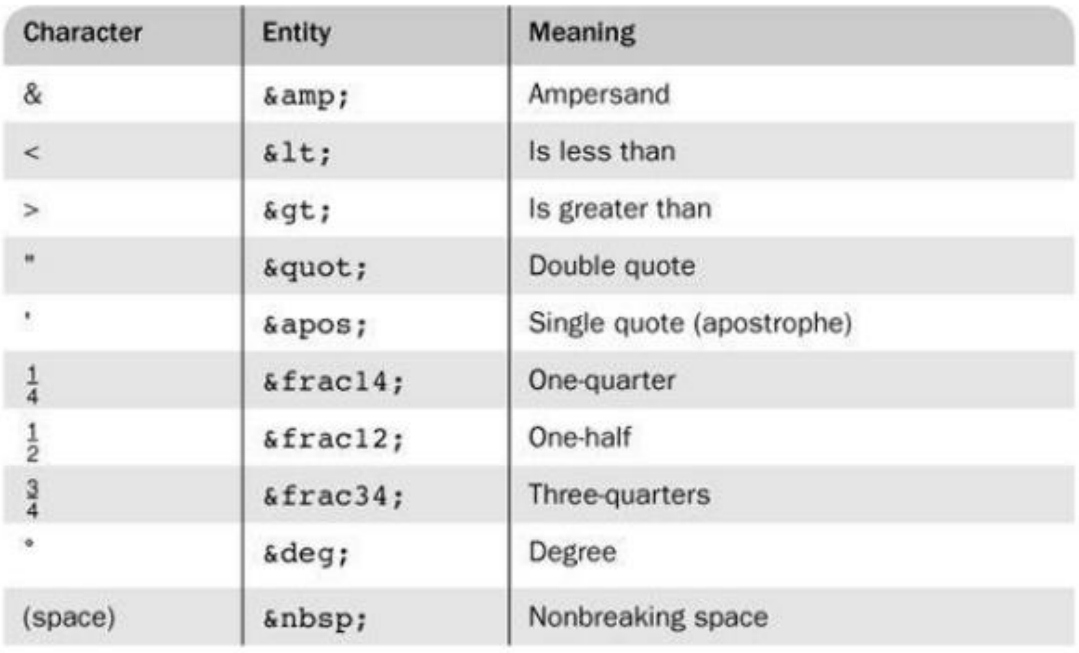
The **emphasis tag, <em>**, specifies that its textual content is special and should be displayed in some way that indicates this distinctiveness. Most browsers use italics for such content.

The **strong tag, <strong>** is like the emphasis tag, but more so. Browsers often set the content of strong elements in bold.

The **code tag, <code>**, is used to specify a monospace font, usually for program code. Subscript and superscript characters can be specified by the <sub> and <sup> tags, respectively.

Character Entities

A collection of special characters are provided. These special characters are defined as entities, which are codes for the characters



Character	Entity	Meaning
&	`&amp;`	Ampersand
<	`&lt;`	Is less than
>	`&gt;`	Is greater than
"	`&quot;`	Double quote
'	`&apos;`	Single quote (apostrophe)
$\frac{1}{4}$	`&frac14;`	One-quarter
$\frac{1}{2}$	`&frac12;`	One-half
$\frac{3}{4}$	`&frac34;`	Three-quarters
°	`&deg;`	Degree
(space)	`&nbsp;`	Nonbreaking space

Horizontal Rules

The parts of a document can be separated from each other, making the document easier to read, by placing horizontal lines between them. Such lines are called horizontal rules, and the block tag that creates them is <hr />.

The <hr /> tag causes a line break (ending the current line) and places a line across the screen. The browser chooses the thickness, length, and horizontal placement of the line. Typically, browsers display lines that are three pixels thick. The slash in the <hr /> tag, indicating that this tag has no content and no closing tag.

Meta Element

The meta element is used to provide additional information about a document. The meta tag has no content; rather, all of the information provided is specified with attributes. The two attributes that are used to provide information are name and content. The user makes up a name as the value of the name attribute and specifies information through the content attribute. Web search engines use the information provided with the meta element to categorize Web documents in their indices.

Table in HTML

Tables are common fixtures in printed documents, books, and, of course, Web documents. Tables provide a highly effective way of presenting many kinds of information. A table is a matrix of cells. The cells in the top row often contain column labels, those in the leftmost column often contain row labels, and most of the rest of the cells contain the data of the table. The content of a cell can be almost any document element, including text, a heading, a horizontal rule, an image, and a nested table.

Basic Table tags

A table is specified as the content of the block tag <table>. There are two kinds of lines in tables: the line around the outside of the whole table is called the **border**; the lines that separate the cells from each other are called **rules**. A displayed table is preceded by a title, given as the content of a **<caption>** tag.

The cells of a table are specified one row at a time. Each row of a table is specified with a row tag, **<tr>.** Within each row, the row label is specified by the table heading tag, **<th>.** Although the <th> tag has heading in its name. Each data cell of a row is specified with a table data tag, **<td>.**

Multiple-level labels can be specified with the rowspan and colspan attributes.

The **colspan** attribute specification in a table header or table data tag tells the browser to make the cell as wide as the specified number of rows below it in the table. The **rowspan** attribute of the table heading and table data tags does for rows what colspan does for columns.

Cell padding and cell spacing attributes are used. For aligning the contents align and valign attributes are used.

List In HTML

XHTML provides simple and effective ways to specify lists in documents.

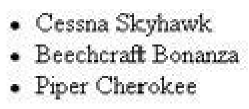
Unordered Lists

The <ul> tag, which is a block tag, creates an unordered list. Each item in a list is specified with an <li> tag (li is an acronym for list item). Any tags can appear in a list item, including nested lists. When displayed, each list item is implicitly preceded by a bullet.

Eg: Grocery List

```
<!-- unordered.html
     An example to illustrate an unordered list
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Unordered list </title>
  </head>
  <body>
    <h3> Some Common Single-Engine Aircraft </h3>
    <ul>
      <li> Cessna Skyhawk </li>
      <li> Beechcraft Bonanza </li>
      <li> Piper Cherokee </li>
    </ul>
  </body>
</html>
```

Output



- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

Ordered Lists

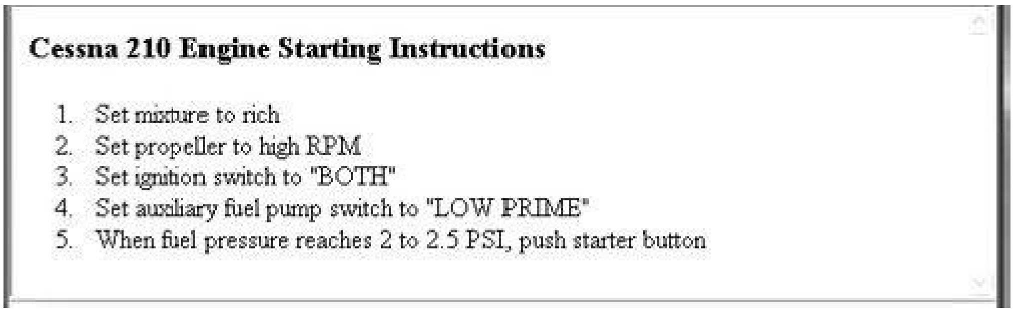
Ordered lists are lists in which the order of items is important. The order of a list is shown in the display of the list by the implicit attachment of a sequential value to the beginning of each item. The default sequential values are Arabic numerals, beginning with 1.

An ordered list is created within the block tag <ol>. The items are specified and displayed just as are those in unordered lists, except that the items in an ordered list are preceded by sequential values instead of bullets.

Eg:

```
<!-- ordered.html
     An example to illustrate an ordered list
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Ordered list </title>
  </head>
  <body>
    <h3> Cessna 210 Engine Starting Instructions </h3>
    <ol>
      <li> Set mixture to rich </li>
      <li> Set propeller to high RPM </li>
      <li> Set ignition switch to "BOTH" </li>
      <li> Set auxiliary fuel pump switch to "LOW PRIME" </li>
      <li> When fuel pressure reaches 2 to 2.5 PSI, push
           starter button
      </li>
    </ol>
  </body>
</html>
```

Output


Cessna 210 Engine Starting Instructions

1. Set mixture to rich
2. Set propeller to high RPM
3. Set ignition switch to "BOTH"
4. Set auxiliary fuel pump switch to "LOW PRIME"
5. When fuel pressure reaches 2 to 2.5 PSI, push starter button

Definition Lists

Definition lists are used to specify lists of terms and their definitions, as in glossaries. A definition list is given as the content of a <dl> tag, which is a block tag.

Each term to be defined in the definition list is given as the content of a <dt> tag. The definitions themselves are specified as the content of <dd> tags. The defined terms of a definition list are usually displayed in the left margin.

Differences between HTML & XHTML

Based on certain features, there are differences between HTML and XHTML. They are:

Case sensitivity: In HTML, tag and attribute names are case insensitive, meaning that <FORM>, <form>, and <Form> are equivalent. In XHTML, all tag and attribute names must be all lowercase.

Closing tags: In HTML, closing tags may be omitted if the processing agent (usually a browser). For example, in HTML, paragraph elements often do not have closing tags. In XHTML, all elements must have closing tags.

Quoted attribute values: In HTML, attribute values must be quoted only if there are embedded special characters or white-space characters. Numeric attribute values are rarely quoted in HTML. In XHTML, all attribute values must be double quoted, regardless of what characters are included in the value.

Explicit attribute values: In HTML, some attribute values are implicit; that is, they need not be explicitly stated. This table tag is invalid in XHTML, in which such an attribute is assigned a string of the name of the attribute.

id and name attributes: HTML uses the name attribute for elements. This attribute was deprecated for some elements in HTML 4.0, which added the id attribute to nearly all elements. In XHTML, the use of id is encouraged and the use of name is discouraged.

Element nesting: HTML has rules against improper nesting of elements, they are not enforced. In XHTML, these nesting rules are strictly enforced.

Forms in HTML

The most common way for a user to communicate information from a Web browser to the server is through a form. XHTML provides tags to generate the commonly used objects on a screen are required to fill out, forms can be described in XHTML and displayed by the browser. XHTML provides tags to generate the commonly used objects on a screen form. These objects are called controls or widgets.

There are controls for single-line and multiple-line text collection, checkboxes, radio buttons, and menus, among others. All control tags are inline tags. Most controls are used to gather information from the user in the form of either text or button selections. Each control can have a value, usually given through user input. Together, the values of all of the controls (that have values) in a form are called the form data. Every form requires a Submit button. When the user clicks the Submit button, the form data is encoded and sent to the Web server for processing.

<form> Tag

All of the controls of a form appear in the content of a <form> tag. A block tag, <form>, can have several different attributes, only one of which, action, is required. The action attribute specifies the URL of the application on the Web server that is to be called when the user clicks the Submit button.

The **method** attribute of <form> specifies one of the two techniques, get or post, used to pass the form data to the server. The default is get, so if no method attribute is given in the <form> tag, get will be used. The alternative technique is post. In both techniques, the form data is coded into a text string when the user clicks the Submit button.

When the get method is used, the browser attaches the query string to the URL of the HTTP request, so the form data is transmitted to the server together with the URL. The browser inserts a question mark at the end of the actual URL just before the first character of the query string so that the server can easily find the beginning of the query string.

When the post method is used, the query string is passed by some other method to the form-processing program. There is no length limitation for the query string with the post method.

<input> Tag

Many of the commonly used controls are specified with the inline tag <input>, including those for text, passwords, checkboxes, radio buttons, and the action buttons Reset, Submit, and plain. The one attribute of <input> that is required for all of the controls discussed in this section is type, which specifies the particular kind of control. The control's kind is its type name, such as checkbox.

For multiple choices, Select tag is used. For multiple area,we use <textarea> tag is used.

XHTML document must begin with an xml declaration element that simply identifies the document as being one based on XML. This element includes an attribute that specifies the version number, which is still 1.0. The xml declaration usually includes a second attribute, encoding, which specifies the encoding used for the document.

Images in HTML

The image is stored in a file, which is specified by an XHTML request. The image is inserted into the display of the document by the browser. Images format suitable for faster transfer over the Internet are Graphic Interchange Format (**GIF**-8bit) and the Joint Photographic Experts Group (**JPEG**-24 bit). Files in both formats are compressed to reduce storage needs and provide.

E.g: **<img src="URL" alt="alternative text" />**

The image tag, <**img** />, which is an inline tag, specifies an image that is to appear in a document. It includes two attributes: **src**, which specifies the file containing the image; and **alt**, which specifies text to be displayed when it is not possible to display the image.

If the file is in the same directory as the XHTML file of the document, the value of src is just the image's file name. Otherwise specify the complete url of the image.

Two optional attributes of img—**width** and **height**—can be included to specify (in pixels) the size of the rectangle for the image.

Hypertext link acts as a pointer to some particular place in some Web resource. That resource can be an XHTML document anywhere on the Web, or it may be the document currently being displayed. A link that points to a different resource specifies the address of that resource. Such an address might be a file name, a directory path and a file name, or a complete URL.

Links are specified in an attribute of an anchor tag (<a>), which is an inline tag. The anchor tag that specifies a link is called the **source** of that link. The document whose address is specified in a link is called the **target** of that link. for creating links, only one attribute is required: **href** (short form for hypertext reference). The value assigned to href specifies the target of the link.

If the target is in another document in the same directory, the target is just the document's file name. If the target document is in some other directory, the UNIX pathname conventions are used.

If the target of a link is some element within the document, in which case there must be some means of specifying it. The target element can include an **id** attribute, which can then be used to identify it in an href attribute. The value of an id attribute must be unique within the document. Consider the following example:

<h2 id = "avionics"> Avionics </h2>

If the target is in the same document as the link, the target is specified in the href attribute value by preceding the id value with a pound sign (#).

<a href = "#avionics"> What about avionics? </a>

Web Technology

Module –III

Cascading Style Sheets

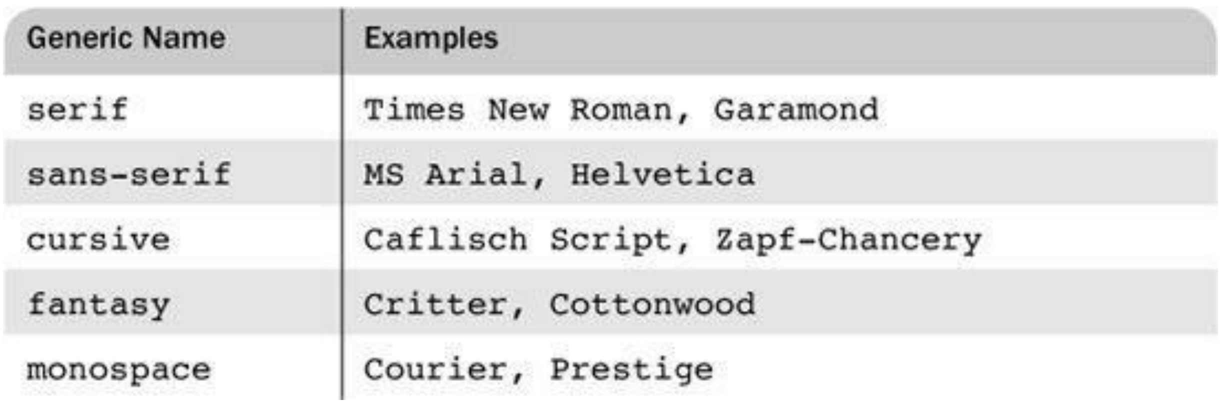
(Ref: Robert Sebesta, Programming the world wide web)

Font Properties

The font properties are among the most commonly used of the style-sheet properties. The **font-family property** is used to specify a list of font names. The browser uses the first font in the list that it supports. For example, the property:

font-family: Arial, Helvetica, Futura .

It tells the browser to use Arial if it supports that font. If not, it will use Helvetica if it supports it. If the browser supports neither Arial nor Helvetica, it will use Futura if it is supported by browser. If the browser does not support any of the specified fonts, it will use an alternative of its choosing. A generic font can be specified as a font-family value. An best approach is to specify fonts is to use a generic font as the last font in the value of a font-family property. For example, because Arial, Helvetica, and Futura are sans-serif fonts.



Generic Name	Examples
serif	Times New Roman, Garamond
sans-serif	MS Arial, Helvetica
cursive	Caflisch Script, Zapf-Chancery
fantasy	Critter, Cottonwood
monospace	Courier, Prestige

If a font name has more than one word, the whole name should be delimited by single quotes. For example:font-family: ‘Times New Roman’.

Font Sizes

The font-size property sets the size of font. For example, the following property specification sets the font size for text to 10 points:

font-size: 10pt

Many relative font-size values are defined, including xx-small, x-small, small, medium, large, x-large, and xx-large. In addition, smaller or larger can be specified. The value can be a percentage value.

Font Variants

The default value of the font-variant property is normal, which specifies the usual character font. This property can be set to small-caps to specify small capital characters.

Font Styles

The font-style property is most commonly used to specify italic, as in Eg:

font-style: italic

An alternative to italic is oblique.

Font Weights

The font-weight property is used to specify the degree of boldness. For eg:

font-weight: bold

The values normal (the default), bolder, and lighter can be specified. The bolder and lighter values are taken as relative to the current level of boldness. Specific numbers also can be given in multiples of 100 from 100 to 900, where 400 is the same as normal and 700 is the same as bold.

Font Shorthands

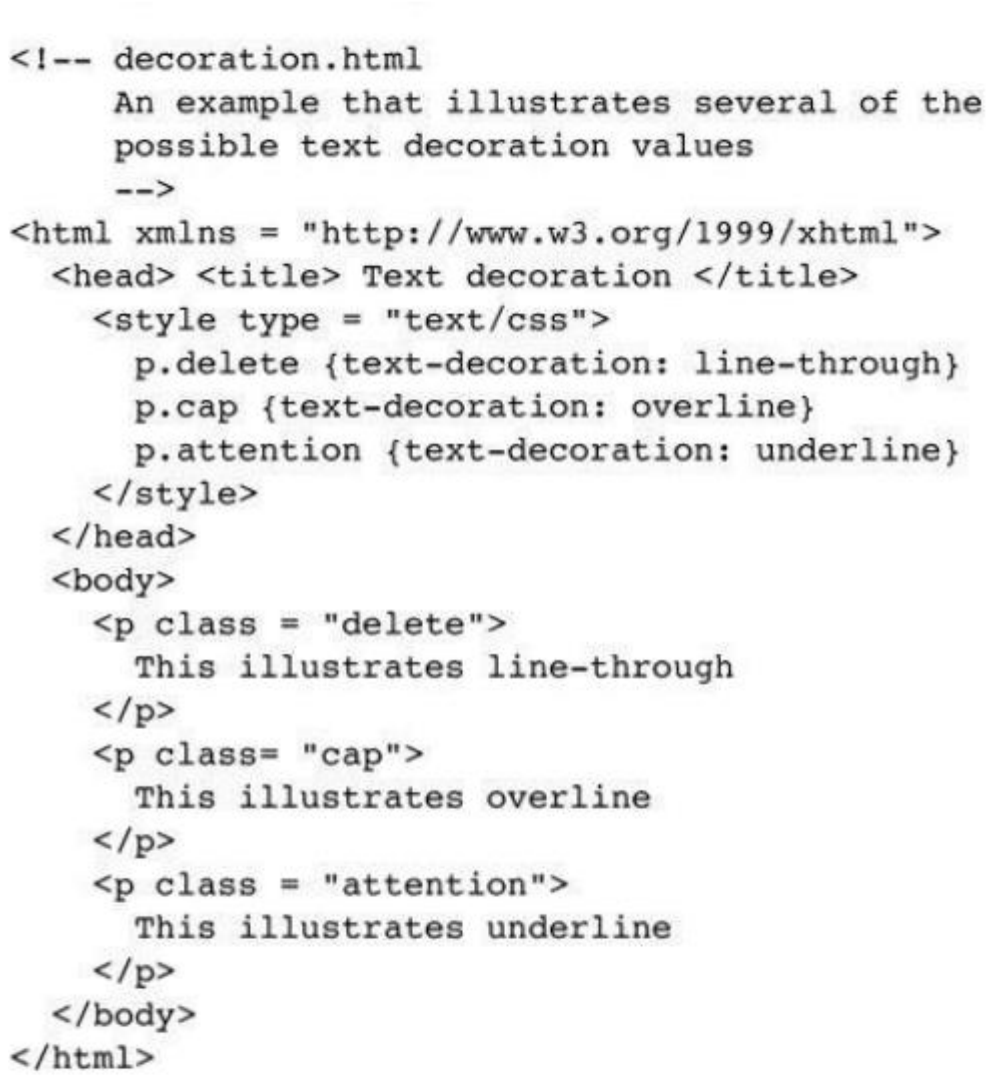
If more than one font property must be specified, the values can be stated in a list as the value of the font property. The browser then has the responsibility for determining which properties to assign from the forms of the values. For example, the property

font: bold 14pt ‘Times New Roman’ Palatino

This specifies that the font weight should be bold, the font size should be 14 points, and either Times New Roman or Palatino font should be used, with precedence given to Times New Roman.

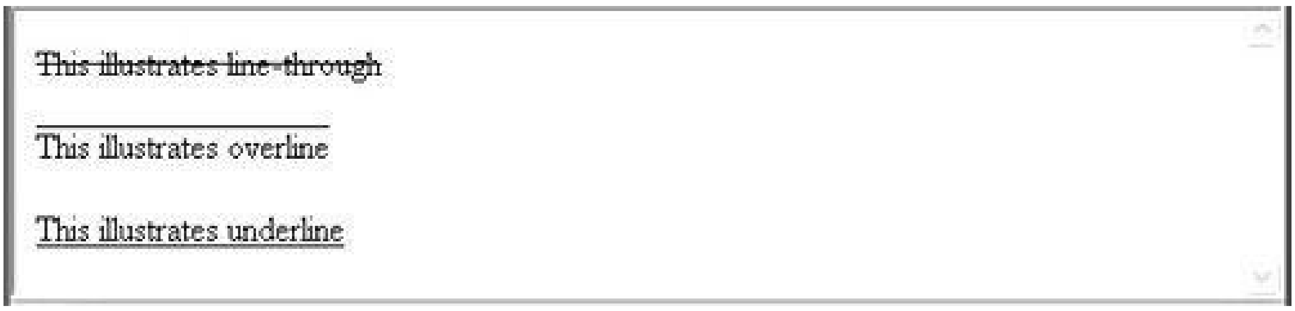
Text Decoration

The text-decoration property is used to specify some special features of text. The available values are line-through, overline, underline, and none, which is the default. Many browsers implicitly underline links. The none value can be used to avoid this underlining.



```
<!-- decoration.html
     An example that illustrates several of the
     possible text decoration values
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Text decoration </title>
    <style type = "text/css">
      p.delete {text-decoration: line-through}
      p.cap {text-decoration: overline}
      p.attention {text-decoration: underline}
    </style>
  </head>
  <body>
    <p class = "delete">
      This illustrates line-through
    </p>
    <p class= "cap">
      This illustrates overline
    </p>
    <p class = "attention">
      This illustrates underline
    </p>
  </body>
</html>
```

The output of the following program is:


~~This illustrates line-through~~

This illustrates overline

This illustrates underline

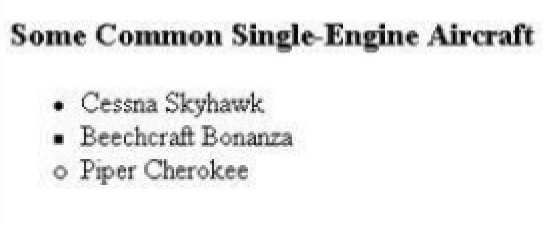
The letter-spacing property controls the amount of space between characters in text. The possible values of letter-spacing are any length property values.

List Properties

The list-style-type property is used to specify both the shape of the bullets that precede the items in an unordered list and the sequencing values that precede the items in an ordered list. The list-style-type property of an unordered list can be set to disc, circle, square, or none. A disc is a small filled circle, a circle is an unfilled circle, and a square is a filled square. The default property value for bullets is disc. Bullets in unordered lists are not limited to discs, squares, and circles. Any image can be used in a list item bullet. Such a bullet is specified with the list-style-image property, whose value is specified with the url form. For Eg:

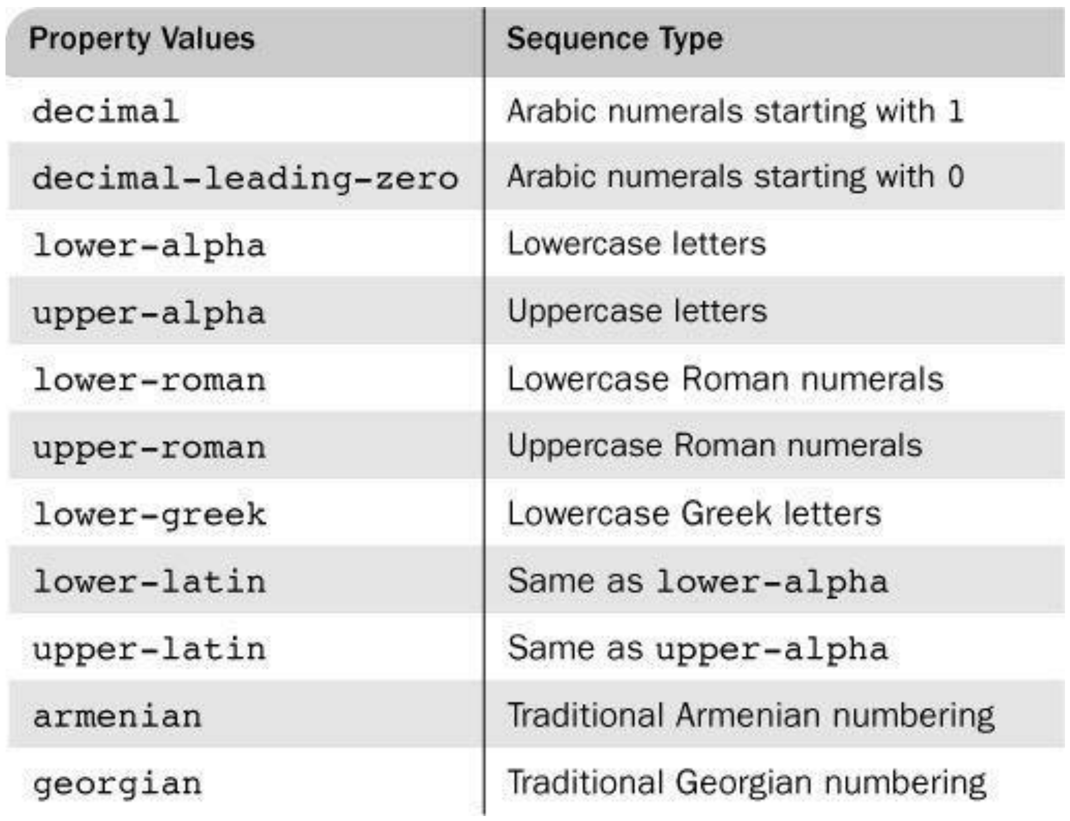
```
<!-- bullets2 -->
<style type = "text/css">
  li.disc {list-style-type: disc}
  li.square {list-style-type: square}
  li.circle {list-style-type: circle}
</style>
...
<h3> Some Common Single-Engine Aircraft </h3>
  <ul>
    <li class = "disc"> Cessna Skyhawk </li>
    <li class = "square"> Beechcraft Bonanza </li>
    <li class = "circle"> Piper Cherokee </li>
  </ul>
```

Output:


Some Common Single-Engine Aircraft

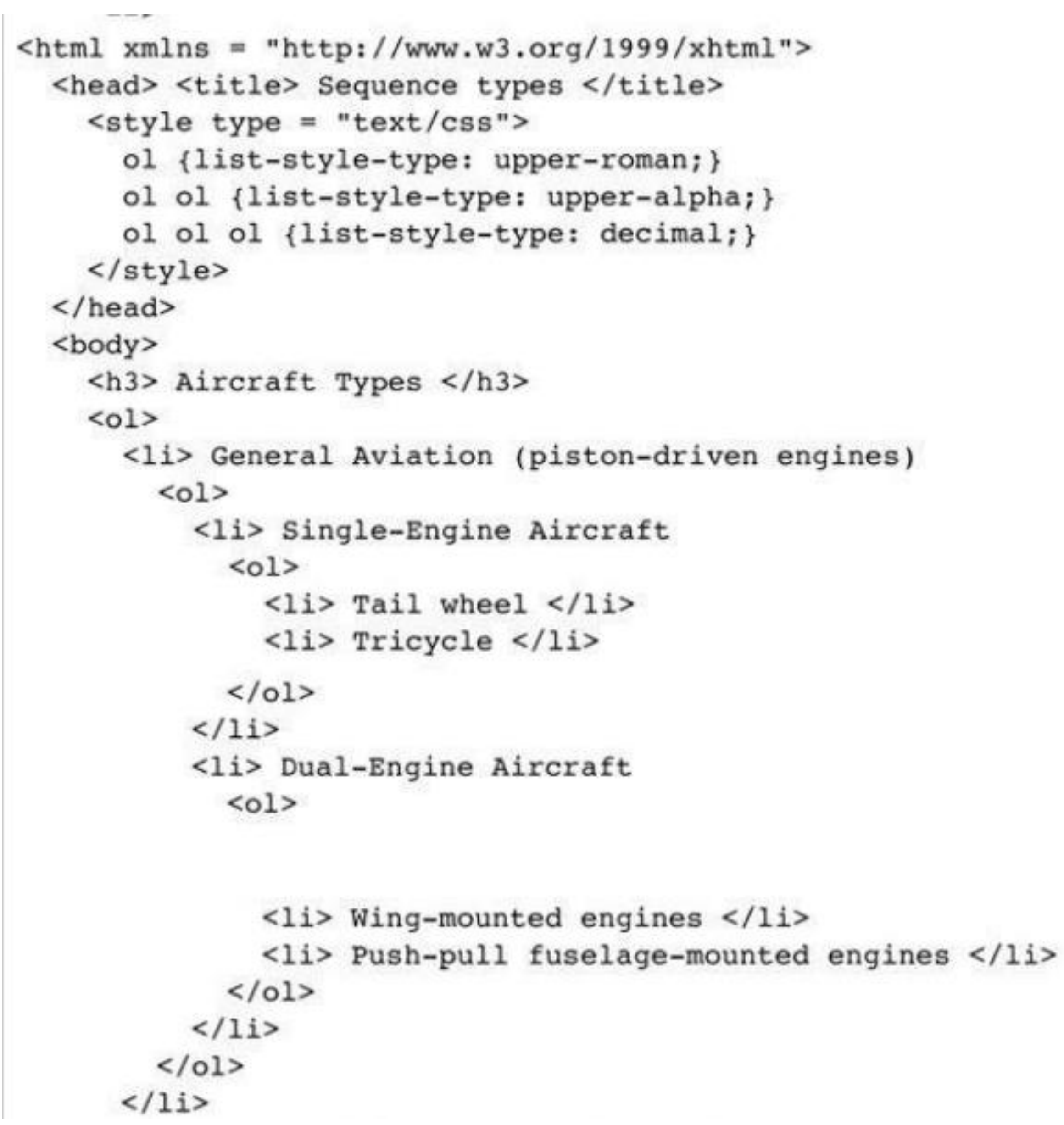
- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

When ordered lists are nested, it is best to use different kinds of sequence values for the different levels of nesting. The list-style-type property can be used to specify the types of sequence values.



Property Values	Sequence Type
`decimal`	Arabic numerals starting with 1
`decimal-leading-zero`	Arabic numerals starting with 0
`lower-alpha`	Lowercase letters
`upper-alpha`	Uppercase letters
`lower-roman`	Lowercase Roman numerals
`upper-roman`	Uppercase Roman numerals
`lower-greek`	Lowercase Greek letters
`lower-latin`	Same as `lower-alpha`
`upper-latin`	Same as `upper-alpha`
`armenian`	Traditional Armenian numbering
`georgian`	Traditional Georgian numbering

The Program for the ordered list property is shown below:

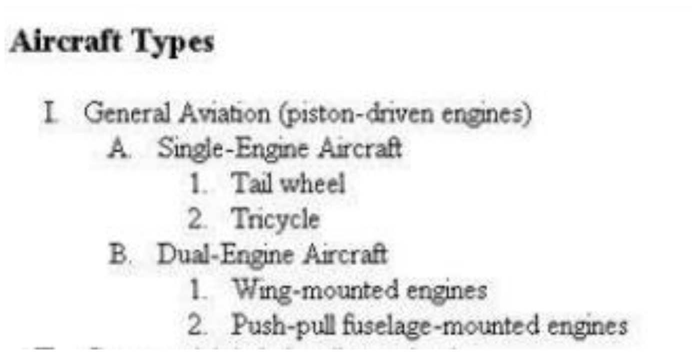


```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Sequence types </title>
    <style type = "text/css">
      ol {list-style-type: upper-roman;}
      ol ol {list-style-type: upper-alpha;}
      ol ol ol {list-style-type: decimal;}
    </style>
  </head>
  <body>
    <h3> Aircraft Types </h3>
    <ol>
      <li> General Aviation (piston-driven engines)
        <ol>
          <li> Single-Engine Aircraft
            <ol>
              <li> Tail wheel </li>
              <li> Tricycle </li>

            </ol>
          </li>
          <li> Dual-Engine Aircraft
            <ol>


              <li> Wing-mounted engines </li>
              <li> Push-pull fuselage-mounted engines </li>
            </ol>
          </li>
        </ol>
      </li>
```

Output:



```
Aircraft Types

  I. General Aviation (piston-driven engines)
       A. Single-Engine Aircraft
            1. Tail wheel
            2. Tricycle
       B. Dual-Engine Aircraft
            1. Wing-mounted engines
            2. Push-pull fuselage-mounted engines
```

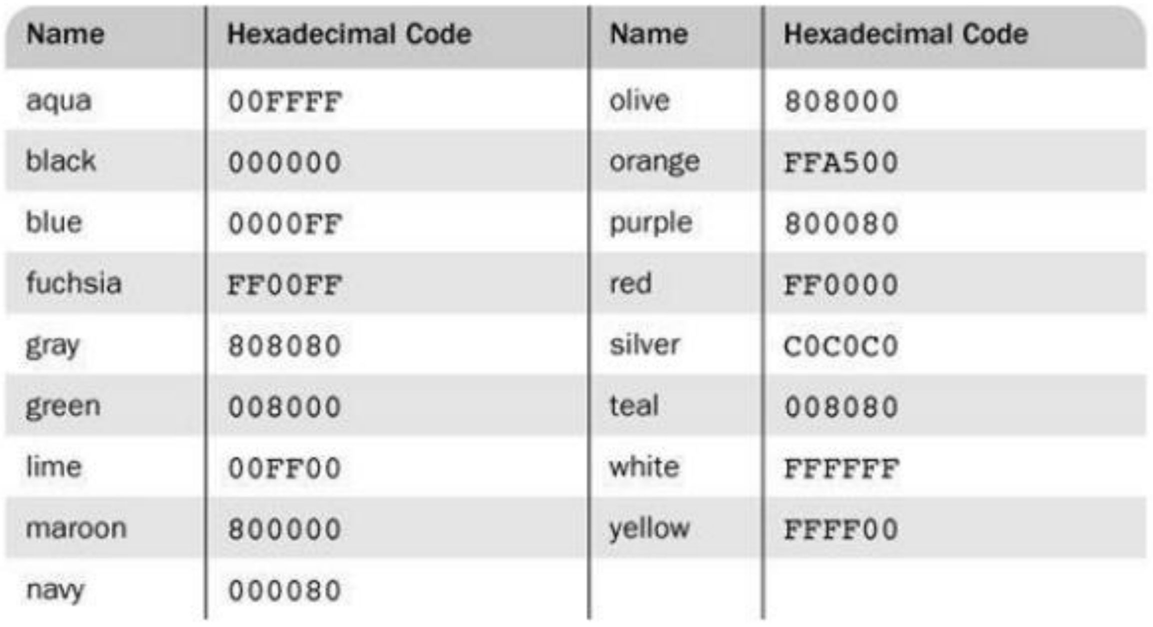
Color

Most of the color names recognized by browsers prevent CSS validation.

Color Groups

Three levels of collections of colors might be used by an XHTML document. The smallest useful set of colors includes only those that have standard names and are guaranteed to be correctly displayable by all browsers on all color monitors. This collection of 17 colors is called the named colors. These are the

Table 3.3 Named colors



Name	Hexadecimal Code	Name	Hexadecimal Code
aqua	00FFFF	olive	808000
black	000000	orange	FFA500
blue	0000FF	purple	800080
fuchsia	FF00FF	red	FF0000
gray	808080	silver	C0C0C0
green	008000	teal	008080
lime	00FF00	white	FFFFFF
maroon	800000	yellow	FFFF00
navy	000080		

only color names that allow CSS validation.

A larger set of colors, called the Web palette, consists of 216 colors. These colors, which are often called Web-safe colors, are displayable by Windows- and Macintosh-based browsers, but may not be correctly displayed with some old terminals used on UNIX systems. Elements of this set of colors have hexadecimal values for red, green, and blue that are restricted to 00, 33, 66, 99, CC, and FF.

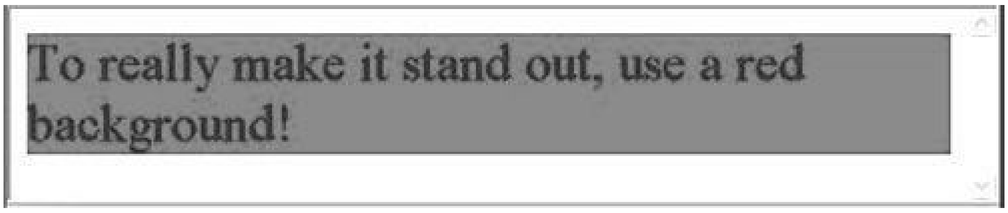
Color Properties

The color property is used to specify the foreground color of XHTML elements.

The background-color property is used to set the background color of an element, where the element could be the whole body of the document. For Eg:

```
<style type = "text/css">
  p.standout {font-size: 24pt; color: blue;
              background-color: red"}
</style>
...
<p class = "standout">
  To really make it stand out, use a red background!
</p>
```

Output:



To really make it stand out, use a red background!

Alignment of Text

The text-indent property can be used to indent the first line of a paragraph. This property takes either a length or a percentage value. For Eg:

```
<style type = "text/css">
  p.indent {text-indent: 0.5in}
</style>
...
<p class = "indent">
  Now is the time for all good Web programmers to begin
  using cascading style sheets for all presentation
  details in their documents. No more deprecated tags
  and attributes, just nice, precise style sheets.
</p>
```

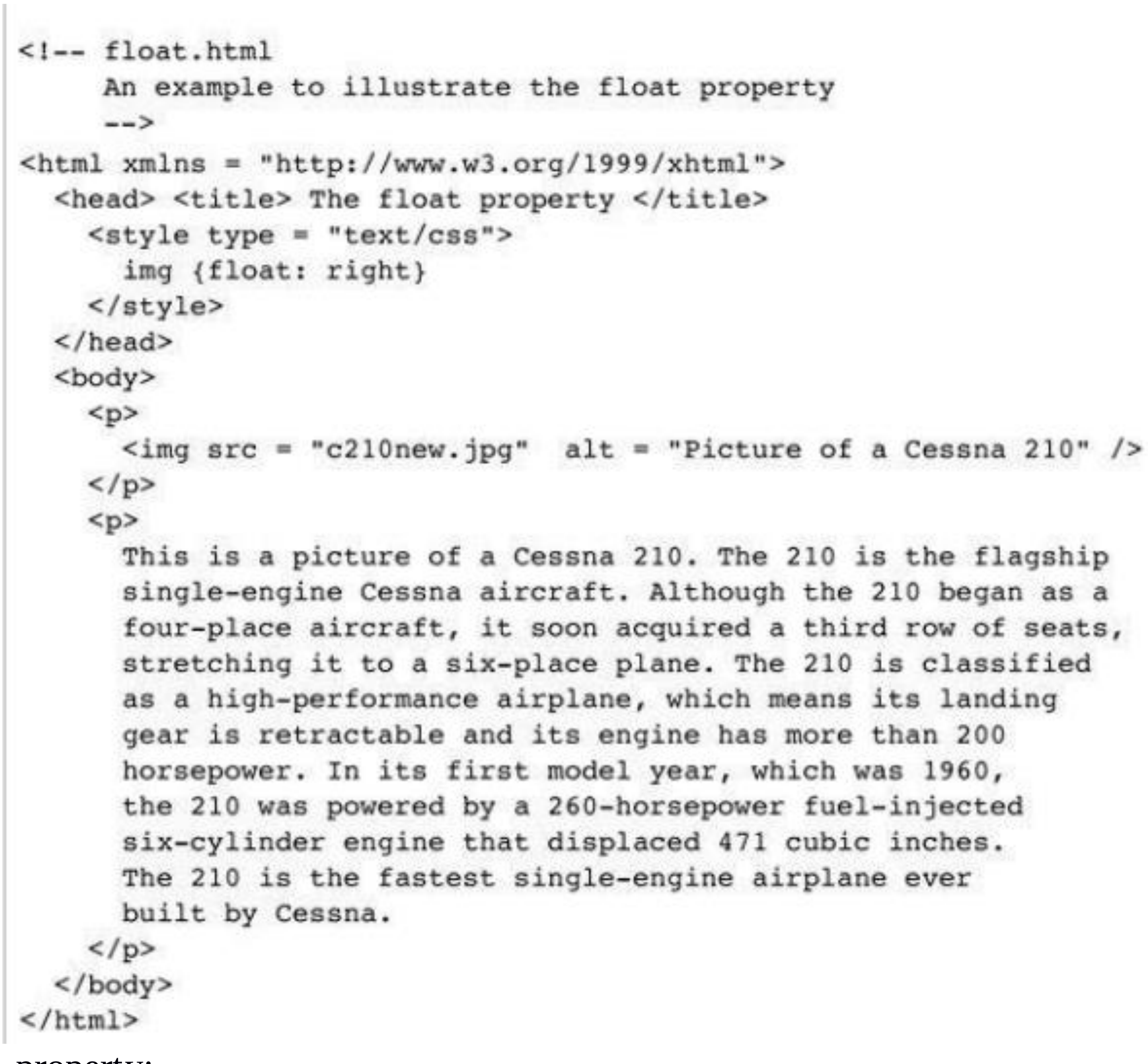
This paragraph would be displayed as follows:

```
      Now is the time for all good Web programmers to begin
using cascading style sheets for all presentation details
in their documents. No more deprecated tags and attributes,
just nice, precise style sheets.
```

The text-align property, for which the possible keyword values are left, center, right, and justify, is used to arrange text horizontally. For example, the following document-level style sheet entry causes the content of paragraphs to be aligned on the right margin:

p {text-align: right}

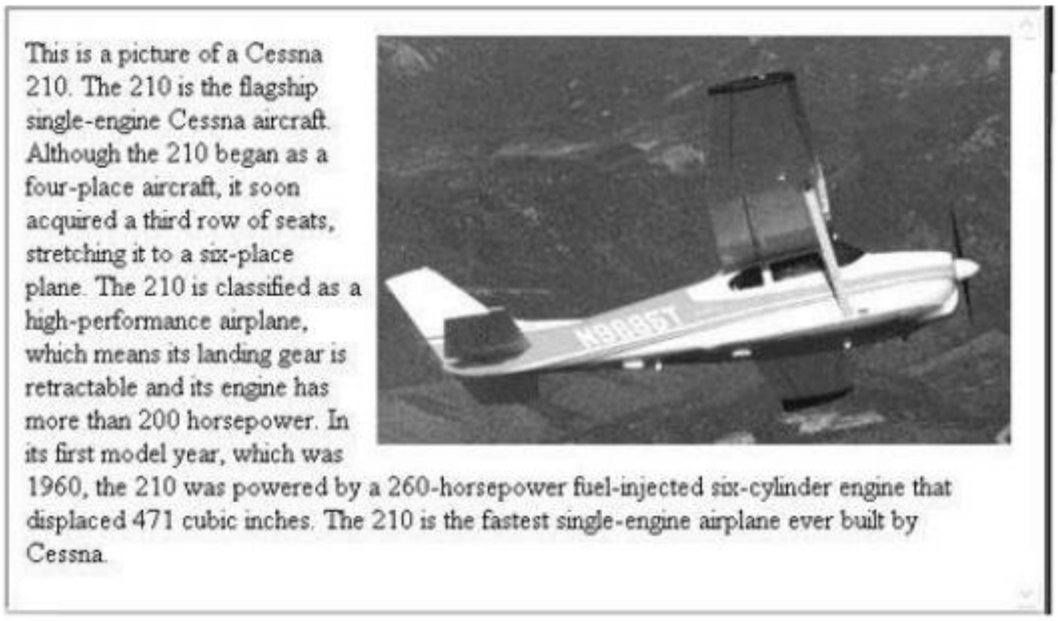
The default value for text-align is left. The float property is used to specify that text should flow around some element, often an image or a table. The possible values for float are left, right, and none, which is the default. Example for float



```
<!-- float.html
     An example to illustrate the float property
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> The float property </title>
    <style type = "text/css">
      img {float: right}
    </style>
  </head>
  <body>
    <p>
      <img src = "c210new.jpg"  alt = "Picture of a Cessna 210" />
    </p>
    <p>
      This is a picture of a Cessna 210. The 210 is the flagship
      single-engine Cessna aircraft. Although the 210 began as a
      four-place aircraft, it soon acquired a third row of seats,
      stretching it to a six-place plane. The 210 is classified
      as a high-performance airplane, which means its landing
      gear is retractable and its engine has more than 200
      horsepower. In its first model year, which was 1960,
      the 210 was powered by a 260-horsepower fuel-injected
      six-cylinder engine that displaced 471 cubic inches.
      The 210 is the fastest single-engine airplane ever
      built by Cessna.
    </p>
  </body>
</html>
```

property:

Output:



This is a picture of a Cessna 210. The 210 is the flagship single-engine Cessna aircraft. Although the 210 began as a four-place aircraft, it soon acquired a third row of seats, stretching it to a six-place plane. The 210 is classified as a high-performance airplane, which means its landing gear is retractable and its engine has more than 200 horsepower. In its first model year, which was 1960, the 210 was powered by a 260-horsepower fuel-injected six-cylinder engine that displaced 471 cubic inches. The 210 is the fastest single-engine airplane ever built by Cessna.

The Box Model

Virtually all document elements can have borders with various styles, such as color and width. The amount of space between the content of an element and its border, known as padding, can be specified, as well as the space between the border and an adjacent element, known as the margin. This model is represented as:

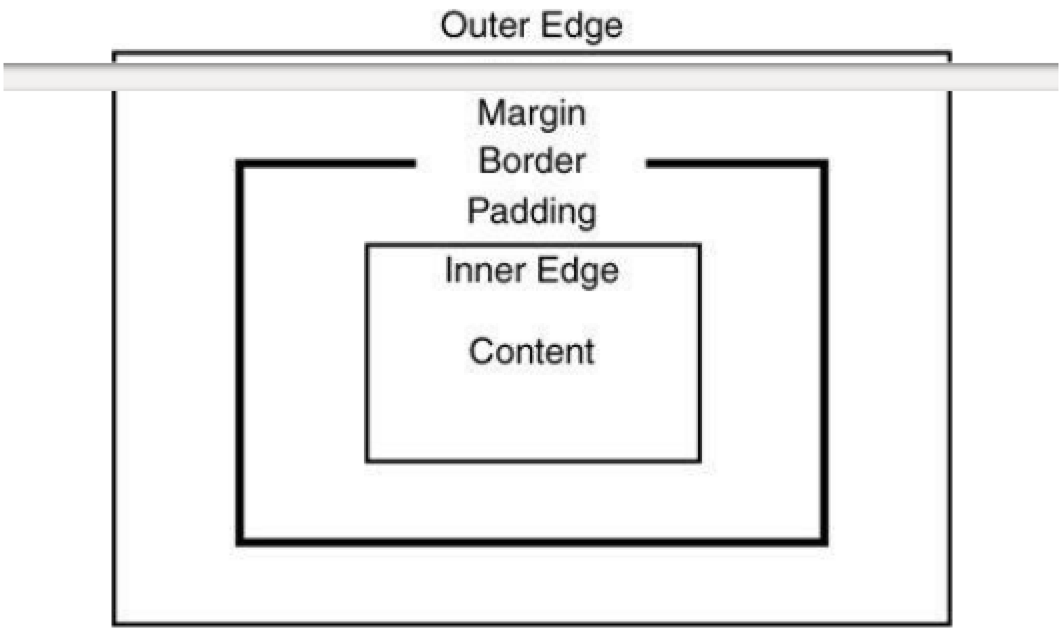


Figure 3.7 The box model

Borders

Every element has a property, border-style, that controls whether the element's content has a border, as well as specifying the style of the border. CSS provides several different border styles. They are: **dotted, dashed, and double**. The default value for border-style is none, which is why the contents of elements do not normally have borders. The styles of one particular side of an element can be set with border-top-style, border-bottom-style, border-left-style, and border-right-style.

The border-width property is used to specify the thickness of a border. Its possible values are thin, medium (the default), thick, or a length value in pixels. Setting border-width sets the thickness of all four sides of an element. The widths of the four borders of an element can be different and are specified with border-top-width, border-bottom-width, border-left-width, and border-right-width.

The color of a border is controlled by the border-color property. Once again, the individual borders of an element can be colored differently through the properties border-top-color, border-bottom-color, border-left-color, and border-right-color.

Margins and Padding

Padding is the space between the content of an element and its border. The margin is the space between the border of an element and the element's neighbor. When there is no border, the margin plus the padding is the space between the content of an element and its neighbors.

The margin properties are named margin, which applies to all four sides of an element: margin-left, margin-right, margin-top, and margin- bottom. The padding properties are named padding, which applies to all four sides: padding-left, padding-right, padding-top, and padding-bottom.

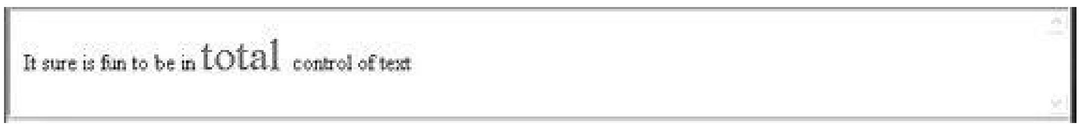
Background Images

The background-image property is used to place an image in the background of an element. For example, an image of an airplane might be an effective background for text about the airplane. The background image is replicated as necessary to fill the area of the element. This replication is called tiling. Tiling can be controlled with the background-repeat property, which can take the value repeat (the default), no-repeat, repeat-x, or repeat-y. The no-repeat value specifies that just one copy of the image is to be displayed. The repeat-x value means that the image is to be repeated horizontally; repeat-y means that the image is to be repeated vertically. In addition, the position of a non repeated background image can be specified with the background-position property, which can take a large number of different values. The keyword values are top, center, bottom, left, and right.

<span> and <div> Tags

We want to apply special font properties to less than a whole paragraph of text. <span> tag can be used for that purpose. In the following example, the word total is not displayed differently from the rest of the paragraph:

```
<style type = "text/css" >
  .spanred {font-size: 24pt;
             font-family: Ariel; color: red}
</style>
...
<p>
  It sure is fun to be in
  <span class = "spanred"> total </span>
  control of text
</p>
```

Output: 

```
It sure is fun to be in total control of text
```

If we want to apply a style to a section of a document rather than to each paragraph. This can be done with the <div> tag. its primary use is to specify presentation details for a section or division of a document.

Frameworks

Responsive Design

Responsive design is an approach to web page creation that makes use of flexible layouts, flexible images and cascading style sheet media queries. The goal of responsive design is to build web pages that detect the visitor's screen size and orientation and change the layout accordingly.

Responsive design pages use x and y coordinates on a grid for layout and mathematical percentages for images instead of fixed- width parameters. Media queries, a feature of cascading style sheets (CSS), allow the developer to specify when a certain style takes effect.

CSS FRAMEWORK

A framework is a standardized set of concepts, practices and criteria for dealing with a common type of problem, which can be used as a reference to help us approach and resolve new problems of a similar nature. It can be also defined as a package made up of a structure of files and folders of standardized code (HTML, CSS, JS documents etc.) which can be used to support the development of websites, as a basis to start building a site.

Front end frameworks usually consist of a package made up of a structure of files and folders of standardized code (HTML, CSS, JS documents etc.) The usual components are: CSS source code to create a grid and this allows the developer to position the different elements that make up the site design in a simple and versatile fashion. Typography style definitions for HTML elements. Solutions for cases of browser incompatibility so the site displays correctly in all browsers. Creation of standard CSS classes which can be used to style advanced components of the user interface.There are two types of framework according to their complexity: **simple frameworks and complete frameworks.**

COMPLETE FRAMEWORKS

They usually offer complete frameworks with configurable features like styled-typography, sets of forms, buttons, icons and other reusable components built to provide navigation, alerts, popovers, and more, images frames, HTML templates, custom settings, etc.

Criteria for choosing best frameworks

Speed of installation : Framework with minimum installation time and steps shuld be selected.

Ease of understanding:

Options: some frameworks are more complex than others and offer more configuration options,widgets and interface options. These will allow you to do better things with your site.

Integration with other systems.

Best long-term support: Some digital projects lack continuity in time and updates and support services stop. It's always better to opt for those that offer continued support guarantees.

Bootstrap Framework

Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS, and JavaScript. Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter. It was released as an open source product in August 2011 on GitHub. Thr main advantages of using bootstrap framework is:

Mobile first approach: Bootstrap 3 framework consists of Mobile first styles throughout the entire library instead of them in separate files.

Browser Support: It is supported by all popular browsers.

Easy to get started: With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has a good documentation.

Responsive design: Bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles.

This framework Provides a clean and uniform solution for building an interface for developers. It contains beautiful and functional built-in components which are easy to customize. It also provides web-based customization and best of all it is an open source.

Bootstrap Package

The bootstrap package includes:

Scaffolding: Bootstrap provides a basic structure with Grid System, link styles, and background.

CSS: Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system.

Components: Bootstrap contains over a dozen reusable components built to provide iconography, drop downs, navigation, alerts, pop-overs, and much more.

JavaScript Plugins: Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one.

Customize: You can customize Bootstrap's components, LESS variables, and Query plugins to get your very own version.

Web Technology

Module –IV

Introduction to JavaScript and jQuery

(Ref: Robert Robert W Sebesta, Programming the World Wide Web, 7/e)

JavaScript Objects

In JavaScript, objects are collections of properties, which correspond to the members of classes in Java and C++. Each property is either a data property or a function or method property. Data properties appear in two categories: primitive values and references to other objects. JavaScript uses non object types for some of its simplest types; these non-object types are called **primitives**. Primitives are used because they often can be implemented directly in hardware, resulting in faster operations on their values. All objects in a JavaScript program are indirectly accessed through variables. Such a variable is like a reference in Java. All primitive values in JavaScript are accessed directly. These are often called **value types**. The root object in JavaScript is Object. All functions are objects and are referenced through variables. The collection of properties of a JavaScript object is dynamic i.e, Properties can be added or deleted at any time.

Differences between Java and JavaScript

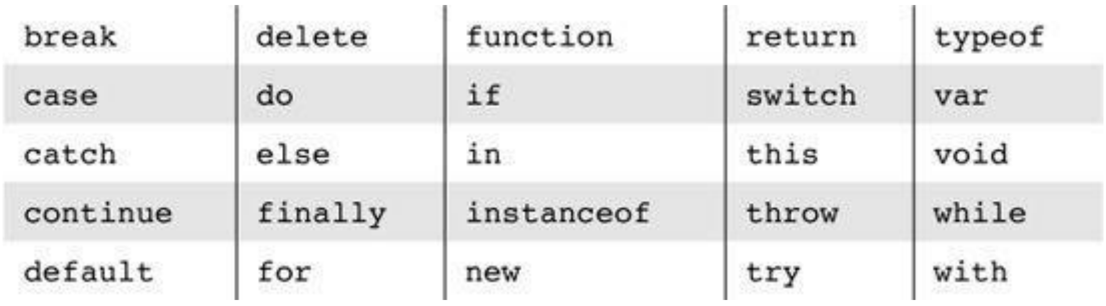
Java	JavaScript
Strongly typed language. Types are known at compile time.	Dynamically typed language. Types are given at run time
Objects used are static	Objects used are dynamic
Requires JVM to run the code	Requires a browser with JavaScript interpreter.
Compiled on server before execution on client.	Interpreted by client

Distinct from XHTML code	Can be placed between HTML & XHTML codes.
Source code is hidden from the user	Source code is accessible to the user
Object oriented	Object based

General Syntactic Characteristics

All JavaScript scripts are embedded, either directly or indirectly, in XHTML documents. Scripts can appear directly as the content of a <script> tag. The type attribute of <script> must be set to "text/JavaScript". The JavaScript script can be indirectly embedded in an XHTML document with the src attribute of a <script> tag, whose value is the name of a file that contains the script. In JavaScript, identifiers, or names, are similar to those of other common programming languages. They must begin with a letter, an underscore (_), or a dollar sign ($). Subsequent characters may be letters, underscores, dollar signs, or digits. There is no length limitation for identifiers. JavaScript is a case sensitive language.

JavaScript has 25 reserved words, which are used for different purposes.



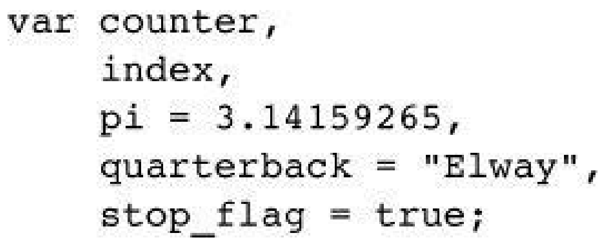
break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	finally	instanceof	throw	while
default	for	new	try	with

Commenting in JavaScript

JavaScript has two forms of comments, both of which are used in other languages. First, whenever two adjacent slashes (//) appear on a line, the rest of the line is considered a comment. Second, /* may be used to introduce a comment, and */ to terminate it, in both single- and multiple-line comments. The XHTML comment used to hide JavaScript uses the normal beginning syntax, <!- -Semicolon is sometimes used as a delimiter.

Declaring Variables

A variable can have the value of any primitive type, or it can be a reference to any object. The type of the value of a particular appearance of a variable in a program can be determined by the interpreter. A variable can be declared either by assigning it a value, in which case the interpreter implicitly declares it to be a variable, or by listing it in a declaration statement that begins with the reserved word **var**. Initial values can be included in a **var** declaration. For Eg:



```
var counter,
    index,
    pi = 3.14159265,
    quarterback = "Elway",
    stop_flag = true;
```

Numeric Operators

JavaScript has the typical collection of numeric operators: the binary operators + for addition, - for subtraction, * for multiplication, / for division, and % for modulus. The unary operators are plus (+), negate (-), decrement (--), and increment (++). The increment and decrement operators can be either prefix or postfix form. All numeric operations are done in double-precision floating point. The precedence rules of a language specify which operator is evaluated first when two operators with different precedence are adjacent in an expression. Adjacent operators are separated by a single operand. Multiplication has more precedence than addition satisfying BODMAS rule. The associativity rules of a language specify which operator is evaluated first when two operators with the same precedence are adjacent in an expression.

The Math Object

The Math object provides a collection of properties of Number objects and methods that operate on Number objects. The Math object has methods for the trigonometric functions, such as sin (for sine) and cos (for cosine), as well as for other commonly used mathematical operations. Among these are floor, to truncate a number; round, to round a number; and max, to return the largest of two given numbers.

The Number Object

The Number object includes a collection of useful properties that have constant values. These properties are referenced through Number. For example, Number. MIN_VALUE, which references the smallest representable number on the computer being used. The Number object has a method, toString, which it inherits from Object but overrides. The toString method converts the number through which it is called to a string. NaN denotes not a number. Method used for checking whether given one is a number or not is isNaN().

For Eg: isNaN(6) – returns false and isNaN(a) – returns true

String Catenation Operator

JavaScript strings are not stored or treated as arrays of characters; rather, they are unit scalar values. String catenation is specified with the operator denoted by a plus sign (+).

Implicit Type Conversions

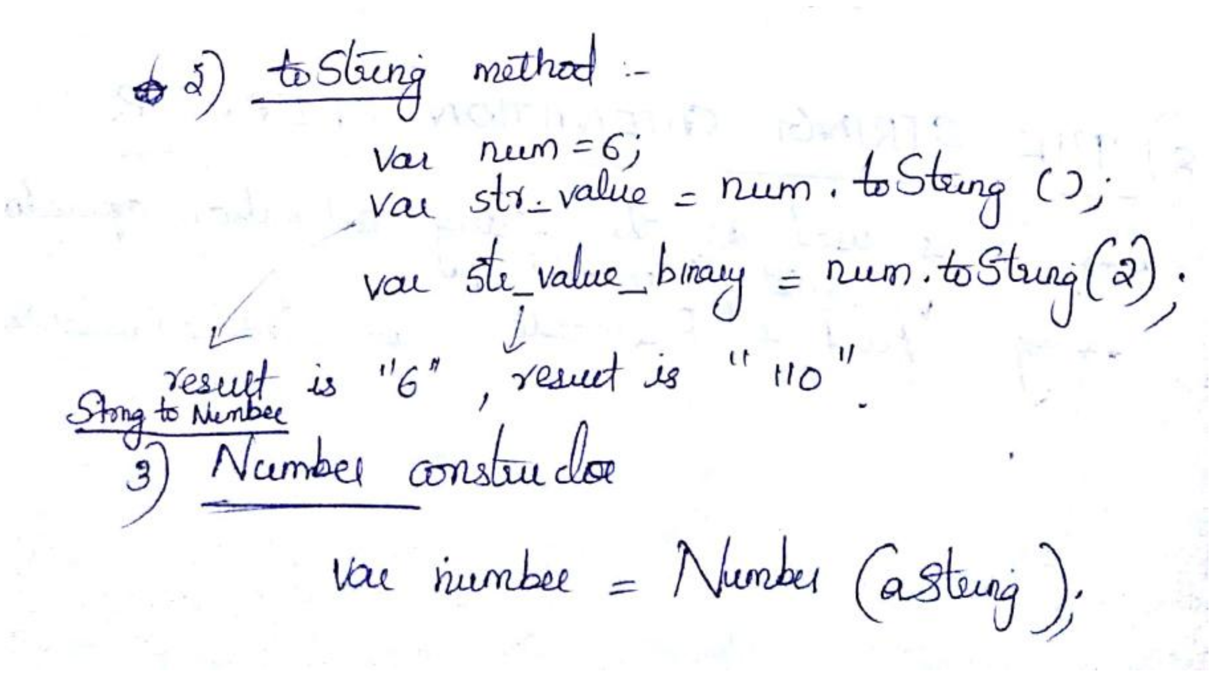
The JavaScript interpreter performs several different implicit type conversions. Such conversions are called **coercions**. In general, when a value of one type is used in a position that requires a value of a different type, JavaScript attempts to convert the value to the type that is required. The most common examples of these conversions involve primitive string and number values.

Explicit Type Conversions

There are several different ways to force type conversions, primarily between strings and numbers. Strings that contain numbers can be converted to numbers with the String constructor, as in the following code:

var str_value = String(value);

Other Methods


2) toString method :-

```
var num = 6;
var str_value = num.toString ();
var str_value_binary = num.toString(2);
```

result is "6" , result is "110".

String to Number

3) Number constructor

```
var number = Number (aString);
```

ParseInt and ParseFloat are also used to convert string to number.

String Properties and Methods

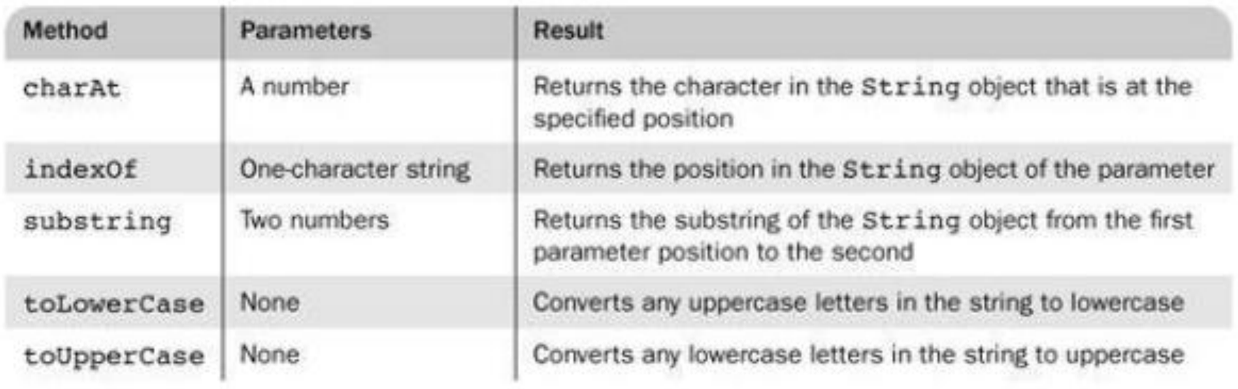
String methods can always be used through String primitive values, as if the values were objects. The String object includes one property, length, and a large collection of methods. The number of characters in a string is stored in the length property as follows:

var str = “George”;

var len = str. length;

In this code, **len** is set to the number of characters in str and answer is 6.

Table 4.4 `String` methods



Method	Parameters	Result
`charAt`	A number	Returns the character in the `String` object that is at the specified position
`indexOf`	One-character string	Returns the position in the `String` object of the parameter
`substring`	Two numbers	Returns the substring of the `String` object from the first parameter position to the second
`toLowerCase`	None	Converts any uppercase letters in the string to lowercase
`toUpperCase`	None	Converts any lowercase letters in the string to uppercase

For example, suppose str has been defined as follows:

var str = "George";

Then the following expressions have the values shown:

str. CharAt(2) - Answer is 'o'

str. indexOf('r') Answer is '3'

str. substring(2, 4) Result is 'org'

str. toLowerCase() Result is 'george'

Assignment Statements

The assignment statement in JavaScript is exactly like the assignment statement in other common C-based programming languages. There is a simple assignment operator, denoted by =, and a host of compound assignment operators, such as += and /=.

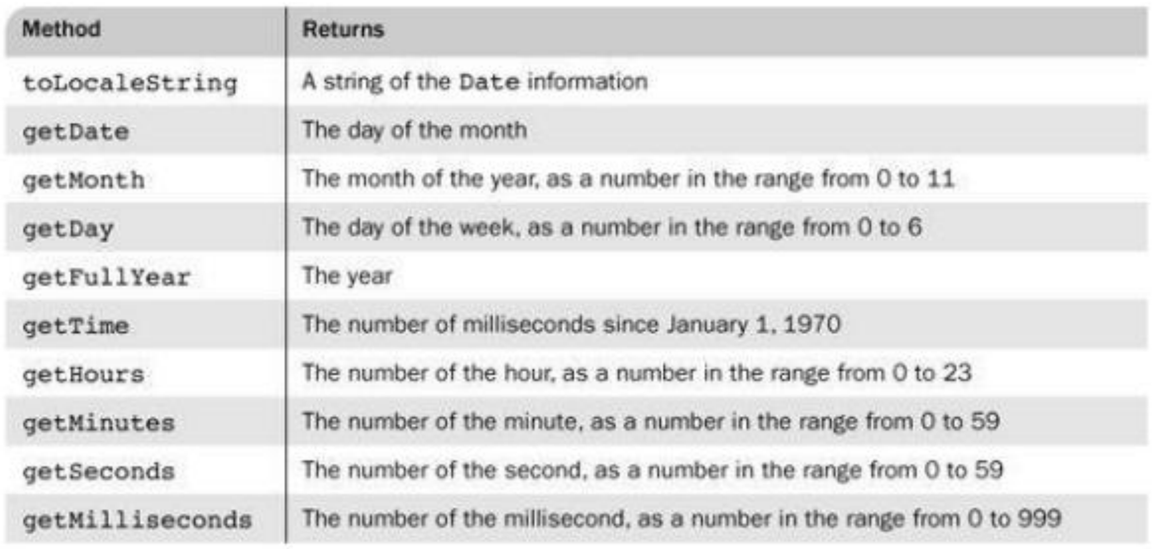
For example, the statement **a += 7;** means the same as **a = a + 7.**

Date Object

it is convenient to be able to create objects that represent a specific date and time and then manipulate them. These capabilities are available in

JavaScript through the Date object and its rich collection of methods. A Date object is created with the new operator and the Date constructor.

Table 4.5 Methods for the `Date` object



Method	Returns
`toLocaleString`	A string of the `Date` information
`getDate`	The day of the month
`getMonth`	The month of the year, as a number in the range from 0 to 11
`getDay`	The day of the week, as a number in the range from 0 to 6
`getFullYear`	The year
`getTime`	The number of milliseconds since January 1, 1970
`getHours`	The number of the hour, as a number in the range from 0 to 23
`getMinutes`	The number of the minute, as a number in the range from 0 to 59
`getSeconds`	The number of the second, as a number in the range from 0 to 59
`getMilliseconds`	The number of the millisecond, as a number in the range from 0 to 999

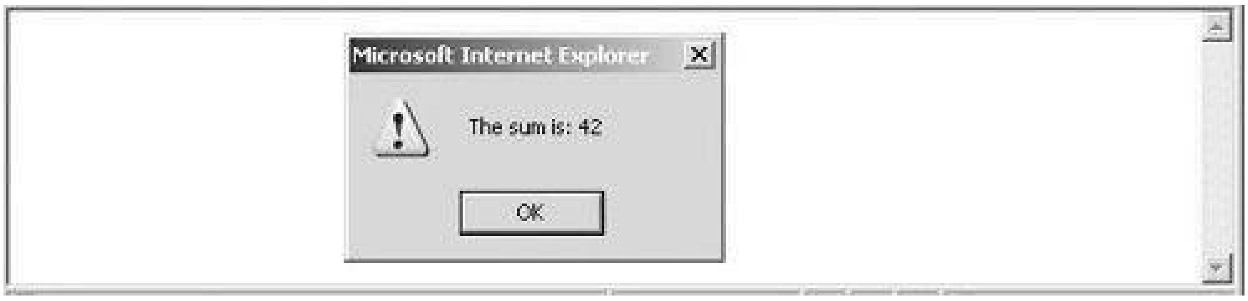
Screen Output and Keyboard Input

JavaScript models the XHTML document with the Document object. The window in which the browser displays an XHTML document is modeled with the Window object. The Window object includes two properties, **document and window.** The **document property** refers to the Document object. The **window property** is self-referential; it refers to the Window object. The document object has several methods, one of them is **Write.** Write methods is used to display or print the content specified in the parameter.

The Window object is the JavaScript model for the browser window. Window includes three methods that create dialog boxes for three specific kinds of user interactions. The three methods are alert, confirm, and prompt.

The **alert method** opens a dialog window and displays its parameter in that window. It also displays an OK button. The string parameter of alert is not XHTML code; it is plain text.

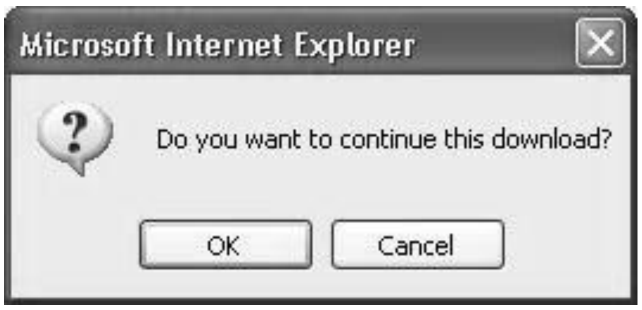
For Eg: **alert(“The sum is:” + sum + “\n”);** Output is



The **confirm method** opens a dialog window in which the method displays its string parameter, along with two buttons: OK and Cancel. confirm returns a Boolean value that indicates the user’s button input: true for OK and false for Cancel. This method is often used to offer the user the choice of continuing some process.

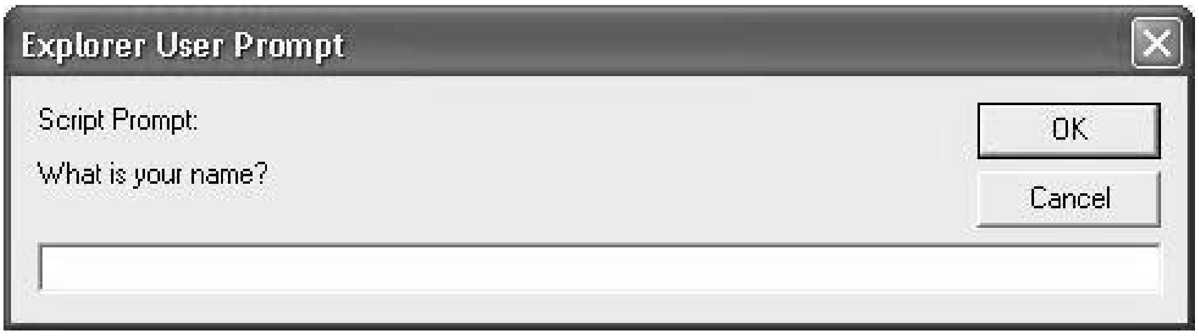
For Eg: **var question = confirm(“Do you want to continue this download?”);**

Output is:


Microsoft Internet Explorer
Do you want to continue this download?
OK
Cancel

The **prompt method** creates a dialog window that contains a text box used to collect a string of input from the user, which prompt returns as its value. As with confirm, this window also includes two buttons: OK and Cancel. prompt takes two parameters: the string that prompts the user for input and a default string in case the user does not type a string before pressing one of the two buttons.

For Eg: **name = prompt(“What is your name?”, ““);**



Script Prompt:
What is your name?

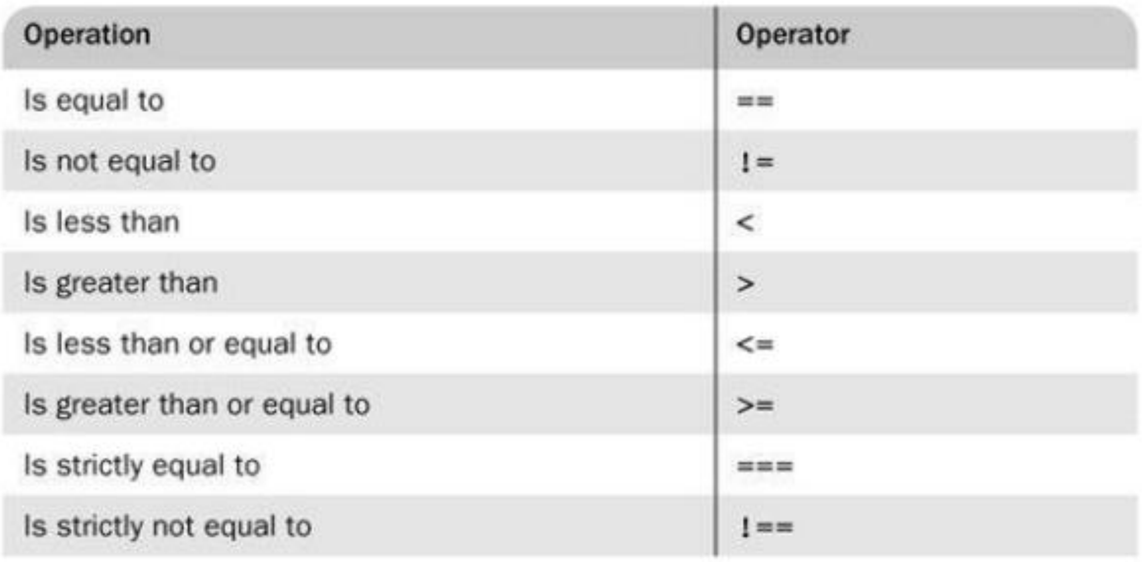
Control Statements

Control statements often require some syntactic container for sequences of statements whose execution they are meant to control. In JavaScript, that container is the compound statement. A compound statement in JavaScript is a sequence of statements delimited by braces. A **control construct** is a control statement together with the statement or compound statement whose execution it controls.

Control Expressions

The expressions upon which statement flow control can be based include primitive values, relational expressions, and compound expressions. The result of evaluating a control expression is one of the Boolean values true and false.

Table 4.6 Relational operators



Operation	Operator
Is equal to	==
Is not equal to	!=
Is less than	<
Is greater than	>
Is less than or equal to	<=
Is greater than or equal to	>=
Is strictly equal to	===
Is strictly not equal to	!==

If the two operands are not of the same type and the operator is neither === nor !==, JavaScript will attempt to convert the operands to a single type. Thus, the expression “3” === 3 evaluates to false, while “3” == 3 evaluates to true.

Selection Statements

The selection statements (if-then and if-then-else) of JavaScript are similar to those of the common programming languages. Either single statements or compound statements can be selected. Based on the condition or expression, if – else statement works. If the condition is true, if part will work and otherwise else part will be executed. The basic syntax is:

```
        If(expression or condition)
{
        Statements1;
} else
{
        Statements2;
}
```

Switch Statement

JavaScript has a switch statement that is similar to that of Java or c. The basic syntax of switch statement is: In any case segment, the statement(s) can be either a sequence of statements or a compound statement. The expression is evaluated when the switch statement is reached in execution. The value is compared with the values in the cases (those values that immediately follow the case reserved words). If one matches, control is transferred to the statement immediately following that case value. A break statement appears as the last statement in each sequence of statements following a case. Break statement is used for exiting from the loop after the execution. The control expression of a switch statement could evaluate to a number, a string, or a Boolean value. Case labels also can be numbers, strings, or Booleans, and different case values can be of different types.

Object Creation and Modification

Objects are often created with a **new** expression, which must include a call to a **constructor method**. The constructor that is called in the new expression creates the properties that characterize the new object. In an object-oriented language such as Java, the new operator creates a particular object. The following statement creates an object that has no properties:
var my_object = new Object();

The properties of an object can be accessed with dot notation, in which the first word is the object name and the second is the property name. Properties are not actually variables. A property for an object is created by assigning a value to that property's name. There is another method to create an object, i.e ,
var my_car = {make: “Ford”, model: “Contour SVT”};

Arrays

In JavaScript, arrays are objects that have some special functionality. Array elements can be primitive values or references to other objects, including other arrays. JavaScript arrays have dynamic lengths.

Array Object Creation

Array objects, unlike most other JavaScript objects, can be created in two distinct ways. The usual way to create any object is with the new operator and a call to a constructor. In the case of arrays, the constructor is named Array. The basic syntax is:

Var list = new Array (1, 2, “Hai”, “Hello”)

.

The second way to create an Array object is with a literal array value, which is a list of values enclosed in brackets:

Var list = { 1, 2, “Hai” , “Hello” }

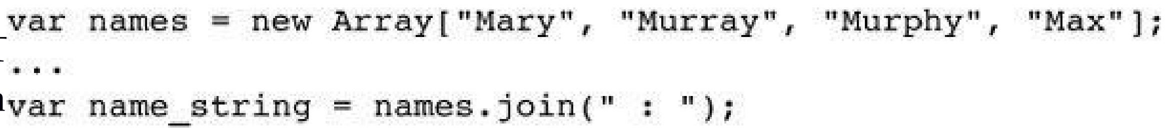
Characteristics of Array Objects

The lowest index of every JavaScript array is zero. Access to the elements of an array is specified with numeric subscript expressions placed in brackets. The length of an array is the highest subscript to which a value has been assigned, plus 1. The length of an array is both read and write accessible through the length property, which is created for every array object by the Array constructor. Consequently, the length of an array can be set to whatever you like by assigning the length property. To support JavaScript’s dynamic arrays, all array elements are allocated dynamically from the heap.

Array methods

Array objects have a collection of useful methods.

The join method: converts all of the elements of an array to strings and catenates them into a single string. If no parameter is provided to join, the values in the new string are separated by commas. If a string parameter is provided, it is used as the element separator.



```
var names = new Array["Mary", "Murray", "Murphy", "Max"];
...
var name_string = names.join(" : ");
```

The value of name_string is now “Mary : Murray : Murphy : Max”.

The sort method: coerces the elements of the array to become strings if they are not already strings and sorts them alphabetically.

names. Sort();

The value of names is now [“Mary”, “Max”, “Murphy”, “Murray”]

The concat method: catenates its actual parameters to the end of the Array object on which it is called.

```
var names = new Array["Mary", "Murray", "Murphy", "Max"];
...
var new_names = names.concat("Moo", "Meow");
```

The new_names array now has length 6, with the elements of names, along with “Moo” and “Meow”, as its fifth and sixth elements.

The slice method: does for arrays what the substring method does for strings, returning the part of the Array object specified by its parameters, which are used as subscripts.

```
var list = [2, 4, 6, 8, 10];
...
var list2 = list.slice(1, 3);
```

The value of list2 is now [4, 6].

When the **toString method**: is called through an Array object, each of the elements of the object is converted to a string. These strings are catenated, separated by commas.   **The pop and push methods**: respectively remove and add an element to the high end of an array. **The shift and unshift methods**: respectively remove and add an element to the beginning of an array.

Functions

A **function definition** consists of the function’s header and a compound statement that describes the actions of the function. This compound statement is called the body of the function. A function header consists of the reserved word function, the function’s name, and a parenthesized list of parameters if there are any. The parentheses are required even if there are no parameters. A return statement returns control from the function in which it appears to the function’s caller. Optionally, it includes an expression, whose value is returned to the caller.

A **function bod**y may include one or more return statements. If there are no return statements in a function or if the specific return that is executed does not include an expression, the value returned is undefined. A call to a function with no parameters states the function’s name followed by an empty pair of parentheses. A call to a function that returns undefined is a standalone statement.

A **call to a function** that returns a useful value appears as an operand in an expression. JavaScript functions are objects, so variables that reference them can be treated as are other object references and they can be passed as parameters, be assigned to other variables, and be the elements of an array.

The **scope of a variable** is the range of statements over which it is visible. When JavaScript is embedded in an XHTML document, the scope of a variable is the range of lines of the document over which the variable is visible. A variable that is not declared with a var statement is implicitly declared by the JavaScript interpreter at the time it is first encountered in the script. Variables that are implicitly declared have global scope; that is, they are visible in the entire XHTML document. It is usually best for variables that are used only within a function to have local scope, meaning that they are visible and can be used only within the body of the function. Any variable explicitly declared with var in the body of a function has local scope.

The parameter values that appear in a call to a function are called **actual parameters**. The parameter names that appear in the header of a function definition, which correspond to the actual parameters in calls to the function, are called **formal parameters**. JavaScript uses the pass-by-value parameter-passing method. When a function is called, the values of the actual parameters specified in the call are, in effect, copied into their corresponding formal parameters, which behave exactly like local variables. They also uses pass by reference method. All parameters are communicated through a **property array, arguments**, Using **arguments.length**, a function can determine the number of actual parameters that were passed.

Web Technology

Module – V

Introduction to Data Interchange Formats

(Ref: Robert W Sebesta, Programming the World Wide Web, 7/e)

Introduction

A meta-markup language is a language for defining markup languages. The Standard Generalized Markup Language (SGML) is a meta-markup language for defining markup languages that can describe a wide variety of document types. In 1986, SGML was approved as an International Standards Organization (ISO) standard. In 1996, the World Wide Web Consortium (W3C) began work on XML, another meta-markup language. The first XML standard, 1.0, was published in February 1998. The second, 1.1, was published in 2004. The motivation for the development of XML was the deficiencies of HTML.

The purpose of HTML is to describe the layout of information in Web documents. The problems are: HTML defines a collection of tags and attributes. One problem with HTML is that it was defined to describe the layout of information without considering the meaning of that information. Another potential problem with HTML is that it enforces few restrictions on the arrangement or order of tags in a document.

XML is a meta- markup language that provides a framework for defining specialized markup languages. HTML itself can be defined as an XML markup language. In fact, XHTML is an XML-based version of HTML.

It provides a simple and universal way of storing any textual data. Data stored in XML documents can be electronically distributed and processed by any number of different applications. These applications are relatively easy to write because of the standard way in which the data is stored. Therefore, XML is a universal data interchange language. an XML tag and its content, together with the closing tag, are called an **element**. Markup language designed with XML is called an **XML application**.

XML Syntax

Syntax of XML can be defined at two levels. First, there is the general low-level syntax of XML, which imposes its rules on all XML documents. The other syntactic level is specified by either **document type definitions (DTD)** or **XML schema.** DTD and XML schema specify the set of tags and attributes that can appear in a particular document or collection of documents and also the orders and arrangements in which they can appear.

An XML document can include several different kinds of statements. The most common of these statements are the data elements of the document. XML documents may also include markup declarations, which are instructions to the XML parser, and processing instructions, which are instructions for an application program that will process the data described in the document. All XML documents begin with an XML declaration. The XML declaration identifies the document as XML and provides the version number of the XML standard used. It may also specify an encoding standard. The XML declaration appears as the first line of all XHTML documents.

XML names are used to name elements and attributes. An XML name must begin with a letter or an underscore and can include digits, hyphens, and periods. XML names are case sensitive, so Body, body, and BODY are all distinct names. There is no length limitation for XML names. A small set of syntax rules applies to all XML documents. Every XML document defines a single root element, whose opening tag must appear on the first line of XML code. All other elements of an XML document must be nested inside the root element. Every XML element that can have content must have a closing tag. Elements that do not include content must use a tag with the following form:

<element_name />

XML tags can have attributes, which are specified with name–value assignments. As with XHTML, all attribute values must been closed by either single or double quotation marks. An XML document that strictly adheres to these syntax rules is considered well formed.

Commenting in XML

Comments in XML are the same as in HTML. They cannot contain two adjacent dashes.

```
<!-- A tag with one nested tag -->
```

Example for a XML program:

```
<?xml version = "1.0" encoding = "utf-8"?>
<ad>
  <year> 1960 </year>
  <make> Cessna </make>
  <model> Centurian </model>
  <color> Yellow with white trim </color>
  <location>
    <city> Gulfport </city>
    <state> Mississippi </state>
  </location>
</ad>
```

XML Document Structure

An XML document often uses two auxiliary files: one that defines its tag set and structural syntactic rules and one that contains a style sheet to describe how the content of the document is to be printed or displayed. The structural syntactic rules are given as either a DTD or an XML schema. An XML document consists of one or more entities, which are logically related collections of information. One of these entities is called the **document entity**, is always physically in the file that represents the document. The document entity can be the entire document.

Document is broken into several multiple entities for making it more manageable. Many documents include information that cannot be represented as text, such as images. Such information units are usually stored as binary data. If a binary data unit is logically part of a document, it must be a separate entity because XML documents cannot include binary data. These entities are called **binary entities.**

Entity names can be any length. They must begin with a letter, a dash, or a colon. After the first character, a name can have letters, digits, periods, dashes, underscores, or colons. A reference to an entity is its name together with a ampersand and an appended semicolon. For example, if apple_image is the name of an entity, &apple_image; is a reference to it.

Character Data

The form of a character data section is

<![CDATA[content]]>

The opening keyword of a character data is [CDATA[. An important consequence of this rule is that there cannot be any spaces between the [and the C or between the A (the last character of CDATA) and the second [. The CDATA section has the closing delimiter,]]>.

Document Type Definitions

A document type definition (DTD) is a set of structural rules called declarations, which specify a set of elements and attributes that can appear in a document and how and where these elements and attributes may appear. DTD also provide entity definitions. DTD is similar to external style sheets. DTD are used when the same tag set definition is used by a collection of documents.

A DTD can be embedded in the XML document whose syntax rules it describes, in which case it is called an **internal DTD**. The alternative is to have the DTD stored in a separate file, in which case it is called an **external DTD.** Because external DTD allow use with more than one XML document, they are preferable. Syntactically, a DTD is a sequence of declarations, each of which has the form of a markup declaration:

<!keyword ... >

Four possible keywords can be used in a declaration: **ELEMENT**, used to define tags; **ATTLIST**, used to define tag attributes; **ENTITTY**, used to define entities; and **NOTATION**, used to define data type notations. Notation is not frequently used.

Declaring Elements

The element declarations of a DTD have a form that is related to that of the rules of context-free grammars, also known as Backus–Naur form (BNF).BNF is used to define the syntactic structure of programming languages. Each element declaration in a DTD specifies the structure of one category of elements. The declaration provides the name of the element whose structure is being defined, along with the specification of the structure of that element.

A Tree structure is used to represent elements. n element is a node in such a tree, either a leaf node or an internal node. If the element is a leaf node, its syntactic description is its character. pattern. If the element is an internal node, its syntactic description is a list of its child elements, each of which can be a leaf node or an internal node.

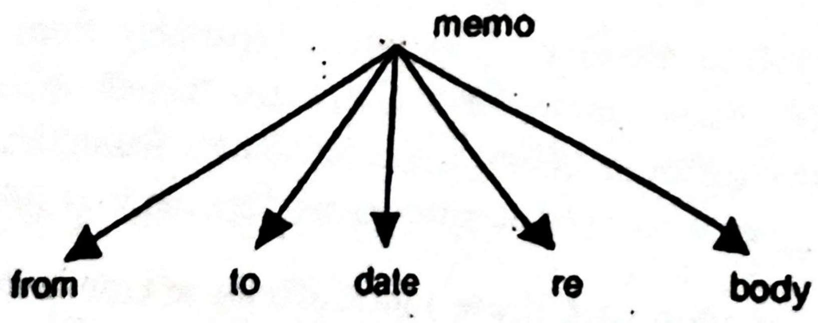
The form of an element declaration for elements that contain elements is as follows:

<!ELEMENT element_name (list of names of child elements)>

For example,

<!ELEMENT memo (from, to, date, re, body)>

This element declaration would describe the document tree structure as:



- **Modifiers** are used to specify how many number of times a child element occurs.

Modifier	Meaning
+	One or more occurrences
*	Zero or more occurrences
?	Zero or One occurrence

The leaf nodes of a DTD specify the data types of the content of their parent nodes, which are elements. In most cases, the content of an element is type **PCDATA**, for **parsable character data.** Parsable character data is a string of any printable characters except "less than" (<), "greater than" (>), and the ampersand (&).

Two other content types can be specified: **EMPTY** and **ANY**. The EMPTY type specifies that the element has no content; The ANY type is used when the element may contain literally any content. The form of a leaf element declaration is as follows:

<!ELEMENT element_name (#PCDATA)>

Declaring Attributes

The attributes of an element are declared separately from the element declaration in a DTD. An attribute declaration must include the name of the element to which the attribute belongs, the attribute's name, its type, and a default option. The general form of an attribute declaration is as follows:

<!ATTLIST element_name attribute_name attribute type default_option>

If more than one attribute is declared for a given element, the declarations can be combined.

```
<!ATTLIST element_name
          attribute_name_1 attribute_type default_option_1
          attribute_name_2 attribute_type default_option_2
          ...
          attribute_name_n attribute_type default_option_n
>
```

There are 10 different attribute types. Only CDATA is used. The default option in an attribute declaration can specify either an actual value or a requirement for the value of the attribute in the XML document.

For example,

```
<!ATTLIST airplane places CDATA "4">
<!ATTLIST airplane engine_type CDATA #REQUIRED>
<!ATTLIST airplane price CDATA #IMPLIED>
<!ATTLIST airplane manufacturer CDATA #FIXED "Cessna">
```

Declaring Entities

Entities can be defined so that they can be referenced anywhere in the content of an XML document, in which case they are called **general entities**. The predefined entities are all general entities. Entities can also be defined so that they can be referenced only in DTD, in which case they are called **parameter entities.**

The form of an entity declaration is

<!ENTITY [%] entity_name "entity_value">

When the optional percent sign (%) is present in an entity declaration, it specifies that the entity is a parameter entity rather than a general entity. When an entity is longer than a few words, such as a section of a technical article, its text is defined outside the DTD. In such cases, the entity is called an **external text entity**. The form of the declaration of an external text entity is

<!ENTITY entity_name SYSTEM "file_location">

The keyword SYSTEM specifies that the definition of the entity is in a different file, which is specified as the string following SYSTEM.

Some XML parsers check documents that have DTDs in order to ensure that the documents conform to the structure specified in the DTD. These parsers are called **validating parsers.** If an XML document specifies a DTD and is parsed by a validating XML parser, and the parser determines that the document conforms to the DTD, the document is called valid.

Internal and External DTD

If the DTD is included in the XML code, it must be introduced with <!DOCTYPE root_name [and terminated with]>. For example:

```
<?xml version = "1.0" encoding = "utf-8"?>
      <!DOCTYPE planes [
            <!-- The DTD for planes -->
      ]>
<!-- The planes XML document -->
```

When the DTD is in a separate file, the XML document refers to it with a DOCTYPE declaration as its second line. This declaration then has the following form:

<!DOCTYPE XML_document_root_name SYSTEM "DTD_file_name">

For Eg:

<!DOCTYPE planes_for_sale SYSTEM "planes.dtd">

Namespaces

An XML namespace is a collection of element and attribute names used in XML documents. The name of a namespace usually has the form of a uniform resource identifier (URI). 2 A namespace for the elements and attributes of the hierarchy rooted at a particular element is declared as the value of the attribute xmlns.

The form of a namespace declaration for an element is

<element_name xmlns[:prefix] = URI>

The square brackets indicate that what is within them is optional. The prefix, if included, is the name that must be attached to the names in the declared namespace. If the prefix is not included, the namespace is the default for the document.

A prefix is used for two reasons. First, most URI are too long to be typed on every occurrence of every name from the namespace. Second, a URI includes characters that are invalid in XML. For Eg:

<birds xmlns:bd = " http://www.audubon.org/names/species">

bd is the prefix for bird.

Attribute names are not included in namespaces because attribute names are local to elements, so a tag set may use the same attribute name in more than one element without causing ambiguity.

XML Schema

DTD have several disadvantages : One is that DTDs are written in a syntax unrelated to XML, so they cannot be analyzed with an XML processor. Also, it can be confusing for people to deal with two different syntactic forms, one that defines a document and one that defines its structure. Another disadvantage is that DTD do not allow restrictions on the form of data that can be the content of a particular tag.

An XML schema is an XML document, so it can be parsed with an XML parser. It also provides far more control over data types than do DTD. The content of a specific element can be required to be any one of 44 different data types.

A schema is similar to a class definition; an XML document that conforms to the structure defined in the schema is similar to an object of the schema's class. Schemas have two primary purposes. First, a schema specifies the structure of its instance XML documents, including which elements and attributes may appear in the instance document, as well as where and how often they may appear. Second, a schema specifies the data type of every element and attribute in its instance XML documents.

Defining a Schema

Every schema has schema as its root element. A schema defines a namespace in the same sense as a DTD defines a tag set. The name of the namespace defined by a schema must be specified with the targetNamespace attribute of the schema element. The name of every top-level (not nested) element that appears in a schema is placed in the target namespace, which is specified by assigning a namespace to the target namespace attribute:

targetNamespace = "http://cs.uccs.edu/planeSchema"

If the elements and attributes that are not defined directly in the schema element are to be included in the target namespace, schema's elementFormDefault must be set to qualified, as follows:

elementFormDefault = "qualified"

- **Defining a Schema Instance**

An instance of a schema must include specifications of the namespaces it uses. These specifications are given as attribute assignments in the tag for the root element of the schema.

The second attribute specification in the root element of an instance document is for the schemaLocation attribute. This attribute is used to name the standard namespace for instances, which includes the name XMLSchema-instance. This namespace corresponds to the XMLSchema namespace used for schema.

The following attribute assignment specifies the XMLSchema-instance namespace and defines the prefix, xsi, for it:

xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"

The instance document must specify the filename of the schema in which the default namespace is defined. This is accomplished with the **schemaLocation** attribute, which takes two values: the namespace of the schema and the filename of the schema. This attribute is defined in the XMLSchema-instance namespace.

```
xsi:schemaLocation = "http://cs.uccs.edu/planeSchema
                      planes.xsd"
```

Data types in XML

time .URI

There are two categories of user-defined schema data types: simple and complex. A simple data type is a data type whose content is restricted to strings. A simple type cannot have attributes or include nested elements. A complex type can have attributes and include other data types as child elements.

The XML Schema defines 44 data types, 19 of which are primitive and 25 of which are derived. The **primitive data types** include string, Boolean, float, time, and anyURI. The **predefined derived types** include byte, long, decimal, unsignedInt, positiveInteger, and NMTOKEN.

User-defined data types are defined by specifying restrictions on an existing type, which is then called a **base type**. Such user-defined types are **derived types**. Both simple and complex types can be named or anonymous. If anonymous, a type cannot be used outside the element in which it is declared.

Data declarations in an XML schema can be either **local or global**. A **local declaration** is a declaration that appears inside an element that is a child of the schema element; that is, a declaration in a grandchild element of schema (or in a more distant descendant) is a local declaration. A **global declaration** is a declaration that appears as a child of the schema element. Global elements are visible in the whole schema in which they are declared.

→ inside child

schem / global / local.

Simple Types

Elements are defined in an XML schema with the element tag, which is from the XMLSchema namespace. An element that is named includes the **name attribute** for that purpose. The other attribute that is necessary in a simple element declaration is **type**, which is used to specify the type of content allowed in the element. For Eg:

<xsd:element name = "engine" type = "xsd:string" />

An element can be given a default value with the **default attribute**. For Eg:

```
<xsd:element name = "engine"    type = "xsd:string"
                 default = "fuel injected V-6"   />
```

Elements can have constant values, meaning that the content of the defined element in every instance document has the same value. Constant values are given with the fixed attribute. For Eg:

```
<xsd:element name = "plane"    type = "xsd:string"
             fixed = "single wing"   />
```

A simple user-defined data type is described in a **simpleType** element with the use of facets.

Facets must be specified in the content of a **restriction element,** which gives the base type name. The facets themselves are given in elements named for the facets: the **value attribute** specifies the value of the facet. For eg:

```
<xsd:simpleType name = "firstName">
  <xsd:restriction base = "xsd:string">
    <xsd:maxLength value = "10" />
  </xsd:restriction>
</xsd:simpleType>
```

The **length facet** is used to restrict the string to an exact number of characters. The **minLength facet** is used to specify a minimum length. The number of digits of a decimal number can be restricted with the **precision facet.**

Complex Types

Complex types are defined with the **complexType** tag. The elements that are the content of an element-only element must be contained in an **ordered group, an unordered group, a choice, or a named group.**

The **sequence element** is used to contain an **ordered group** of elements, as in the following type definition:

```
<xsd:complexType name = "sports_car">
  <xsd:sequence> → ordered group
    <xsd:element name = "make"  type = "xsd:string" />
    <xsd:element name = "model" type = "xsd:string" />
    <xsd:element name = "engine" type = "xsd:string" />
    <xsd:element name = "year"  type = "xsd:decimal" />
  </xsd:sequence>
</xsd:complexType>
```

A complex type whose elements are an **unordered group** is defined in an **all element**. Elements in **all and sequence groups** can include the **minOccurs** and **maxOccurs** attributes to specify the numbers of occurrences.

Displaying Raw XML Documents

An XML-enabled browser— or any other system that can deal with XML documents—cannot know how to format the tags defined in any given document. Contemporary browsers include default style sheets that are used when no style sheet is specified in the XML document.

Displaying XML Documents with CSS

Style-sheet information can be provided to the browser for an XML document in two ways. First, a **Cascading Style Sheet (CSS) file** that has style information for the elements in the XML document can be developed. Second, the **XSLT style-sheet technology**, which was developed by the W3C can be used.

The form of a CSS style sheet for an XML document is simple: It is just a list of element names, each followed by a brace-delimited set of the element's CSS attributes. This is the form of the rules in a CSS document style sheet. For Eg:

```
<!-- planes.css - a style sheet for the planes.xml document -->
ad { display: block; margin-top: 15px; color: blue;}
year, make, model { color: red; font-size: 16pt;}
color {display: block; margin-left: 20px; font-size: 12pt;}
description {display: block; margin-left: 20px; font-size: 12pt;
seller { display: block; margin-left: 15px; font-size: 14pt;}
location {display: block; margin-left: 40px; }
city {font-size: 12pt;}
state {font-size: 12pt;}
```

The only **style property** in this style sheet that has not been discussed earlier is **display**, which is used to specify whether an element is to be displayed inline or in a separate block. These two options are specified with the values **inline and block**. The inline value is the default. When display is set to **block**, the content of the element is usually separated from its sibling elements by line breaks.

The connection of an **XML document** to a **CSS style sheet** is established with the **processing instruction xml-stylesheet**, which specifies the particular type of the style sheet via its **type attribute** and the name of the file that stores the style sheet via its **href attribute**. For Example:

<?xml-stylesheet type = "text/css" href = "planes.css" ?>

XSLT Style Sheets

The eXtensible Stylesheet Language (XSL) is a family of recommendations for defining the presentation and transformations of XML documents. It consists of three related standards: **XSL Transformations (XSLT), XML Path Language (XPath)**, and **XSL Formatting Objects (XSL-FO)**.

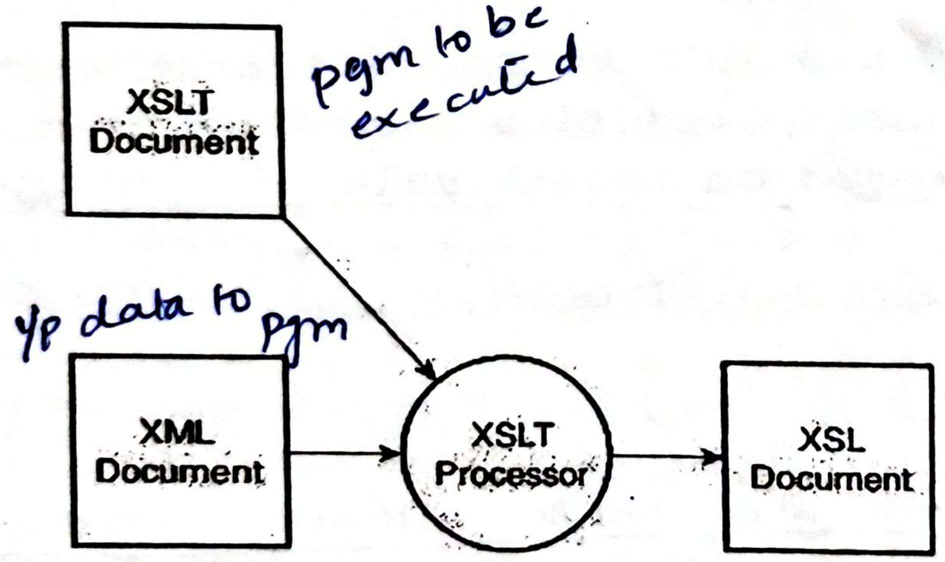
XSLT style sheets are used to transform XML documents into different forms or formats, perhaps using different DTDs. One common use for XSLT is to transform XML documents into XHTML documents, primarily for display.

XPath is a language for expressions, which are often used to identify parts of XML documents, such as specific elements that are in specific positions in the document or elements that have particular attribute values. XSLT requires such expressions to specify transformations. XPath is also used for XML document querying languages, such as XQL, and to build new XML document structures with XPointer.

Overview of XSLT

XSLT is actually a simple functional-style programming language.. Included in XSLT are functions, parameters, names to which values can be bound, selection constructs, and conditional expressions for multiple selection.

XSLT processors take both an XML document and an XSLT document as input. The XSLT document is the program to be executed; the XML document is the input data to the program. Parts of the XML document are selected, possibly modified, and merged with parts of the XSLT document to form a new document, which is sometimes called an XSL document. The transformation process can be represented as:



An XSLT document consists primarily of one or more templates, which use XPath to describe element–attribute patterns in the input XML document.

One XSLT model of processing XML data is called the template-driven model, which works well when the data consists of multiple instances of highly regular data collections. XSLT can also deal with irregular and recursive data, using template fragments in what is called the data-driven model. A single XSLT style sheet can include the mechanisms for both the template- and data-driven models.

XSL Transformations for Presentation

·XSLT style sheets can be used to control page layout, including orientation, writing direction, margins, and page numbering.

An XSLT style sheet is an XML document whose root element is the special-purpose element **stylesheet**. The stylesheet tag defines namespaces as its attributes and encloses the collection of elements that defines its transformations. It also identifies the document as an XSLT document. The namespace for all XSLT elements is specified with a W3C URL. For Eg;

```
<?xml-stylesheet type = "text/xsl" href =
                                "XSL_stylesheet_name" ?>
```

A style-sheet document must include at least one **template element**. The template opening tag includes a **match attribute** to specify an XPath expression that selects a node in the XML document. The content of a template element specifies what is to be placed in the output document.

To produce complete XHTML documents as output from XSLT documents, the **output element** can be included before the first template. This element can include **doctype-public and doctype-system** attributes to specify the two parts of the DOCTYPE declaration, respectively. XPath expressions that **begin with the slash are absolute addresses** within the document. Those that **do not begin with a slash** are **relative addresses.**

The **apply-templates** element applies appropriate templates to the descendant nodes of the current node. This element can include a **select attribute** to specify the descendant nodes whose templates should be applied. If no select attribute is included, the XSLT processor will apply a template to every descendant node.

The content of an element of the XML document is to be copied to the output document. This is done with the **value-of element**, which uses a **select attribute** to specify the element of the XML document whose contents are to be copied. The **attribute value "."** for the select attribute of value-of denotes the selection of all elements within the current element.

24 [illegible]

Web Technology

Module – VI

Introduction to PHP

(Ref: Robert W Sebesta, Programming the World Wide Web, 7/e)

Origins and Uses of PHP

PHP was developed by **Rasmus Lerdorf,** a member of the Apache Group in 1994. Its initial purpose was to provide a tool to help Lerdorf track visitors to his personal Web site. In 1995 he developed a package called **Personal Home Page Tools,** which became the first publicly distributed version of PHP. Originally, PHPwas an acronym for **Personal Home Page.** Later it was renamed as **Hypertext preprocessor.** PHP was being used at a large number of Web sites. As a server-side scripting language, PHP is naturally used for form handling and database access. PHP supports the common electronic mail protocols POP3 and IMAP. It also supports the distributed object architectures COM and CORBA.

Overview of PHP

PHP is a server-side HTML- or XHTML-embedded scripting language. When a browser requests a document that includes PHP script, the Web server that provides the document calls its PHP processor. The server determines that a document includes PHP script by the **file-name extension**. If it is **.php, .php3, or .phtml.** The PHP processor has two modes of operation: **copy mode and interpret mode.** The syntax and semantics of PHP are closely related to the syntax and semantics of JavaScript. PHP uses dynamic typing: The type of a variable is set every time it is assigned a value. PHP's arrays are a merge of the arrays of common programming languages and associative arrays, having the characteristics of both.

PHP supports both **procedural and object-oriented programming.**

General Syntactic Characteristics

PHP scripts either are embedded in HTML or XHTML documents or are in files that are referenced by such documents. PHP code is embedded in documents by enclosing it between the <?php and ?> tags.

If a PHP script is stored in a different file, it can be brought into a document with the include construct, which takes the filename as its parameter— for example,

include("table2.inc");

All variable names in PHP begin with a dollar sign ($). The part of the name after the dollar sign is like the names of variables in many common programming languages: a letter or an underscore followed by any number (including zero) of letters, digits, or underscores. PHP variable names are case sensitive. PHP has a list of reserved words, but they are not case sensitive. List of reserved words:

`and`	`else`	`global`	`require`	`virtual`
`break`	`elseif`	`if`	`return`	`xor`
`case`	`extends`	`include`	`static`	`while`
`class`	`false`	`list`	`switch`	
`continue`	`for`	`new`	`this`	
`default`	`foreach`	`not`	`true`	
`do`	`function`	`or`	`var`	

PHP allows comments to be specified in three different ways. Single-line comments can be specified either with # or with //. Multiple-line comments are delimited with /* and */.

PHP statements are terminated with semicolons. Braces are used to form compound statements for control structures.

Primitives, Operations, and Expressions

PHP has **four scalar types** - Boolean, integer, double, and string; **two compound types** - array and object; and **two special types** - resource and NULL.

Variables

PHP is dynamically typed, it has no type declarations. The type of a variable is set every time the variable is assigned a value. An unassigned variable, sometimes called an **unbound variable,** has the value NULL, which is the only value of the NULL type. If the context specifies a number, NULL is coerced to 0; if the context specifies a string, NULL is coerced to the empty string. A variable can be tested to determine whether it currently has a value. The test is carried out with the **IsSet function,** which takes the variable's name as its parameter and returns a Boolean value.

A variable that has been assigned a value retains that value until either it is assigned a new value or it is set back to the unassigned state, which is done with the **unset function**. when an unbound variable is referenced, include a call to the **error_reporting function** to change the error-reporting level of the PHP interpreter to 15. The **default error-reporting level is** 7, which does not require the interpreter to report the use of an unbound variable.

Data types

Mainly four scalar data types.

Integer Type

PHP has a single integer type, named integer. This type corresponds to the long type of C.

Double Type

PHP's double type corresponds to the double type of C and its successors. Double literals can include a **decimal point, an exponent, or both.** The exponent has the usual form of an **E or an e,** followed by a possibly signed integer literal.

String Type

Characters in PHP are single bytes; UNICODE is not supported. There is no character type. A single character data value is represented as a string of length 1. String literals are defined with either single-quote (') or double-quote (") delimiters.

Boolean Type

The only two possible values for the Boolean type are TRUE and FALSE, both of which are case insensitive.

Arithmetic Operators and Expressions

PHP has the usual collection of arithmetic operators (+, -, *, /, %, ++, and - -). PHP has a large number of predefined functions that operate on numeric values.

Function	Parameter Type	Returns
`floor`	Double	Largest integer less than or equal to the parameter
`ceil`	Double	Smallest integer greater than or equal to the parameter
`round`	Double	Nearest Integer
`srand`	Integer	Initializes a random-number generator with the parameter
`rand`	Two numbers	A pseudorandom number greater than the first parameter and smaller than the second
`abs`	Number	Absolute value of the parameter
`min`	One or more numbers	Smallest
`max`	One or more numbers	Largest

String Operations

The only string operator is the catenation operator, specified with a period (.). PHP includes many functions that operate on strings. They are:

Function	Parameter Type	Returns
`strlen`	A string	The number of characters in the string
`strcmp`	Two strings	Zero if the two strings are identical, a negative number if the first string belongs before the second (in the ASCII sequence), or a positive number if the second string belongs before the first
`strpos`	Two strings	The character position in the first string of the first character of the second string if the second string is in the first string; `false` if it is not there
`substr`	A string and an integer	The substring of the string parameter, starting from the position indicated by the second parameter; if a third parameter (an integer) is given, it specifies the length of the returned substring
`chop`	A string	The parameter with all white-space characters removed from its end
`trim`	A string	The parameter with all white-space characters removed from both ends
`ltrim`	A string	The parameter with all white-space characters removed from its beginning
`strtolower`	A string	The parameter with all uppercase letters converted to lowercase
`strtoupper`	A string	The parameter with all lowercase letters converted to uppercase

Scalar Type Conversions

Includes both implicit and explicit type conversions. Implicit type conversions are called **coercions**. Explicit type conversions can be specified in three different ways. One method is **Casting:** The cast is a type name in parentheses preceding the expression. For Eg:

(int)$sum

Another way to specify explicit type conversion is to use one of the **functions intval, doubleval, or strval**. For example,

intval($sum)

The third way to specify an explicit type conversion is with the **settype function**, which takes two parameters: a variable and a string that specifies a type name. For example,

settype($sum, "integer");

The type of the value of a variable can be determined in two different ways. The first is the **gettype function,** which takes a variable as its parameter and returns a string that has the name of the type of the current value of the variable. One possible return value of gettype is "unknown".

The other way to determine the type of the value of a variable is to use one or more of the type-testing functions, each of which takes a variable name as a parameter and returns a Boolean value. These functions are **is_int, is_integer, and is_long,** which test for integer type; **is_double, is_float, and is_real,** which test for double type; is_bool, which tests for Boolean type; and **is_string,** which tests for string type.

Assignment Operators

PHP has the same set of assignment operators as its predecessor language, C, including the compound assignment operators such as += and /=.

Output

The **print function** is used to create simple unformatted output. It can be called with or without parentheses around its parameter. For example,

print "Apples are red <br /> Kumquats aren't <br />";

PHP borrows the **printf** function from C. It is used when control over the format of displayed data is required. The general form of a call to printf is as follows:

printf(literal_string, param1, param2, ...)

The form of the format codes is a percent sign (%) followed by a field width and a type specifier. The most common type specifiers are **s for strings, d for integers, and f for floats and doubles.**

%10s—a character string field of 10 characters

%6d—an integer field of six digits

%5.2f—a float or double field of eight spaces, with two digits to the right of the decimal point, the decimal point, and five digits to the left.

Relational Operators

PHP uses the **eight relational operators** of JavaScript. The usual six operators are (>, <, >=, <=, !=, and ==). PHP also has ===, which produces TRUE only if both operands are the same type and have the same value, and !==, the opposite of ===.

Boolean Operators

There are six Boolean operators: **and, or, xor, !, &&, and ||**. The and and && operators perform the same operation, as do or and ||. The difference between them is that the precedence of and and or is lower than that of && and ||.

Selection Statements

If Statements

The control expression can be an expression of any type, but its value is coerced to Boolean. The controlled statement segment can be either a single statement or a compound statement. For Eg:

```
if ($day == "Saturday" || $day == "Sunday")
  $today = "weekend";
else {
  $today = "weekday";
  $work = true;
}
```

The switch statement has the form and semantics of that of JavaScript. The type of the control expression and of the case expressions is either integer, double, or string. If necessary, the values of the case expressions are coerced to the type of the control expression for the comparisons. A default case can be included. For Eg:

```
switch ($bordersize) {
  case "0": print "<table>";
            break;
  case "1": print "<table border = '1'>";
            break;
  case "4": print "<table border = '4'>";
            break;
  case "8": print "<table border = '8'>";
            break;
  default: print "Error-invalid value: $bordersize <br />";
}
```

Loop Statements

The while, for, and do-while statements of PHP are exactly like those of JavaScript. PHP also has a foreach statement.

While loop

Similar to other programming languages. For Eg:

```
$fact = 1;
$count = 1;
while ($count < $n) {
  $count++;
  $fact *= $count;
}
```

do -while loop

Exit controlled loop similar to that in other programming languages. For Eg:

```
$count = 1;
$sum = 0;
do {
  $sum += $count;
  $count++;
} while ($count <= 100);
```

For Loop

Can be used for repaeating a statement many times. For Eg:

```
for ($count = 1, $fact = 1; $count < $n;) {
  $count++;
  $fact *= $count;
}
```

Mathematical Function

The **sqrt function** returns the square root of its parameter; the **pow function** raises its first parameter to the power of its second parameter.

Arrays

Each array element consists of two parts: **a key and a value**. If the array has a logical structure that is similar to a hash, its keys are strings and the order of its elements is determined with a system-designed hashing function.

Array Creation

There are two ways to create an array in PHP. The assignment operation creates scalar variables. The same operation works for arrays: Assigning a value to a subscripted variable that previously was not an array creates the array. For example,

$list[0] = 17;

The second way to create an array is with the **array construct**. The parameters of array specify the values to be placed in a new array and sometimes also the keys. If the array is like a traditional array, only the values need to be specified; the PHP interpreter will furnish the numeric keys. For example,

$list = array(17, 24, 45, 91);

An array construct with empty parentheses creates an empty array. For example, in the following statement, $list becomes a variable whose value is an array with no elements:

$list = array();

The following statement creates an array that has the form of a hash:

$ages = array("Joe" => 42, "Mary" => 41, "Bif" => 17);

List Construct

The list construct can be used to assign multiple elements of an array to scalar variables in one statement. For example,

```
$trees = array("oak", "pine", "binary");
list($hardwood, $softwood, $data_structure) = $trees;
```

`$hardwood`, `$softwood`, and `$data_structure` are set to `"oak"`, `"pine`
and `"binary"`, respectively.

Functions for Dealing with Arrays

Unset function

A whole array can be deleted with **unset**, as with a scalar variable. Individual elements of an array also can be removed with unset, as in the following code:

$list = array(2, 4, 6, 8);

unset($list[2]);

After executing these statements, $list has three remaining elements with keys 0, 1, and 3 and elements 2, 4, and 8.

The **array_keys function** takes an array as its parameter and returns an array of the keys of the given array. The **array_values function** is used to give the output as values in the array. The **existence of an element of a specific key** can be determined with the **array_key_exists function**, which returns a Boolean value.

The **is_array function** is similar to the is_int function: It takes a variable as its parameter and returns TRUE if the variable is an array, FALSE otherwise. The **in_array function** takes two parameters—an expression and an array—and returns TRUE if the value of the expression is a value in the array; otherwise, it returns FALSE.

The number of elements in an array can be determined with the **sizeof function.** PHP allows to convert between strings and arrays. These conversions can be done with the **implode and explode functions.**

The **explode function** explodes a string into substrings and returns them in an array. The delimiters of the substrings are defined by the first parameter of explode, which is a string; the second parameter is the string to be converted. For example,

$str = "April in Paris, Texas is nice";

$words = explode(" ", $str);

$words contains ("April", "in", "Paris,", "Texas", "is", "nice").

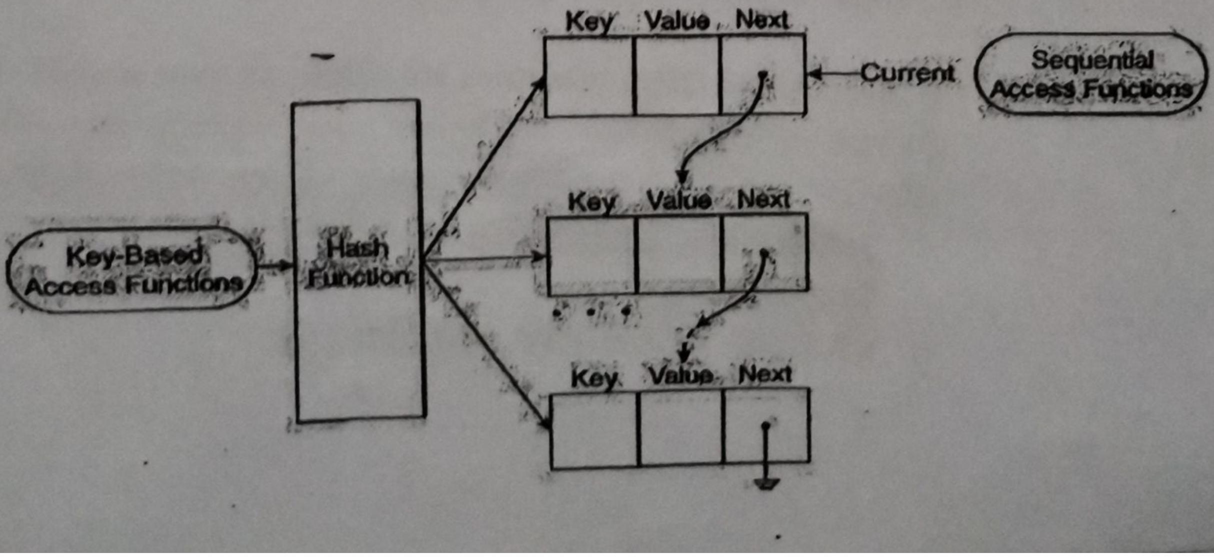
The **implode function** does the inverse of explode. Given a separator character (or a string) and an array, it catenates the elements of the array together,using the given separator string between the elements, and returns the result as a string. For Eg:

$words = array("Are", "you", "lonesome", "tonight");

$str = implode(" ", $words);

Now $str has "Are you lonesome tonight"

The elements of an array are stored in a linked list of cells, where each cell includes both the key and the value of the element. The cells themselves are stored in memory through a key-hashing function so that they are randomly distributed in a reserved block of storage. Accesses to elements through string keys are implemented through the hashing function. The logic internal structure of array can be represented as:



Sequential Access to Array Elements

PHP includes several different ways to access array elements in sequential order. Every array has an internal pointer that references one element of the array. We call this pointer as the **"current"** pointer which points to the current value. Usually first value will be current value.

The "current" pointer can be moved with the **next function**, which both moves the pointer to the next array element and returns the value of that element. The "current" pointer can be moved backward (i. e., to the element before the "current" element) with the **prev function.** The "current" pointer can be set to the first element with the **reset function,** which also returns the value of the first element. The **key function,** when given the name of an array, returns the key of the "current" element of the array.

The **array_push and array_pop functions** provide a simple way to implement a stack in an array. The **array_push function** takes an array as its first parameter. After this first parameter, there can be any number of additional parameters. The values of all subsequent parameters are placed at the end of the array.

The **array_pop function** takes a single parameter: the name of an array. It removes the last element from the array and returns it. The **foreach statement** is designed to build loops that process all of the elements of an array. This statement has two forms:

foreach (array as scalar_variable) loop body

foreach (array as key => value) loop body

Sorting Arrays

The **sort function,** which takes an array as a parameter, sorts the values in the array, replacing the keys with the numeric keys, 0, 1, 2, ... The array can have both string and numeric values. The string values migrate to the beginning of the array in alphabetical order.

The **asort function** is used to sort arrays that correspond to hashes. It sorts the elements of a given array by their values, but keeps the original key–value associations.

The **ksort function** sorts its given array by keys, rather than values.

Functions

The general form of a PHP function definition is as follows:

```
function name([parameters]) {
...
}
```

The square brackets around the parameters mean that the parameters are optional. The **return statement** is used in a function to specify the value to be returned to the caller. Function execution ends when a return statement is encountered or the last statement in the function has been executed.

Parameters

We call the parameters in the call to a function **actual parameters.** We call the parameters that are listed in the function definition **formal parameters.** An actual parameter can be any expression. A formal parameter must be a variable name.The default **parameter-passing mechanism** of PHP is **pass by value.** This means that, in effect, the values of actual parameters are copied into the memory locations associated with the corresponding formal parameters in the called function.

Pass-by-reference parameters can be specified in PHP in two ways. One way is to add an **ampersand (&) to the beginning of the name of the formal parameter** that you want to be passed by reference. The other way is to add an **ampersand to the actual parameter** in the function call.

Scope of Variables

The default scope of a variable defined in a function is local. A local variable is visible only in the function in which it is used. PHP has the global declaration. When a variable is listed in a global declaration in a function, that variable is expected to be defined outside the function. So, such a variable has the same meaning inside the function as outside. In PHP, a local variable in a function can be specified to be static by declaring it with the reserved word **static**.

Pattern Matching

PHP includes **two different kinds** of string pattern matching using regular expressions: one that is based on **POSIX regular expressions** and one that is based on **Perl regular expressions.**

Functions use are:

The **preg_match function** takes two parameters, the first of which is the Perl-style regular expression as a string. The second parameter is the string to be searched. For Eg:

```
if (preg_match("/^PHP/", $str))
  print "\$str begins with PHP <br />";
else
  print "\$str does not begin with PHP <br />";
```

The **preg_split function** operates on strings but returns an array and uses patterns. The function takes two parameters, the first of which is a Perl-style pattern as a string. The second parameter is the string to be split. For example,

$fruit_string = "apple : orange : banana";

$fruits = preg_split("/ : /", $fruit_string);

The array $fruits now has ("apple", "orange", "banana").

Form Handling

One common way for a browser user to interact with a Web server is through forms. A form is presented to the user, who is invited to fill in the text boxes and click the buttons of the form. The user submits the form to the server by clicking the form's Submit button. The contents of the form are encoded and transmitted to the server. The approach used is is to use the implicit arrays $_POST and $_GET for form values. These arrays have keys that match the form element names and values that were input by the client. For example, if a form has a text box named phone and the form method is POST, the value of that element is available in the PHP script as follows:

$_POST["phone"]

Cookies - exclusively exchanged b/w one specific browser & one specific serve

PHP includes convenient support for creating and using cookies. A session is the time span during which a browser interacts with a particular server. A session begins when a browser connects to the server. That session ends either when the browser is terminated or because the server terminated the session because of client inactivity. The length of time a server uses as the maximum time of inactivity is set in the configuration of the server. For example, the default maximum for the Tomcat server is 30 minutes. The HTTP protocol is essentially **stateless**:

One of the most common needs for information about a session is to implement shopping carts on Web sites. Each user's shopping cart is identified by a **session identifier**, which could be **implemented as a cookie**. So, cookies can be used to identify each of the customers visiting the site at a given time. (1) Another common use of cookies is for a Web site to **create profiles of visitors** (2) by remembering which parts of the site are perused by that visitor. Sometimes this is called **personalization**. Later sessions can use such profiles to target advertising to the client in line with the client's past interests.

A **cookie** is a small object of information that includes a name and a textual value. A cookie is created by some software system on the server. At the time it is created, a cookie is assigned a **lifetime.** When the time a cookie has existed reaches its associated lifetime, the cookie is deleted from the browser's host machine.

A cookie is set in PHP with the **setcookie function.** This function takes one or more parameters. The first parameter, which is mandatory, is the **cookie's name** given as a string. The second, if present, is the **new value for the cookie**, also a string. If the value is absent, setcookie undefines the cookie.

The third parameter, when present, is the **expiration time** in seconds for the cookie, given as an integer. The default value for the expiration time is zero, which specifies that the cookie is destroyed at the end of the current session. When specified, the expiration time is often given as the number of seconds.

For Eg:

name value expiration time

setcookie("voted", "true", time() + 86400);

This call creates a cookie named "voted" whose value is "true" and whose lifetime is one day (86,400 is the number of seconds in a day).

In PHP, cookie values are treated much as are form values. All cookies that arrive with a request are placed in the implicit **$_COOKIES array**, which has the cookie names as keys and the cookie values as values. A PHP script can test whether a cookie came with a request by using the **IsSet predicate function** on the associated variable.

Session Tracking

Rather than using one or more cookies, a single session array can be used to store information about the previous requests of a client during a session. session arrays often store a unique session ID for a session. One significant way that session arrays differ from cookies is that they can be stored on the server, whereas cookies are stored on the client.

session_start function, which takes no parameters. The first call to session_start in a session causes a session ID to be created and recorded. On subsequent calls to session_start in the same session, the function retrieves the $_SESSION array, which stores any session variables and their values. Session key—value pairs are created or changed by assignments to the $_SESSION array. They can be destroyed with the **unset operator**.

CONTENT BEYOND SYLLABUS

JavaServer Pages

JavaServer Pages (**JSP**) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems. JSP is similar to PHP and ASP, but it uses the Java programming language. To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat is required. Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime, therefore JSP is a Servlet; each JSP servlet is cached and re-used until the original JSP is modified.

JSP can be used independently or as the view component of a server-side model–view–controller design, normally with JavaBeans as the model and Java servlets (or a framework such as Apache Struts) as the controller. This is a type of Model 2 architecture.

JSP allows Java code and certain predefined actions to be interleaved with static web markup content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java byte code rather than machine code. Like any other Java program, they must be executed within a Java virtual machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment. JSPs are usually used to deliver HTML and XML documents, but through the use of Output Stream, they can deliver other types of data as well.

The Web container creates JSP implicit objects like request, response, session, application, config, page, pageContext, out and exception. JSP Engine creates these objects during translation phase.

Syntax

JSP pages use several delimiters for scripting functions. The most basic is **<% ... %>**, which encloses a JSP *scriptlet.* A scriptlet is a fragment of Java code that is run when the user requests the page. Other common delimiters include **<%= ... %>** for *expressions,* where the scriptlet and delimiters are replaced with the result of evaluating the expression, and *directives*, denoted with **<%@ ... %>**.[5]

Java code is not required to be complete or self-contained within a single scriptlet block. It can straddle markup content, provided that the page as a whole is syntactically correct. For example, any Java *if/for/while* blocks opened in one scriptlet must be correctly closed in a later scriptlet for the page to successfully compile.

Content that falls inside a split block of Java code (spanning multiple scriptlets) is subject to that code. Content inside an *if* block will only appear in the output when the *if* condition evaluates to true. Likewise, content inside a loop construct may appear multiple times in the output, depending upon how many times the loop body runs.

The following would be a valid for loop in a JSP page:

```
<p>Counting to three:</p>
<% for (int i=1; i<4; i++) { %>
   <p>This number is <%= i %>.</p>
<% } %>
<p>OK.</p>
```

The output displayed in the user's web browser would be:

```
Counting to three:

This number is 1.

This number is 2.

This number is 3.

OK.
```

Expression Language

Version 2.0 of the JSP specification added support for the Expression Language (EL), used to access data and functions in Java objects. In JSP 2.1, it was folded into the Unified Expression Language, which is also used in JavaServer Faces.

An example of EL syntax:

The value of "variable" in the object "javabean" is ${javabean.variable}.

Additional tag

The JSP syntax add additional tags, called JSP actions, to invoke built-in functionality.[Additionally, the technology allows for the creation of custom JSP *tag libraries* that act as extensions to the standard JSP syntax.[7] One such library is the JSTL, with support for common tasks such as iteration and conditionals (the equivalent of "for" and "if" statements in Java.)

Compiler

A **JavaServer Pages compiler** is a program that parses JSPs, and transforms them into executable Java Servlets. A program of this type is usually embedded into the application server and run automatically the first time a JSP is accessed, but pages may also be precompiled for better performance, or compiled as a part of the build process to test for errors

Some JSP containers support configuring how often the container checks JSP file timestamps to see whether the page has changed. Typically, this timestamp would be set to a short interval (perhaps seconds) during software development, and a longer interval (perhaps minutes, or even never) for a deployed Web application

Advantages of JSP

There are plenty advantages of using JSP. In general, JSP allows developers to easily distribute application functionality to a wide range of page authors. These authors do not have to know the Java programming language or know anything about writing servlet code, so they can concentrate on writing their HTML code while you concentrate on creating your objects and application logic.

- JSP pages easily combine static templates, including HTML or XML fragments, with code that generates dynamic content.
- JSP pages are compiled dynamically into servlets when requested, so page authors can easily make updates to presentation code. JSP pages can also be precompiled if desired.
- JSP tags for invoking JavaBeans components manage these components completely, shielding the page author from the complexity of application logic.
- Developers can offer customized JSP tag libraries that page authors access using an XML-like syntax.
- Web authors can change and edit the fixed template portions of pages without affecting the application logic. Similarly, developers can make logic changes at the component level without editing the individual pages that use the logic.